Titre: Title:	Planification et optimisation des points de présence dans les réseaux IP
Auteur: Author:	Marc St-Hilaire
Date:	2003
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	St-Hilaire, M. (2003). Planification et optimisation des points de présence dans les réseaux IP [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/6994/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/6994/
Directeurs de recherche: Advisors:	Samuel Pierre, & Steven Chamberland
Programme: Program:	Génie électrique

UNIVERSITÉ DE MONTRÉAL

PLANIFICATION ET OPTIMISATION DES POINTS DE PRÉSENCE DANS LES RÉSEAUX IP

MARC ST-HILAIRE
DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

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

AVRIL 2003

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

PLANIFICATION ET OPTIMISATION DES POINTS DE PRÉSENCE DANS LES RÉSEAUX IP

présenté par : ST-HILAIRE Marc
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. GAGNON Michel, Ph.D., président
M. PIERRE Samuel, Ph.D., membre et directeur de recherche
M. CHAMBERLAND Steven, Ph.D., membre et codirecteur de recherche
M. QUINTERO Alejandro, Ph.D., membre

REMERCIEMENTS

Mes premiers remerciements vont naturellement à mes deux directeurs M. Steven Chamberland et M. Samuel Pierre. Chacun, avec sa rigueur intellectuelle, a joué un rôle important dans la complexité de ce projet. Sans leur encadrement et leurs nombreux conseils, cette recherche n'aurait pas vu le jour.

Ma reconnaissance va aussi à ma mère et mon père dont l'amour, l'éducation et le support moral m'ont permis d'être ce que je suis aujourd'hui.

À mon frère Bruno, à Lise, à Robert et à toute ma famille, je vous dis merci pour votre soutien et vos encouragements.

Pour terminer, je ne peux passer sous le silence ma compagne dévouée qui a fait énormément pour moi durant cette recherche. Merci Catherine d'avoir enduré mes sauts d'humeurs et mes absences. Merci pour tout. Je t'en suis grandement reconnaissant.

RÉSUMÉ

Avec une demande croissante pour les services Internet, les fournisseurs de service de télécommunications n'ont d'autre choix que d'investir d'importantes sommes d'argent dans leurs réseaux IP (*Internet Protocol*). Avec ces investissements, le nombre de routeurs augmente sans cesse et plusieurs de ceux-ci sont co-localisés formant des points de présence.

Dans ce mémoire, nous commençons par présenter un modèle de programmation mathématique afin de résoudre le problème d'optimisation des points de présence dans les réseaux IP. Ce modèle consiste à sélectionner le nombre de routeurs et leur type (où le type d'un routeur est caractérisé par son nombre de fentes et sa capacité), sélectionner le type des cartes d'interfaces (où le type d'une carte d'interface est caractérisé par la technologie, la vitesse de chaque port et le nombre de ports) et finalement, connecter les liens d'accès et les liens dorsaux aux ports. Le but est de minimiser le coût total du point de présence. Dans ce document, nous supposons que la topologie des routeurs co-localisés a été fixée par le planificateur de réseau ainsi que le type de liens utilisés pour les interconnecter.

Étant donné que ce problème est NP-complet, il est peu probable que nous soyons capable de résoudre des problèmes de grande taille en temps raisonnable. En effet, avec une implantation commerciale de l'algorithme de séparation et évaluation progressive, nous avons remarqué que le temps d'exécution était beaucoup trop long et que même la mémoire de l'ordinateur venait à manquer et ce, même pour des problèmes de taille moyenne. C'est pour cette raison que nous avons développé une approche heuristique qui sera en mesure de trouver de bonnes solutions pour des problèmes de toutes tailles en un temps raisonnable.

Afin d'évaluer l'efficacité de notre heuristique, nous allons la comparer avec une borne inférieure trouvée en résolvant une version relaxée du modèle. Notre heuristique a été testée pour différents problèmes générés aléatoirement. Les résultats obtenus avec

cette méthode sont quasi optimaux. En effet, la moyenne des écarts est de 1.6%. De plus, nous avons remarqué que plus le nombre de nœuds connectés au point de présence est élevé, meilleur est le comportement de notre heuristique. En ce qui concerne le temps d'exécution, notre heuristique est beaucoup plus performante que la borne inférieure.

ABSTRACT

With an increasing demand for Internet network services, telecommunication service providers are constantly investing in their Internet protocol (IP) network infrastructure. As a result, several backbone routers are co-located in the same central office forming large IP network point of presence.

In this document, we first propose a mathematical programming model for the point of presence optimization problem in IP networks. It consists of selecting the number of routers and their types (where a router type is characterized by its number of slots and its switch fabric capacity), selecting the interface card types (where an interface card type is characterized by its technology, its port rate and its number of ports) and finally, connecting the access and the backbone links to the ports. The goal is to find the minimum cost point of presence. In this paper, we suppose that the topology of the network between the co-located routers is fixed by the network planner as well as the link types used to interconnect these routers.

Since this problem is NP-hard, it is unlikely that large size instances of this problem can be solved to optimality. In fact, with a commercial implementation of the branch and bound algorithm, we found that it takes too much time to solve it and the computer also run out of memory, even for real size instance of the problem. This disadvantage creates the need for a heuristic approach that will always work even for large size instances of the problem.

In order to prove the efficiency of the heuristic, we compare it with a lower bound found by solving a relaxed version of the model using branch and bound. The heuristic was tested for a set of randomly generated instances and we found that quasi-optimal solutions are obtained.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ANNEXES

LISTE DES ABRÉVIATIONS

Abréviations	Signification
BB	Branch and Bound
Gbps	Giga bits par seconde
GHz	Giga Hertz
IP	Internet Protocol
Mbps	Mega bits par seconde
OC	Optical Carrier
OPL	Optimization Programming Language
O-POP	Other POP
OSPF	Open Shortest Path First
PCC	Plus Court Chemin
POP	Point Of Presence
RIP	Routing Information Protocol
SONET	Synchronous Optical NETwork

CHAPITRE 1

INTRODUCTION

Depuis plusieurs années, nous assistons à une forte augmentation de la demande pour les services Internet. Pour satisfaire cette demande, les fournisseurs de services de télécommunications n'ont d'autre choix que d'investir d'importantes sommes d'argent dans leurs réseaux IP (*Internet Protocol*). Avec ces investissements massifs, les fournisseurs se retrouvent avec plusieurs routeurs rapprochés situés dans le même bureau central, formant ainsi un point de présence. Il s'avère opportun pour ces fournisseurs de services de diminuer au maximum leurs coûts afin de demeurer le plus compétitif possible. Considérant que le prix des routeurs et des cartes d'interfaces est encore très élevé, il est impératif de disposer d'une méthode d'optimisation pour les différents points de présence, méthode qui doit également permettre de résoudre des problèmes de taille variable en un temps raisonnable. L'objet de ce mémoire est de proposer une méthode efficace pour minimiser le coût de ces points de présence. Dans ce chapitre d'introduction, nous présentons les concepts de base qui serviront à énoncer les éléments de la problématique qui nous préoccupe. Par la suite, nous précisons les objectifs de recherche de même que la méthodologie qui sera utilisée afin d'atteindre ces objectifs. Enfin, nous terminons par une esquisse des grandes lignes de ce mémoire.

1.1 Définitions et concepts de base

Un réseau informatique peut être vu comme un ensemble de nœuds (ordinateurs, commutateurs, routeurs, etc.) et un ensemble de liaisons (sans fil, câbles à paires torsadées, câbles de fibre optique, etc.) qui interconnecte les nœuds (Pierre et Elgibaoui, 1997). Le but du réseau est d'assurer la communication entre ces différents matériels géographiquement dispersés.

Un commutateur est un élément d'interconnexion comprenant plusieurs ports permettant de relier (ou d'isoler) les échanges entre les ordinateurs du réseau sans pratiquement aucune perte de bande passante.

Pour acheminer un paquet à destination, nous avons besoin d'un protocole de routage, permettant d'établir une route entre un nœud d'origine et un nœud de destination. Le routage est l'opération qui consiste à faire passer les paquets IP d'un noeud à un autre. Un protocole de routage peut être basé sur différents facteurs comme le nombre de sauts, le coût des liens, etc. Cette opération est assurée par un nœud spécialisé que l'on appelle routeur.

Un routeur est un dispositif matériel ou logiciel qui sert à diriger les données informatiques à travers un réseau. Leur rôle, lorsque les données se présentent sur une interface d'entrée, est de trouver pour elles une interface de sortie (Tanenbaum, 1996). Ils assurent le routage des paquets IP jusqu'à leur destination. Chaque routeur est caractérisé par sa capacité (*switch fabric capacity*) et par le nombre de fentes qu'il possède. Les fentes servent à insérer différents types de cartes d'interfaces dans le routeur. Ces deux caractéristiques sont des contraintes à respecter quand nous utilisons un routeur puisqu'elles sont fixées par le fabricant.

Une carte d'interface est un dispositif matériel que l'on insert dans un routeur. La connexion physique entre le routeur et le « client » s'effectue à travers ces cartes d'interface. Une carte est normalement caractérisée par sa technologie, par le nombre de ports qu'elle possède et par la vitesse de chaque port. Une carte peut supporter un nombre de connexions qui est égal ou inférieur au nombre de ports qu'elle possède. Par exemple, une carte qui possède quatre ports peut supporter au maximum quatre connexions.

Lorsqu'on parle de technologie de la carte d'interface, on fait référence typiquement à la norme SONET (*Synchronous Optical NETwork*) qui inclut un ensemble de vitesses de signal qui sert à transmettre les signaux numériques sur la fibre optique. La vitesse de base (OC-1) est de 51.84 Mbps. Cependant, certains multiples de la vitesse de base sont utilisés dans les équipements de communication. Ce sont ces

multiples que nous utiliserons tout au long de ce mémoire. Ils sont décrits dans le Tableau 1.1.

Tableau 1.1 Les différents types de technologie de transmission utilisés

Technologie	Capacité
OC-3	155.52 Mbps
OC-12	622.08 Mbps
OC-48	2.488 Gbps
OC-192	10 Gbps

Un autre concept est le point de présence ou POP (*Point Of Presence*). Un POP est un ensemble de routeurs co-localisés (habituellement dans le même local) qui sont interconnectés entre eux. Plusieurs clients sont généralement connectés à un POP par des liens d'accès de type OC-3 ou OC-12. Les POPs peuvent aussi être connectés à d'autres POPs (O-POPs, *Other* POPs) par des liens dorsaux. Ces liens sont habituellement de plus grande capacité car le trafic est généralement plus important. C'est l'ensemble de tous les POPs qui forment un réseau. La Figure 1.1 illustre une architecture typique d'un réseau IP. Nous pouvons remarquer la composition d'un POP ainsi que l'interconnexion de celui-ci avec le reste du réseau.

Il est très important de bien distinguer la notion de trafic et celle de flot. On parle de trafic lorsqu'on fait référence au nombre de bits échangés entre le nœud d'origine et le nœud de destination. Par contre, un flot est la somme de tous les trafics (origine – destination) passant sur un lien reliant deux nœuds successifs. Par exemple, dans la Figure 1.2, nous avons un premier trafic entre les nœuds « a » et « c » et un deuxième entre les nœuds « b » et « d ». Ces trafics génèrent un flot entre « a » et « b » qui est égal au trafic # 1, un flot entre « c » et « d » qui est égal au trafic # 2 et un flot entre « b » et « c » qui est égal à la somme des trafic # 1 et # 2.

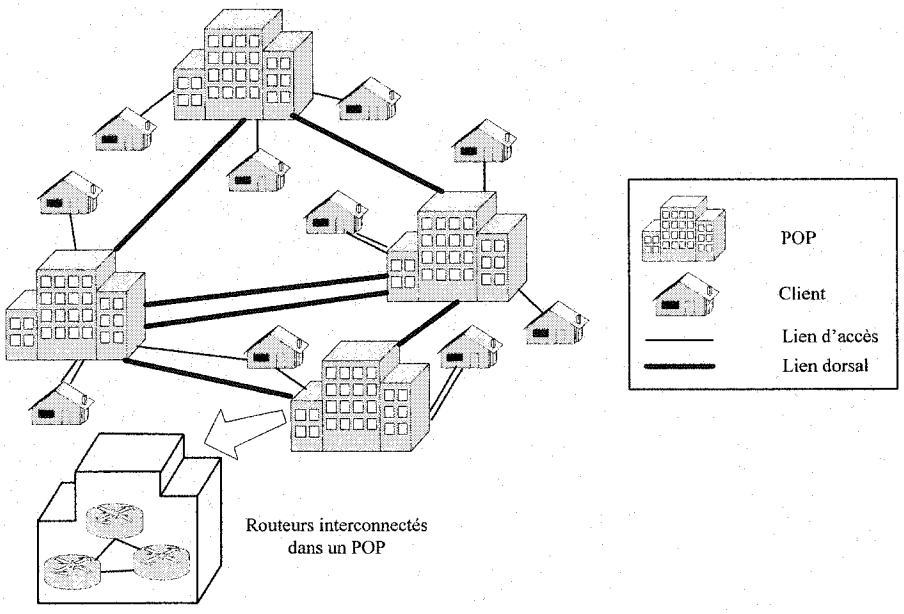


Figure 1.1 Architecture d'un réseau IP

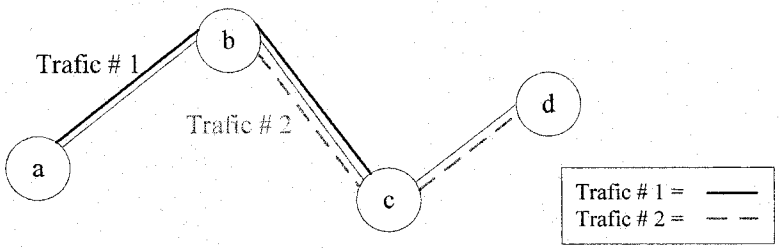
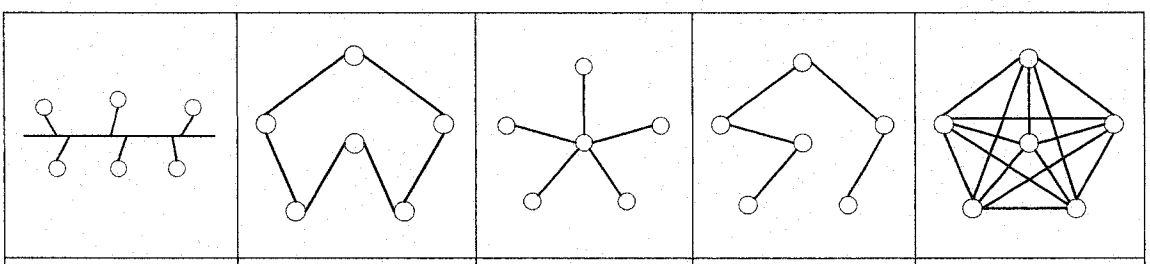


Figure 1.2 Trafic vs flot

Un autre concept est celui de la topologie du réseau. La topologie d'un réseau se réfère à l'ensemble des liens qui connecte les nœuds ensemble (Pierre et Legault, 1998). Comme le démontre la Figure 1.3, il existe cinq grands types de topologies élémentaires. Il s'agit du bus, de l'anneau, de l'étoile, de l'arbre et du maillage. Il existe aussi des topologies qui sont formés de combinaison des topologies élémentaires. La conception topologique consiste à trouver une topologie qui satisfait des contraintes reliées à la qualité de service ainsi qu'à la fiabilité à un coût minimum (Pierre et Elgibaoui, 1997).



Bus	Anneau	Étoile	Arbre	Maillage

Figure 1.3 Types de topologies

Ceci est un problème bien connu pour sa difficulté. En effet, pour un réseau de n nœuds, nous avons $2^{1/2 \cdot n(n-1)}$ topologies possibles. Pour un n grand, il est donc irréaliste de penser générer exhaustivement toutes ces topologies.

Enfin, pour simplifier sa résolution, la planification d'un réseau peut être divisée en trois tâches différentes. Premièrement, on résout la configuration topologique, puis on détermine le routage permettant d'affecter les flots aux liaisons, et finalement on affecte les capacités aux liaisons.

Tous ces concepts servent d'ancrage à la réalisation de ce travail de recherche. Ils viennent en préciser les contours et permettent de comprendre les éléments de la problématique.

1.2 Éléments de la problématique

La planification et le dimensionnement des réseaux de télécommunications sont des aspects extrêmement importants dans tous les réseaux. Évidemment, l'importance de ces aspects est proportionnelle à la taille du réseau. Cependant, plus la taille du réseau est importante, plus il est difficile de trouver la topologie idéale. Il en va de même pour la planification des POPs. Plus il y a des clients et des POPs dans le réseau, plus il est difficile de trouver une solution optimale. Avec un nombre élevé de composants, les méthodes d'optimisation énumératives sont complètement dépassées. Par exemple, un réseau de seulement 12 nœuds nous donne $2^{(1/2)\cdot 12\cdot(12-1)} = 2^{66}$ topologies différentes. Si un ordinateur prend 1 ns pour tester une solution, nous aurions besoin de plus de 2300

années pour tester toutes ces possibilités. Il apparaît donc impérieux de disposer de méthodes plus efficaces capables de trouver une topologie de coût minimum malgré un grand nombre d'alternatives.

Si une solution optimale qui reflète la réalité est recherchée, le modèle topologique n'est alors pas suffisant. Nous devons également tenir compte du trafic qui circule à travers ce POP. En effet, le trafic vient ajouter de la complexité au modèle topologique de base. Le trafic va influencer le nombre de liens qui est installé entre les routeurs d'un même POP. Par exemple, si le flot qui circule entre deux routeurs du même POP est plus grand que la capacité du lien, nous devrons alors ajouter un ou plusieurs liens supplémentaires.

La méthode optimale en serait une de type énumérative, en ce sens que dans le pire des cas, toutes les possibilités réalisables seront calculées pour finalement retenir celle dont le coût est minimum. Cependant, quand un POP a beaucoup de connexions cette méthode devient inefficace. Nous démontrerons que ce problème est un problème NP-complet. Cela signifie qu'il est peu probable de trouver un algorithme exact pour trouver des solutions rapidement pour des problèmes de grande taille. Le temps d'exécution de la méthode optimale augmente exponentiellement avec la taille du problème. C'est pourquoi une méthode heuristique (approximative), dont le temps d'exécution varie peu en fonction de la taille du problème, apparaît plus appropriée. Cette méthode nous permettra de traiter des problèmes de très grande taille en un temps raisonnable, même si la solution obtenue n'est pas optimale. Lorsqu'elle est efficace, une méthode heuristique génère des résultats dont le coût est légèrement supérieur ou égal à celle de la solution optimale.

Ces éléments de la problématique nous amènent à nous poser les trois questions suivantes :

- Serons-nous en mesure de proposer un modèle qui représentera bien la réalité des réseaux d'aujourd'hui ?
- Est-ce que notre heuristique génèrera des résultats qui seront acceptables comparativement à la méthode optimale ?

- Serons-nous en mesure de résoudre des problèmes de toutes tailles en un temps raisonnable ?

C'est sur ces constats qu'il est permis d'établir les objectifs de cette recherche.

1.3 Objectifs de recherche

L'objectif principal de ce travail est de mettre au point un procédé pour résoudre le problème d'optimisation des POPs dans les réseaux IP. Plus précisément, nous visons à :

- concevoir un modèle de programmation mathématique qui résout le problème d'optimisation de façon exacte (au prix d'efforts calculatoires éventuellement prohibitifs);
- créer une méthode heuristique pour traiter des problèmes de grande taille en un temps raisonnable;
- évaluer la performance de notre heuristique en la comparant avec une borne inférieure.

Pour déterminer la manière dont ces objectifs seront atteints, la section suivante esquisse la méthodologie qui sera utilisée.

1.4 Esquisse méthodologique

Pour atteindre les objectifs, nous commencerons par implanter une méthode optimale qui servira de référence. Cette méthode sera implémentée à l'aide d'un langage d'optimisation OPL (*Optimization Programming Language*) et résolue par séparation et évaluation progressive (ILOG, 2000). Par la suite, nous élaborerons une méthode heuristique qui générera des résultats qui seront le plus près possible de la méthode optimale. Le but de ces deux méthodes est de minimiser le coût total du POP.

Par la suite, nous effectuerons une série de tests afin de vérifier le comportement des algorithmes. Pour démontrer que notre heuristique est efficace, nous la comparerons avec une borne inférieure. Cette borne sera obtenue en relaxant un certain nombre de

contraintes du modèle de programmation mathématique. C'est ainsi que nous pourrons évaluer la performance de notre heuristique par rapport au coût minimum d'une certaine topologie. Nous pensons qu'une différence de moins de cinq pourcent serait plus qu'acceptable.

1.5 Plan du mémoire

Outre ce chapitre d'introduction, ce mémoire comprend trois autres chapitres. Le deuxième chapitre présente les modèles et les algorithmes d'optimisation des POPs dans les réseaux IP. De plus, le deuxième chapitre comporte une section portant sur la littérature du domaine de la planification et de l'optimisation des réseaux. Le chapitre trois présente les détails d'implantation et les résultats; nous y développons un exemple détaillé suivi des résultats de plusieurs problèmes générés aléatoirement ainsi que l'analyse des résultats obtenus. Le quatrième chapitre, en guise de conclusion, effectue une synthèse des travaux qui ont été effectués, des résultats obtenus et expose les limitations du modèle et ouvre la porte à des travaux futurs.

CHAPITRE 2

MODÈLES ET ALGORITHMES

Dans ce chapitre, nous présenterons un modèle de programmation mathématique et une méthode heuristique pour le problème de la planification et d'optimisation des POPs dans les réseaux IP. Le modèle a pour objectif de minimiser le coût total du POP. Pour ce faire, nous commencerons par présenter une revue des travaux antérieurs portant sur la planification et l'optimisation des réseaux. Ensuite, nous présenterons une vue d'ensemble du processus de conception d'un réseau et nous effectuerons un bref rappel de la problématique. Après avoir énoncé les hypothèses, nous présenterons le modèle de programmation mathématique. Afin de mieux définir ce modèle, nous exposerons la notation utilisée ainsi que la fonction de coût de ce dernier. Ensuite, nous étudierons la complexité du problème. Puisque le problème est NP-complet, nous proposerons une méthode heuristique afin d'être capable de traiter des problèmes de grande taille en un temps raisonnable.

2.1 Revue des travaux antérieurs

Depuis plusieurs années, la demande pour les réseaux de télécommunications ne cesse d'augmenter. Avec cette demande grandissante, plusieurs chercheurs se sont penchés sur des méthodes afin de planifier et de concevoir des réseaux performants tout en minimisant les coûts. Cette tâche est un problème difficile surtout pour des réseaux de grande taille. Cependant, personne n'a encore étudié la planification et la conception des points de présence dans les réseaux IP comme nous nous apprêtons à le faire.

La problématique la plus proche est celle de la planification des réseaux à deux niveaux ou aussi appelé planification conjointe des réseaux tributaires (réseau d'accès) et dorsal (*backbone*). Commençons tout d'abord par définir ce que nous entendons par réseau tributaire et réseau dorsal. Un réseau tributaire connecte plusieurs nœuds (clients)

à un point d'accès (aussi appelé concentrateur, commutateur, etc.). C'est en reliant tous les points d'accès qu'on forme le réseau dorsal. La Figure 2.1 représenté ce type d'architecture ou les nœuds A, B, C, D et E représentent les points d'accès. Les réseaux dorsaux et tributaires peuvent être représentés par différentes topologies (anneau, étoile, arbre, etc.). Par exemple, le réseau tributaire associé avec le nœud D est un arbre.

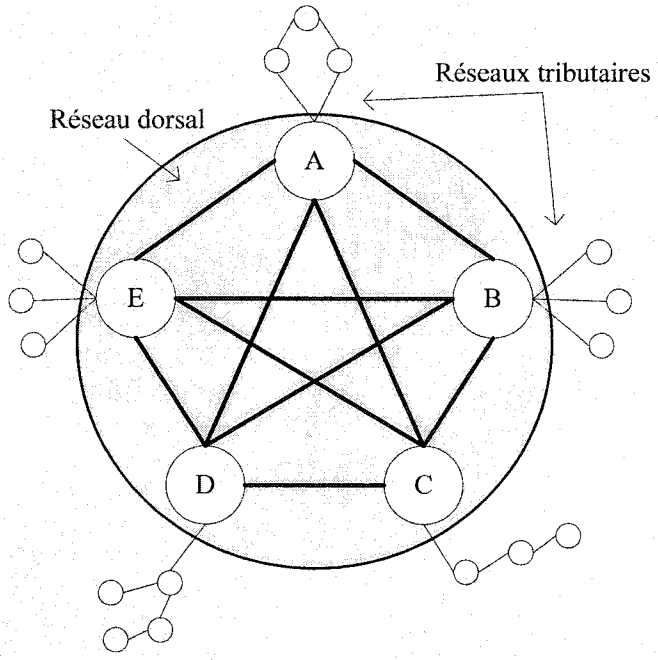


Figure 2.1 Réseau tributaire et réseau dorsal

Plusieurs modèles utilisant des structures en couches ont été développés dans les dernières années. En effet, une revue de littérature de Klincewicz (1998) comporte une liste exhaustive des travaux effectués avant 1998 sur la planification conjointe des réseaux tributaires et dorsal. Plus récemment, Chamberland et al. (2000a, 2000b, 2001) ont proposé des heuristiques pour ce problème de planification.

Considérons une architecture de réseau dont le réseau tributaire a une topologie en étoile et le réseau dorsal une topologie maillée. La Figure 2.2 représente cette architecture.

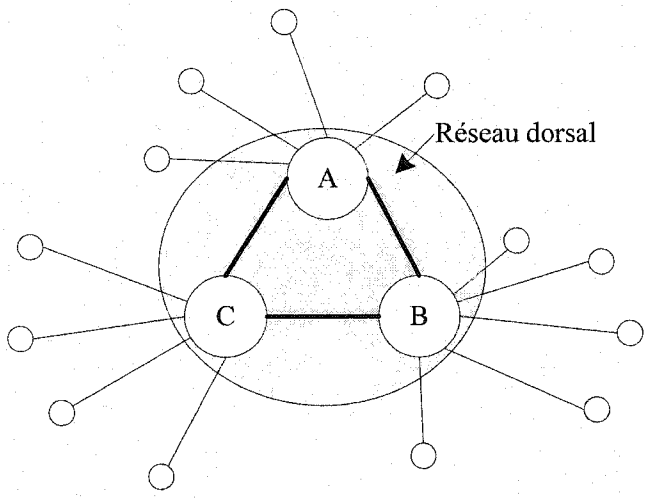


Figure 2.2 Réseau de type étoile – maillé

En regardant plus attentivement la Figure 2.2, nous pouvons constater que cette architecture comporte plusieurs similitudes à celle de notre projet. En effet, le réseau dorsal peut être considéré comme étant un point de présence dans lequel plusieurs routeurs sont installés. La seule différence réside dans la localisation des routeurs. Dans un réseau dorsal, les routeurs peuvent être dispersés géographiquement tandis que dans un POP, les routeurs sont tous localisés dans le même local. Le problème de la localisation des points d'accès ne se pose donc plus. Notre but est de minimiser le coût total du POP. La seule optimisation possible est dans le choix des routeurs, le choix des cartes d'interface et le dimensionnement des liens.

En ce qui concerne les réseaux tributaires, ils représentent tous les clients et tous les autres POPs qui sont connecté au POP que nous voulons optimiser. Aucune optimisation n'est possible dans cette partie puisque nous assumons que chaque client et chaque autre POP possèdent déjà leurs liens afin de se connecter au POP. La Figure 2.3 représente comment cette architecture peut ressembler au problème d'optimisation des points de présence.

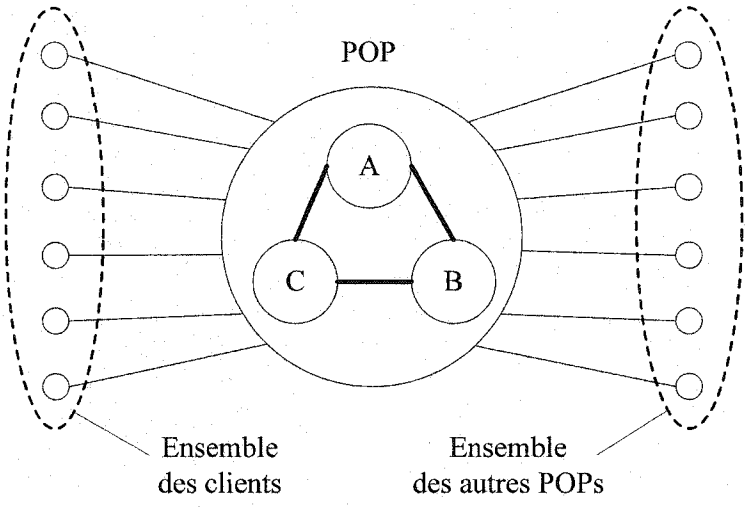


Figure 2.3 Point de présence

Cependant, pourquoi tous les travaux portant sur les modèles en couche ne répondent-ils pas à nos besoins? Simplement parce que les points suivants n'ont jamais été considérés conjointement.

- Problème de conception topologique des réseaux tributaires et dorsal;
- Problème du dimensionnement des liens et des commutateurs;
- Utilisation de commutateurs modulaires supportant différentes technologies de transport et plusieurs ports par cartes d'interfaces.

Après avoir analysé les travaux antérieurs, nous allons maintenant décrire le processus de conception d'un réseau

2.2 Processus de conception d'un réseau

Dans cette section, nous effectuerons un bref survol du processus de conception d'un réseau en général. La conception d'un réseau est basée sur un mécanisme itératif qui comprend plusieurs étapes. La première étape consiste à déterminer les conditions que le réseau doit satisfaire. Ceci implique inévitablement une recherche d'information sur les prix, les performances et les capacités des équipements à acheter (routeurs, cartes d'interface, liens, etc.). Par exemple, une carte d'interface peut avoir quatre ports de type

OC-3 pour un prix de 3000$. De plus, nous devons estimer le trafic engendré par chaque utilisateur afin d'avoir une idée globale de la charge du trafic dans le réseau. Il est évident que si le type de trafic est constitué de simples données, la charge sera probablement moins élevée que si des vidéos sont transférés. Ces informations seront utiles afin de concevoir un modèle qui représentera le plus fidèlement possible la réalité. De façon générale, il est plus facile de recueillir des données statiques que des données dynamiques. En effet, les données statiques demeurent constantes dans le temps tandis que les données dynamiques peuvent changer avec le temps. Par exemple, le trafic est un type de donnée dynamique car il varie en fonction du temps. Toutes ces informations sont utilisées dans la deuxième étape qui est celle du processus de conception topologique et dimensionnement. Plusieurs algorithmes peuvent être utilisés pour cela. Ces méthodes consistent à sélectionner les liens, à effectuer le routage du trafic et à déterminer la capacité des liens. Après qu'une solution ait été développée, elle doit être analysée afin de mesurer les paramètres qui intéressent le planificateur du réseau. Par exemple, les paramètres d'intérêts sont habituellement le coût, la fiabilité, l'utilisation ainsi que le délai moyen dans le réseau. Cette étape est appelée analyse des performances. Ces trois étapes constituent une itération dans le processus de conception d'un réseau. Par la suite, une autre itération peut débuter soit en modifiant des entrées, soit en utilisant une autre approche de conception (Mann-Rubinson et Terplan, 1998). Tout ce processus est résumé à la Figure 2.4.

Le but d'une approche itérative est de générer plusieurs topologies différentes parmi lesquelles nous choisirons celle qui répond le mieux à nos besoins. Malheureusement, cette tâche peut être très difficile si aucun outil informatique n'est utilisé. Même en présence d'outils informatiques, il est pratiquement impossible de générer la topologie optimale tellement le nombre de possibilités est grand. C'est pour cette raison que nous avons recours aux méthodes heuristiques. Celles-ci génèrent des solutions approximatives en un temps raisonnable. Une fois que l'heuristique a généré une topologie, il peut être très payant d'utiliser une autre méthode pour raffiner les détails de cette solution.

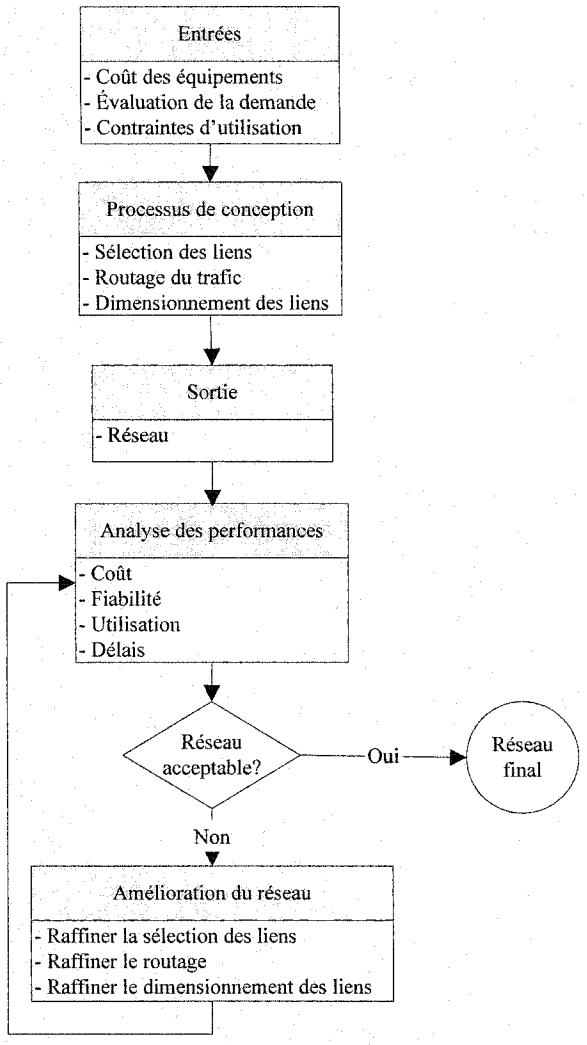


Figure 2.4 Vue d'ensemble du processus de conception de réseaux

Juste avant de présenter les modèles, il serait important de faire un bref rappel de notre problématique afin de mieux comprendre le besoin de ces méthodes.

2.3 Rappel de la problématique

Comme nous l'avons indiqué dans le premier chapitre, plus il y a de clients et de POPs connectés à un autre POP, plus il est difficile de trouver une solution optimale. Avec un nombre élevé de composants, il devient essentiel de disposer de méthodes

efficaces afin de trouver une topologie de coût minimal. Comme nous le démontrerons, certaines méthodes donnent des résultats optimaux au détriment d'effort calculatoire prohibitif. Ces méthodes sont généralement très coûteuses en terme de temps d'exécution et de mémoire utilisée. Par contre, d'autres méthodes moins précises ont l'avantage de générer des résultats en un temps raisonnable. Ces deux approches seront présentées dans la suite de ce chapitre.

Pour construire un modèle, certaines hypothèses doivent être avancées afin de mieux définir le cadre de travail de ce dernier. La section suivante présente les hypothèses ainsi que les informations qui nous sont connues.

2.4 Hypothèses

Dans notre cas, nous faisons les trois hypothèses suivantes sur l'organisation de chaque POP :

- H1) Un POP est composé d'un ou plusieurs routeurs co-localisés (dans le même local) et interconnectés.
- H2) Le nombre de routeurs installés dans un POP ne peut pas excéder le nombre maximal alloué.
- H3) La somme des vitesses de transmission des ports installés dans un routeur ne peut excéder la capacité de ce routeur (*switch fabric capacity*).

De plus, nous supposons que l'information suivante est connue :

- I1) Les clients qui sont connectés au POP.
- I2) Les autres POPs qui sont connectés au POP que l'on désire optimiser.
- I3) Les types de liens d'accès et liens du réseau dorsal qui sont connectés au POP.

I4) Le nombre maximum de routeurs pouvant être installés dans le POP.

I5) Les différents types de routeurs pouvant être installés dans le POP.

I6) Les différents types de cartes d'interface.

I7) Le coût d'achat de chaque type de routeur ainsi que le coût d'installation (incluant l'espace, les câbles, le support et la main d'oeuvre).

I8) Le coût d'achat de chaque type de carte d'interface ainsi que le coût d'installation (incluant les câbles et la main d'oeuvre).

I9) Le coût pour interconnecter deux routeurs dans le même POP (incluant les câbles et la main d'oeuvre);

I10) Le trafic qui circule dans le POP.

Les deux modèles qui seront proposés se basent sur ces hypothèses et ces informations. De façon générale, les deux méthodes utilisent un procédé similaire qui consiste à :

- sélectionner le nombre et le type de routeurs (caractérisé par le nombre de fentes et la capacité) à installer dans le POP;
- sélectionner le nombre et le type de cartes d'interface (caractérisé par la technologie, le nombre de ports, la vitesse de chaque port ainsi que le nombre de fentes nécessaires pour installer la carte dans le routeur);
- connecter les liens aux ports;
- sélectionner la topologie et le type de liens pour interconnecter les routeurs.

Nous commencerons donc par présenter le modèle de programmation mathématique. Après avoir évalué sa complexité, nous exposerons la méthode heuristique.

2.5 Modèle de programmation mathématique

De façon générale, le modèle de programmation mathématique est un programme dans lequel nous devons minimiser une fonction objectif tout en respectant un certain nombre de contraintes. Plusieurs logiciels d'optimisation sont disponibles sur le marché pour faciliter la résolution de ces problèmes. Afin d'utiliser un langage de programmation, nous devons définir une notation spécifique qui modélisera le problème. Après avoir exposé la notation, nous expliquerons la fonction objectif (qui est la fonction de coût), nous présenterons le modèle mathématique et nous analyserons la complexité de ce dernier.

2.5.1 La notation

La notation suivante est utilisée dans le modèle de programmation mathématique. La notation est composée d'ensembles, de variables de décision, de paramètres de coût ainsi que de constantes.

Les ensembles

- R, l'ensemble des liens et des types de port (où α_r est la vitesse de transmission (en Mbps) du lien/port de type $r \in R$) tel que $R = R^A \cup R^B \cup R^P$ où R^A est l'ensemble des types de lien/ports utilisés dans le réseau d'accès, R^B est l'ensemble des types de liens/ports utilisés dans le réseau dorsal et R^P est l'ensemble des types de liens/ports utilisés pour interconnecter les routeurs dans le POP;
- I, l'ensemble des clients connectés au POP;
- J, l'ensemble des autres POPs (dénommé O-POPs pour *Other* POPs) présents dans le réseau. C'est-à-dire les POPs dans le réseau qui sont connectés au POP que nous voulons optimiser;
- K, l'ensemble des sites potentiels des routeurs dans le POP;

- T, l'ensemble des types de routeur (où m^t est le nombre de fentes que le routeur de type $t \in T$ possède et β^t est sa capacité en Mbps);
- L, l'ensemble des types de carte d'interface (où n^l est le nombre de ports sur la carte de type $l \in L$ et h^l le nombre de fentes nécessaires pour l'insérer dans un routeur) tel que $L = \cup_{r \in R} L_r$ où L_r est l'ensemble des cartes d'interfaces avec les ports de type $r \in R$.

Les variables de décision

- u_k^t, une variable qui peut prendre les valeurs 0 ou 1 tel que $u_k^t = 1$ si et seulement si le routeur de type $t \in T$ est installé au site $k \in K$;
- v_k^l, le nombre de cartes de type $l \in L$ installées au site $k \in K$;
- x_{ik}^r, le nombre de liens de type $r \in R^A$ provenant du client $i \in I$ vers le site $k \in K$;
- y_{jk}^r, le nombre de liens de type $r \in R^B$ provenant du O-POP $j \in J$ vers le site $k \in K$;
- $z_{kk'}^r$, le nombre de liens de type $r \in R^P$ installés entre le site $k \in K$ et le site $k' \in K$ (pour $k < k'$);
- $f_{ik}^o (f_{ki}^o)$, le flot du trafic (en Mbps) du client $i \in I$ vers le site $k \in K$ (du site $k \in K$ vers le client $i \in I$) originaire de $o \in (I \cup J)$;
- $f_{jk}^o (f_{kj}^o)$, le flot du trafic (en Mbps) du O-POP $j \in J$ vers le site $k \in K$ (du site $k \in K$ vers le O-POP $j \in J$) originaire de $o \in (I \cup J)$;
- $f_{kk'}^o$, le flot du trafic (en Mbps) du site $k \in K$ au site $k' \in K$ originaire de $o \in (I \cup J)$.

Les paramètres du coût

- a_k^l, le coût d'une carte de type $l \in L$ et de l'installation au site $k \in K$;

- b_k^t, le coût d'achat d'un routeur de type $t \in T$ et de l'installation au site $k \in K$;
- $c_{kk'}^r$, le coût (incluant le coût de l'installation) pour connecter le site $k \in K$ au site $k' \in K$ avec un lien de type $r \in R^P$.

Les constantes

- p^{od}, le trafic (en Mbps) de $o \in (I \cup J)$ à $d \in (I \cup J) \setminus \{o\}$ passant à travers le POP;
- $q_i^r (q_j^r)$, le nombre de liens de type $r \in R$ de $i \in I$ (de $j \in J$) au POP.

2.5.2 La fonction de coût

La fonction objectif représente le coût total du POP. Ce coût est composé de trois composants : le coût des liens, le coût des routeurs ainsi que le coût des cartes d'interfaces. Voici l'expression détaillée de chacun de ces coûts.

Le coût des liens (noté C_L), représenté par l'équation suivante, inclut le coût des liens utilisés pour interconnecter les routeurs à l'intérieur du POP.

$$C_L(z) = \sum_{r \in R^P} \sum_{k \in K} \sum_{\substack{k' \in K \\ k<k'}} c_{kk'}^r z_{kk'}^r \tag{2.1}$$

Le coût des routeurs (noté C_R), inclut le coût des routeurs ainsi que le coût d'installation de ceux-ci.

$$C_R(u) = \sum_{t \in T} \sum_{k \in K} b_k^t u_k^t \tag{2.2}$$

Le coût des cartes d'interfaces (noté C_I), inclut le coût des cartes d'interfaces ainsi que le coût d'installation de celles-ci.

$$C_I(v) = \sum_{k \in K} \sum_{l \in L} a_k^l v_k^l \tag{2.3}$$

2.5.3 Le modèle mathématique

Le modèle mathématique (noté P) consiste à minimiser la fonction objectif tout en respectant un certain nombre de contraintes. Nous commencerons donc par présenter le modèle. Ensuite, nous expliquerons la signification des différentes contraintes.

P :

$$\min_{f,u,v,x,y,z} C_L(z)+C_R(u)+C_I(v) \tag{2.4}$$

sujet à

Contraintes d'affectation des clients

$$\sum_{k\in K} x_{ik}^r = q_i^r \qquad \left(r\in R^A, i\in I\right) \tag{2.5}$$

Contraintes d'affectation des O-POPs

$$\sum_{k\in K} y_{jk}^r = q_j^r \qquad \left(r\in R^B, j\in J\right) \tag{2.6}$$

Contraintes d'unicité des types de routeurs

$$\sum_{t\in T} u_k^t \le 1 \qquad \left(k\in K\right) \tag{2.7}$$

Contraintes de capacité des routeurs (au niveau des fentes)

$$\sum_{l\in L} h^l v_k^l \le \sum_{t\in T} m^t u_k^t \qquad \left(k\in K\right) \tag{2.8}$$

Contraintes de capacité des routeurs (au niveau de la capacité de commutation)

$$\sum_{r\in R} \alpha_r \sum_{l\in L_r} n^l v_k^l \le \sum_{t\in T} \beta^t u_k^t \qquad \left(k\in K\right) \tag{2.9}$$

Contraintes de capacité des routeurs (au niveau des ports)

$$\sum_{i\in I} x_{ik}^r \le \sum_{l\in L_r} n^l v_k^l \qquad \left(r\in R^A \setminus \left(R^B\cup R^P\right), k\in K\right) \tag{2.10}$$

$$\sum_{j\in J} y_{jk}^r \le \sum_{l\in L_r} n^l v_k^l \qquad \left(r\in R^B \setminus \left(R^A\cup R^P\right), k\in K\right) \tag{2.11}$$

$$\sum_{\substack{k'\in K\\ k<k'}} z^r_{kk'} + \sum_{\substack{k'\in K\\ k>k'}} z^r_{k'k} \le \sum_{l\in L_r} n^l v^l_k \qquad \left(r\in R^P \setminus \left(R^A \cup R^B\right), k\in K\right) \tag{2.12}$$

$$\sum_{i\in I} x^r_{ik} + \sum_{j\in J} y^r_{jk} \le \sum_{l\in L_r} n^l v^l_k \qquad \left(r\in \left(R^A \cap R^B\right)\setminus R^P, k\in K\right) \tag{2.13}$$

$$\sum_{i\in I} x^r_{ik} + \sum_{\substack{k'\in K\\ k<k'}} z^r_{kk'} + \sum_{\substack{k'\in K\\ k>k'}} z^r_{k'k} \le \sum_{l\in L_r} n^l v^l_k \qquad \left(r\in \left(R^A \cap R^P\right)\setminus R^B, k\in K\right) \tag{2.14}$$

$$\sum_{j\in J} y^r_{jk} + \sum_{\substack{k'\in K\\ k<k'}} z^r_{kk'} + \sum_{\substack{k'\in K\\ k>k'}} z^r_{k'k} \le \sum_{l\in L_r} n^l v^l_k \qquad \left(r\in \left(R^B \cap R^P\right)\setminus R^A, k\in K\right) \tag{2.15}$$

$$\sum_{i\in I} x^r_{ik} + \sum_{j\in J} y^r_{jk} + \sum_{\substack{k'\in K\\ k<k'}} z^r_{kk'} + \sum_{\substack{k'\in K\\ k>k'}} z^r_{k'k} \le \sum_{l\in L_r} n^l v^l_k \qquad \left(r\in R^A \cap R^B \cap R^P, k\in K\right) \tag{2.16}$$

Contraintes de topologie du réseau entre les routeurs co-localisés dans le POP

Contraintes de topologie du réseau entre les routeurs co-localisés	(2.17)

Contraintes de capacité des liens d'accès

$$f^i_{ik} \le \sum_{r\in R^A} \alpha_r x^r_{ik} \quad \left(i\in I, k\in K\right) \tag{2.18}$$

$$\sum_{o\in\left(I\setminus\{i\}\right)\cup J} f^o_{ki} \le \sum_{r\in R^A} \alpha_r x^r_{ik} \qquad \left(i\in I, k\in K\right) \tag{2.19}$$

Contraintes de capacité des liens du réseau dorsal

$$f^j_{jk} \le \sum_{r\in R^B} \alpha_r y^r_{jk} \qquad \left(j\in J, k\in K\right) \tag{2.20}$$

$$\sum_{o\in I\cup\left(j\setminus\{j\}\right)} f^o_{kj} \le \sum_{r\in R^B} \alpha_r y^r_{jk} \qquad \left(j\in J, k\in K\right) \tag{2.21}$$

Contraintes de capacité des liens entre les routeurs co-localisés dans le POP

$$\sum_{o\in I\cup J} f^o_{kk'} \le \sum_{r\in R^P} \alpha_r z^r_{kk'} \quad \left(k<k', k, k'\in K\right) \tag{2.22}$$

$$\sum_{o\in I\cup J} f^o_{k'k} \le \sum_{r\in R^P} \alpha_r z^r_{kk'} \quad \left(k<k', k, k'\in K\right) \tag{2.23}$$

Contraintes de la conservation du trafic

$$\sum_{k\in K} f_{ik}^{i} = \sum_{d\in(I\setminus\{i\})\cup J} p^{id} \quad (i\in I) \tag{2.24}$$

$$\sum_{k\in K} f_{ki}^{o} = p^{oi} \quad \left(o\in\left(I\setminus\{i\}\right)\cup J, i\in I\right) \tag{2.25}$$

$$\sum_{k\in K} f_{jk}^{j} = \sum_{d\in I\cup(J\setminus\{j\})} p^{id} \quad (j\in J) \tag{2.26}$$

$$\sum_{k\in K} f_{kj}^{o} = p^{oj} \quad \left(o\in I\cup\left(J\setminus\{j\}\right), j\in J\right) \tag{2.27}$$

$$\sum_{i\in I}\left(f_{ik}^{o}-f_{ki}^{o}\right)+\sum_{j\in J}\left(f_{jk}^{o}-f_{kj}^{o}\right)+\sum_{k'\in K\setminus\{k\}}\left(f_{k'k}^{o}-f_{kk'}^{o}\right)=0 \quad (o\in I\cup J, k\in K) \tag{2.28}$$

Contraintes d'intégralité et de non-négativité

$$f\in\mathbb{R}_{+}^{|I||J||K|(|I|+|J|+|K|-1)}, u\in\{0,1\}^{|K||T|}, v\in\mathbb{N}^{|K||L|}, x\in\mathbb{N}^{|I||K||R^{A}|}, y\in\mathbb{N}^{|J||K||R^{B}|}, z\in\mathbb{N}^{\frac{|K|}{2}(|K|-1)}. \tag{2.29}$$

Comme mentionné précédemment, la fonction objective (2.4) de P est composée de trois termes. Ces termes représentent respectivement le coût des liens, le coût des routeurs et le coût des cartes d'interface. Cette fonction est celle à minimiser et représente le coût total du POP. Les contraintes (2.5) et (2.6) sont respectivement les contraintes d'affectation des clients et des O-POPs. Ces contraintes requièrent que chaque client soit connecté au POP avec les types de liens d'accès requis et que chaque O-POP soit connecté au POP avec les types de liens dorsaux (*backbone links*) requis. Les contraintes d'unicité des types de routeurs (2.7) imposent qu'au plus un seul type de routeur soit installé au site $k \in K$. Les contraintes (2.8) nécessitent que le nombre total de fentes utilisées par les cartes d'interfaces au site $k \in K$ soit plus petit ou égal au nombre de fentes disponibles dans le type de routeur installé à ce site. Les contraintes (2.9) imposent que la somme des vitesses des ports installés dans un routeur au site $k \in K$ doit être plus petite ou égale à sa capacité. Les contraintes (2.10) à (2.16) spécifient que le nombre de liens de type $r \in R$ connectés à un routeur doit être plus petit ou égal au nombre de ports disponibles de ce type. Les contraintes (2.17) n'ont pas été spécifiées car, comme nous l'avons expliqué au premier chapitre, la topologie entre les routeurs co-

localisés dans le POP peut varier. En effet, celle-ci dépend des objectifs du planificateur de réseau. Plusieurs topologies différentes peuvent être considérées. Pour notre modèle, nous avons choisi une topologie maillée (*full-mesh topology*). C'est-à-dire que chaque routeur est directement relié à tous les autres. Cette topologie n'est pas très évolutive car elle peut-être très coûteuse si plusieurs routeurs sont installés. En effet, pour n routeurs, nous avons besoin de $n(n-1)/2$ liens. Cependant, elle garantit une grande robustesse en cas de panne car il existe plusieurs liens redondants. De plus, elle est la plus performante car on passe par au plus deux routeurs pour entrer et sortir du POP. Cette topologie peut être décrite par les contraintes suivantes.

$$\sum_{t\in T} u_k^t + \sum_{t\in T} u_{k'}^t \leq \sum_{r\in R^P} z_{kk'}^r + 1 \qquad (k<k', k, k' \in K) \tag{2.30}$$

Ces contraintes (2.30) exigent la présence d'au moins un lien entre deux sites si et seulement si un routeur a été installé à ces deux sites.

Les contraintes de capacité des liens d'accès (2.18) et (2.19) assurent que le trafic du client $i \in I$ vers le POP sera plus petit ou égal au total de la capacité des liens d'accès de ce client (dans les deux directions). Similairement, les contraintes de capacité des liens du réseau dorsal (2.20) et (2.21) garantissent que le trafic du O-POP $j \in J$ vers le POP n'excèdera pas le total de la capacité des liens dorsaux entre l'O-POP $j \in J$ et le POP. Les contraintes de capacité des liens entre les routeurs (2.22) et (2.23) imposent que le flot du site $k \in K$ au site $k' \in K$ soit plus petit ou égal à la capacité total des liens installés entre k et k'. Les contraintes (2.24) à (2.28) assurent la conservation du trafic dans chaque routeur. Finalement, (2.29) sont les contraintes d'intégralité et de non-négativité.

Considérons les propositions suivantes.

Proposition 1. Les inégalités suivantes sont valides pour P.

$$x_{ik}^r \le q_i^r \sum_{t\in T} u_k^t \quad (r \in R^A, i \in I, k \in K) \tag{2.31}$$

$$y_{jk}^r \le q_j^r \sum_{t\in T} u_k^t \quad (r \in R^B, j \in J, k \in K) \tag{2.32}$$

Preuve

Dans l'expression (2.31), $\sum_{t\in T} u_k^t$ peut uniquement prendre les valeurs 0 ou 1 puisqu'il peut seulement y avoir un seul routeur par site $k \in K$.
Donc, si

$$\sum_{t\in T} u_k^t = 0, \text{ alors } x_{ik}^r = 0$$

et si

$$\sum_{t\in T} u_k^t = 1, \text{ alors } x_{ik}^r \le q_r^i$$

Il est important de noter que x_{ik}^r peut être plus petit ou égal à q_r^i. Il est égal quand tous les liens d'un client sont connectés au même site. Par contre, il peut être inférieur si les liens d'un client sont connectés à deux sites différents. L'équation (2.32) se démontre de la même façon que l'équation (2.31).

Proposition 2. Les inégalités suivantes sont valides pour P.

$$\sum_{k\in K}\sum_{t\in T} u_k^t \ge \left\lceil \frac{1}{2}\left(\left(A \max_{t\in T}\{m^t\} + 1 \right) - \left(\left(A \max_{t\in T}\{m^t\} + 1 \right)^2 - 4AB \right)^{1/2} \right) \right\rceil \tag{2.33}$$

$$\sum_{k\in K}\sum_{t\in T} u_k^t \le \min\left\{ \left\lfloor \frac{1}{2}\left(\left(A \max_{t\in T}\{m^t\} + 1 \right) + \left(\left(A \max_{t\in T}\{m^t\} + 1 \right)^2 - 4AB \right)^{1/2} \right) \right\rfloor, |K| \right\} \tag{2.34}$$

où

$$A = \max_{l \in L_r : r \in R^P} \left\{ \frac{n^l}{h^l} \right\} \tag{2.35}$$

et

$$B = \sum_{r \in R^A} \frac{\sum_{i \in I} q_i^r}{\max_{l \in L_r} \left\{ \frac{n^l}{h^l} \right\}} + \sum_{r \in R^B} \frac{\sum_{j \in J} q_j^r}{\max_{l \in L_r} \left\{ \frac{n^l}{h^l} \right\}} \tag{2.36}$$

Dans les inéquations ci-dessus, A représente le nombre maximum de ports de type $r \in R^P$ par fente et B le nombre minimum de fentes nécessaires pour connecter les liens d'accès et les liens dorsaux.

Preuve

Nous pouvons multiplier les éléments de la somme du côté droit de l'équation (2.10) par h^l/h^l sans changer la valeur de l'équation. En sortant le terme n^l/h^l de la sommation, nous obtenons l'inéquation suivante.

$$\sum_{i \in I} x_{ik}^r \leq \max_{l \in L_r} \left\{ \frac{n^l}{h^l} \right\} \sum_{l \in L_r} h^l v_k^l$$

De façon similaire, en utilisant les inéquations (2.10) à (2.16) nous obtenons

$$\sum_{r \in R^A} \frac{\sum_{i \in I} x_{ik}^r}{\max_{l \in L_r} \left\{ \frac{n^l}{h^l} \right\}} + \sum_{r \in R^B} \frac{\sum_{j \in J} y_{jk}^r}{\max_{l \in L_r} \left\{ \frac{n^l}{h^l} \right\}} + \sum_{r \in R^P} \frac{\sum_{k' \in K, k < k'} z_{kk'}^r + \sum_{k' \in K, k > k'} z_{k'k}^r}{\max_{l \in L_r} \left\{ \frac{n^l}{h^l} \right\}} \leq \sum_{l \in L} h^l v_k^l \,. \tag{2.37}$$

Avec la topologie que nous avons utilisée, nous pouvons exprimer l'inéquation (2.30) de la façon suivante :

$$\left(\sum_{t\in T} u_k^t\right)\left(\sum_{t\in T} u_{k'}^t\right) \le \sum_{r\in R^P} z_{kk'}^r \quad (k<k' \text{ et } k,k'\in K)$$

En utilisant l'inéquation (2.8) nous obtenons :

$$\sum_{r\in R^A} \frac{\sum_{i\in I} x_{ik}^r}{\max_{l\in L_r}\left\{\frac{n^l}{h^l}\right\}} + \sum_{r\in R^B} \frac{\sum_{j\in J} y_{jk}^r}{\max_{l\in L_r}\left\{\frac{n^l}{h^l}\right\}} \le \sum_{t\in T} m^t u_k^t - \frac{\sum_{k'\in K\setminus\{k\}}\left(\sum_{t\in T} u_k^t\right)\left(\sum_{t\in T} u_{k'}^t\right)}{\max_{l\in L_r, r\in R^P}\left\{\frac{n^l}{h^l}\right\}} \tag{2.38}$$

Si nous sommons sur $k\in K$ les deux cotés de l'équation (2.38) nous obtenons l'inégalité suivante où A et B sont respectivement définis par les équations (2.35) et (2.36).

$$B \le \sum_{k\in K}\sum_{t\in T} m^t u_k^t - \frac{1}{A}\sum_{k\in K}\sum_{t\in T} u_k^t\left(\sum_{k\in K}\sum_{t\in T} u_k^t - 1\right) \tag{2.39}$$

À partir de l'inéquation (2.39) nous obtenons

$$B \le \max_{t\in T}\left\{m^t\right\}\sum_{k\in K}\sum_{t\in T} u_k^t - \frac{1}{A}\sum_{k\in K}\sum_{t\in T} u_k^t\left(\sum_{k\in K}\sum_{t\in T} u_k^t - 1\right). \tag{2.40}$$

Ensuite, en multipliant par A de chaque côté, nous avons

$$AB \le A\max_{t\in T}\left\{m^t\right\}\sum_{k\in K}\sum_{t\in T} u_k^t - \sum_{k\in K}\sum_{t\in T} u_k^t\left(\sum_{k\in K}\sum_{t\in T} u_k^t - 1\right) \tag{2.41}$$

où

$$-\left(\sum_{k\in K}\sum_{t\in T} u_k^t\right)^2 + \left(A\max_{t\in T}\left\{m^t\right\}+1\right)\sum_{k\in K}\sum_{t\in T} u_k^t - AB \ge 0. \tag{2.42}$$

En remplaçant $\sum_{k\in K}\sum_{t\in T} u_k^t$ par x (ou $x \in \mathbb{R}$) dans l'inéquation (2.42), nous obtenons l'équation suivante

$$f(x) = -x^2 + \left(A \max_{t\in T}\left\{m^t\right\} + 1\right)x - AB\,. \tag{2.43}$$

Les solutions de (2.43) sont

$$x_- = \frac{1}{2}\left(\left(A \max_{t\in T}\left\{m^t\right\} + 1\right) - \left(\left(A \max_{t\in T}\left\{m^t\right\} + 1\right)^2 - 4AB\right)^{1/2}\right)$$

et

$$x_+ = \frac{1}{2}\left(\left(A \max_{t\in T}\left\{m^t\right\} + 1\right) + \left(\left(A \max_{t\in T}\left\{m^t\right\} + 1\right)^2 - 4AB\right)^{1/2}\right)$$

et la valeur maximum de *f(x)* est obtenue en égalant à zéro la dérivée de l'équation (2.43) :

$$x_{\max} = \frac{A \max_{t\in T}\left\{m^t\right\} + 1}{2}$$

À la Figure 2.5, l'équation $f(x)$ a été représentée avec un intervalle de x tel que $f(x) \geq 0$. La proposition est exacte car $\sum_{k\in K}\sum_{t\in T} u_k^t$ est un entier pour toutes les solutions réalisables de P. De plus, nous pouvons remarquer que (2.33) et (2.34) indiquent respectivement le nombre minimum et le nombre maximum de routeurs à installer dans le POP afin de desservir la demande de tous les clients et de tous les O-POPs.

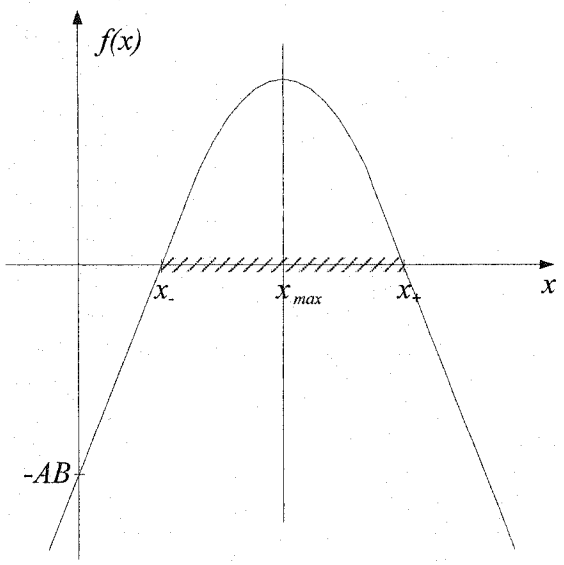


Figure 2.5 Équation *f(x)*

Proposition 3. L'inégalité suivante est valide pour P.

$$AB \leq \left\lceil \frac{A \max_{t \in T} \left\{ m^t \right\} + 1}{2} \right\rceil \left\lfloor \frac{A \max_{t \in T} \left\{ m^t \right\} + 1}{2} \right\rfloor \tag{2.44}$$

Preuve

Dans la proposition 2, nous avons montré que le coté droit de l'inégalité (2.41) est obtenu quand $\sum_{k \in K} \sum_{t \in T} u_k^t$ est égal à $\left\lceil \left(A \max_{t \in T} \left\{ m^t \right\} + 1 \right) / 2 \right\rceil$ et $\left\lfloor \left(A \max_{t \in T} \left\{ m^t \right\} + 1 \right) / 2 \right\rfloor$. Donc, si nous remplaçons $\sum_{k \in K} \sum_{t \in T} u_k^t$ dans l'équation (2.41), nous obtenons (2.44).

Proposition 4. P est réalisable si les inéquations (2.44) et (2.45) sont respectées.

$$|K| \geq \left\lceil \frac{1}{2} \left(\left(A \max_{t \in T} \left\{ m^t \right\} + 1 \right) - \left(\left(A \max_{t \in T} \left\{ m^t \right\} + 1 \right)^2 - 4AB \right)^{1/2} \right) \right\rceil \tag{2.45}$$

Preuve

Les inéquations (2.44) et (2.45) ont été obtenues à partir des contraintes du problème *P*. Si *P* est réalisable, alors ces inéquations sont respectées pour toutes les solutions possibles.

2.5.4 Complexité du modèle P

Dans cette section, nous voulons démontrer la complexité du problème P. Tout d'abord, voici un bref résumé sur la NP-complétude.

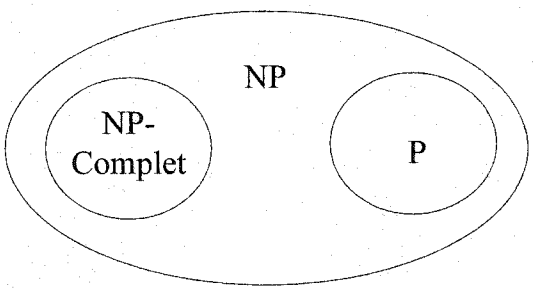


Figure 2.6 Les différentes classes de problèmes

Comme illustré dans la Figure 2.6, il existe différentes classes de problèmes. La classe NP est la classe la plus générale et englobe les classes P et NP-complet. En effet, la classe NP est celle pour laquelle les problèmes peuvent être résolus par un algorithme non-déterministe en un temps polynomial. Un algorithme non-déterministe est constitué de deux étapes. La première étape consiste à choisir une solution aléatoirement parmi l'ensemble de tous les cas possibles. La deuxième étape consiste à vérifier si la solution choisie à l'étape précédente résout le problème de départ. Si cette vérification s'effectue en temps polynomial, alors nous avons un problème qui se situe dans la classe NP. Les problèmes de la classe P peuvent être résolus par un algorithme déterministe en temps polynomial. P est donc un sous-ensemble de NP puisque tout algorithme déterminisme est un cas particulier d'un algorithme non-déterminisme. Les problèmes de la classe NP-complet sont un groupe de problèmes équivalents pour lesquels il est peu probable de

trouver une solution exacte à l'aide d'un algorithme en temps polynomial. Pour traiter ce type de problèmes, l'utilisation de méthodes heuristiques est employée. Une méthode heuristique est une méthode approximative avec un temps de calcul raisonnable.

Il existe de nombreux problèmes concrets et pratiques qui sont NP-complets. Des problèmes aussi divers que le commis voyageur, la coloration optimale de graphes, le sac à dos et le cycle hamiltonien (Brassard et Bratley, 1987).

Toute cette théorie sur la NP-complétude doit son origine à Cook et Karp qui ont introduit les notions de NP-complétude et de réduction. La réduction est un des outils utilisés pour démontrer qu'un problème est NP-complet. En démontrant que notre problème se rapporte à un problème qui a déjà été démontré NP-complet, nous pourrons alors affirmer que notre problème sera aussi NP-complet. Si nous étions en mesure de trouver un algorithme polynomial pour un problème NP-complet, alors ça serait le cas aussi de tous les problèmes NP-complets.

Proposition 5. P est un problème de type NP-complet.

Preuve

En effet, notre problème peut se réduire au problème du sac à dos entier qui a été démontré comme étant NP-complet (Garey et Johnson, 1979). Celui-ci se résume de la façon suivante : nous avons un sac de capacité K et nous voulons maximiser la somme de la valeur des objets insérés dans ce sac tout en respectant la contrainte de capacité de ce dernier.

Afin de faire cette réduction, nous devons poser quelques conditions. Premièrement, nous faisons l'hypothèse qu'aucun autre POP n'est connecté au POP que nous voulons optimiser ($|J| = 0$). Ceci a pour effet de faire disparaître la variable y^r_{jk} de notre modèle. Deuxièmement, nous supposons qu'il n'y a qu'un seul site disponible ($|K| = 1$). La variable $z^r_{kk'}$ disparaît du modèle puisqu'il n'y aura pas de liens inter-routeurs. Comme troisième hypothèse, nous présumons qu'il n'y a pas de trafic circulant dans notre POP ($p^{od} = 0$, pour tout o, $d \in I \cup J$). Les variables $f^o_{ik}(f^o_{ki})$ et $f^o_{jk}(f^o_{kj})$ disparaissent donc aussi du modèle. Ensuite, nous assumons qu'il y a seulement un type

de lien d'accès possible pour connecter les clients ($|I| \neq \varnothing$) au POP ($|R^A| = 1$, R^B et $R^P = \varnothing$). De plus, nous supposons que chaque client a exactement un seul lien d'accès au POP ($q_r^i = 1$ pour tout $i \in I$). Finalement, un seul type de routeur est disponible et celui-ci est installé au site 1 ($|T| = 1$ et $u_k^t = u_1^1 = 1$).

Avec ces conditions, notre modèle devient le suivant.

$$\min \sum_{l \in L} a_1^l v_1^l \tag{2.46}$$

sujet à

$$x_{i1}^1 = 1 \tag{2.47}$$

$$\sum_{l \in L} h^l v_1^l \leq m^1 \tag{2.48}$$

$$\alpha_1 \sum_{l \in L} n^l v_1^l \leq \beta^1 \tag{2.49}$$

$$|I| \leq \sum_{l \in L} n^l v_1^l \tag{2.50}$$

$$v \in \mathbb{N}. \tag{2.51}$$

Maintenant, supposons que les valeurs de m^l et de β^1 des inéquations (2.48) et (2.49) sont très grandes, nous nous retrouvons avec le problème suivant.

$$\min \sum_{l \in L} a_1^l v_1^l \tag{2.46}$$

sujet à

$$\sum_{l \in L} n^l v_1^l \geq |I| \tag{2.50}$$

$$v \in \mathbb{N} \tag{2.51}$$

ce qui est exactement le problème de sac à dos entier. Nous pouvons donc conclure que notre problème est NP-complet. Pour plus de détails à propos du problème de sac à dos, voir Nemhauser et Wolsey (1988).

2.6 Méthode heuristique

Comme nous l'avons mentionné, les méthodes heuristiques sont typiquement utilisées lorsque nous voulons trouver de bonnes solutions à un problème NP-complet en temps raisonnable. Par contre, le coût de la solution trouvée est supérieur ou égal à celui de la solution optimale. Pour vérifier si ce coût est acceptable, nous devons le comparer avec une valeur de référence. Cette valeur est soit la valeur optimale ou la valeur d'une borne inférieure. Cette comparaison fera l'objet du prochain chapitre portant sur l'implantation et les résultats. Dans cette section, nous présenterons et expliquerons notre heuristique.

Avant de présenter l'algorithme, il est important de définir les concepts de routeur d'accès et de routeur dorsal. Un routeur d'accès est un routeur du POP qui est utilisé exclusivement pour connecter les liens d'accès. Un routeur dorsal est un routeur du POP qui est utilisé pour connecter les liens dorsaux. Cependant, s'il reste de la capacité, des liens d'accès peuvent y être connectés.

Étant donné la complexité de l'heuristique, nous l'avons schématisée de deux moyens différents. Le premier est un algorithme d'optimisation des points de présence et le second est un organigramme. L'algorithme est présenté à la Figure 2.7 et l'organigramme à la Figure 2.8. La description de toutes les étapes de l'algorithme est présentée dans les paragraphes suivants.

L'heuristique va effectuer un certain nombre d'itérations dépendamment des valeurs de n_{min} et n_{max} (où n représente le nombre de routeurs). En fait, $n_{max} - n_{min}$ itérations seront effectuées. N_{min} et n_{max} sont calculés respectivement à l'aide de la partie droite des inéquations (2.33) et (2.34). Si n est plus grand que n_{max}, alors l'algorithme prend fin en affichant la meilleure solution obtenue. Dans le cas contraire, nous installons n routeurs aux n premiers sites (site 0 jusqu'au site n-1).

Étape 1 **(Initialisation)**

Calculer n_{min} et n_{max} et assigner $n := n_{min}$.

Étape 2 **(Générer une solution avec *n* routeurs)**

2.1 Installer n routeurs.

2.2 Calculer le nombre de cartes d'interface nécessaires.

2.3 Calculer la capacité à réserver dans chaque routeur.

2.4 Calculer b, le nombre minimum de routeurs dorsaux.

2.5 **(Calculer une solution avec *b* routeurs dorsaux)**

2.5.1 Installer les cartes des liens dorsaux dans les b routeurs dorsaux. S'il n'y a pas assez de fentes disponibles, aller à l'étape 2.5.10.

2.5.2 Assigner les liens aux ports.

2.5.3 Connecter les b routeurs entre eux et calculer le flot sur chaque lien reliant deux routeurs. Pour chacun de ces liens, si le flot est supérieur à la capacité, ajouter des liens supplémentaires.

2.5.4 Calculer la capacité résiduelle. Si des contraintes de capacité d'un ou plusieurs routeurs sont violées aller à l'étape 2.5.10.

2.5.5 Installer les cartes des liens d'accès dans les $(n - b)$ routeurs d'accès. Utiliser les routeurs dorsaux si nécessaire. S'il n'y a pas assez de fentes disponibles, aller à l'étape 2.5.10.

2.5.6 Assigner les liens d'accès aux ports.

2.5.7 Connecter les n routeurs entre eux et recalculer le flot sur chaque lien. Si le flot entre deux routeurs est plus grand que la capacité, ajouter des liens supplémentaires. Pour tout autres liens, si le flot est plus grand que la capacité, aller à l'étape 2.5.10.

2.5.8 Calculer la capacité résiduelle. Si des contraintes de capacité d'un ou plusieurs routeurs sont violés aller à l'étape 2.5.10.

2.5.9 **(Mise à jour de la meilleure solution)**

Calculer le coût de la solution présente. Si le coût de cette solution est inférieur à la meilleure solution trouvée jusqu'à présent, garder cette solution en mémoire.

2.5.10 Si $b < n$ alors $b := b + 1$ et aller à l'étape 2.5. Sinon, aller à l'étape 3.

Étape 3 **(Test de terminaison)**

Si n < n_{max} alors $n := n + 1$ et aller à l'étape 2. Sinon, afficher la solution en mémoire.

Figure 2.7 Algorithme d'optimisation des points de présence

L'étape suivante consiste à calculer le nombre de cartes d'interfaces nécessaire pour brancher tous les clients et tous les autres POPs aux routeurs. Pour ce faire, nous calculons la somme du nombre de liens de type $r \in R^A$ de tous les clients et la somme du nombre de liens de type $r \in R^B$ de tous les autres POPs. Ensuite, nous divisons ce résultat par le nombre de ports que contient la carte d'interface pour ce type de technologie. Par exemple, prenons les données du Tableau 2.1. Si nous calculons la somme du nombre de liens de type OC-3 pour tous les clients, nous obtenons quatre. Étant donné que la carte d'interface utilisée pour la technologie OC-3 comporte quatre ports, nous aurons seulement besoin d'une carte de type *A*. Par contre, la somme du nombre de liens de type OC-12 est cinq et le nombre de ports de la carte d'interface utilisée pour cette technologie est de quatre. Nous aurons donc besoin d'une carte de type *A* et de deux cartes de type *B* afin de combler les besoins de tous les clients.

Ensuite, nous devons calculer le nombre de fentes ainsi que la capacité à réserver dans chaque routeur pour les liens inter-routeurs de type $r \in R^P$. En effet, pour n routeurs, la topologie utilisée génère $n(n-1)/2$ liens inter-routeurs. Le nombre de fentes et la capacité à réserver sont proportionnels au nombre de routeurs installés. Par exemple, si quatre routeurs sont installés dans un POP, nous aurons six liens inter-routeurs et nous devrons réserver trois fentes et 30 Gbps dans chacun d'eux (supposant qu'on a besoin d'une fente par lien OC-192).

Tableau 2.1 Connexions des clients et cartes d'interface

Client	Nombre de liens	
	OC-3	OC-12
1	2	0
2	0	1
3	1	2
4	0	1
5	1	1

Cartes d'interface		
Type	Technologie	Nombre de ports
A	OC-3	4
B	OC-12	4

Par la suite, nous essayons de calculer une solution réalisable pour un certain nombre b de routeurs dorsaux. Le nombre de routeurs dorsaux varie entre le nombre minimum de routeurs dorsaux et n. Toutes les valeurs possibles de b seront essayées à tour de rôle. Le nombre minimum de routeurs dorsaux est calculé à partir du nombre de cartes nécessaires pour connecter tous les autres POPs.

Après avoir déterminé le nombre de routeurs dorsaux, nous installons les cartes d'interface nécessaires pour pouvoir connecter tous les autres POPs dans ces routeurs. Si un certain nombre de cartes d'interface n'ont pas pu être placées dans un des routeurs dorsaux, alors cette solution n'est pas réalisable. Nous devons alors incrémenter de un le nombre de routeurs dorsaux et recommencer le processus. Par contre, si toutes les cartes ont été placées dans les b routeurs, alors nous passons à la prochaine étape qui consiste à assigner les liens dorsaux aux ports. Dans cette étape, chaque lien du réseau dorsal est assigné à un port particulier d'une carte d'interface.

L'étape suivante consiste à connecter les b routeurs dorsaux entre eux. Ensuite, nous devons calculer le flot sur ces liens afin de s'assurer que la capacité du lien est plus grande ou égale à la somme des trafics circulant sur ce lien. Pour cette partie, le trafic provient seulement des autres POPs car les clients ne sont pas encore connectés. Si le flot obtenu entre deux routeurs dorsaux est plus grand que la capacité du lien, un lien supplémentaire sera alors ajouté entre ces deux sites. Si des liens ont été ajoutés, nous devons alors soustraire la capacité et le nombre de fentes utilisées par l'ajout de ces liens. Si un routeur à trop de cartes installées ou que sa capacité résiduelle est plus petite que zéro, alors cette configuration n'est pas une solution réalisable.

Il est maintenant temps d'installer les cartes des liens d'accès dans les routeurs d'accès. Il existe des routeurs d'accès si et seulement si b est plus petit que n. Sinon, cela signifie que tous les routeurs sont des routeurs dorsaux. Nous devrons alors essayer d'installer les cartes pour les liens d'accès dans les fentes libres de ces routeurs.

Comme nous l'avons fait avec la partie dorsale, nous devons maintenant assigner les liens d'accès aux ports. Ensuite, nous connectons les n routeurs entre eux. Chaque routeur aura $(n - 1)$ connexions destinées aux autres routeurs. Une fois que tous les

routeurs sont connectés, nous recalculons le flot sur chaque lien. Cette fois ci, le trafic provient des autres POPs et des clients. Ensuite, nous vérifions que le flot de chaque lien n'excède pas la capacité. Si la capacité d'un lien connectant deux routeurs est insuffisante, alors nous ajoutons un autre lien entre ces deux sites. Par contre, si la capacité d'un lien connectant un client ou un autre POP à un routeur est insuffisante, la solution proposée n'est donc pas réalisable.

La dernière étape consiste à calculer le nombre de fentes et la capacité résiduelle. En effet, s'il y a plus de cartes insérées dans un routeur que celui-ci peut en contenir, ou que la capacité résiduelle est inférieure à zéro, alors nous sommes en présence d'une solution impossible à réaliser. Sinon, nous sommes en présence d'une solution réalisable. Il ne reste plus qu'à calculer le coût de cette solution. Si la solution a un coût inférieur à toutes les autres solutions trouvées, alors nous gardons cette solution en mémoire comme étant la meilleure solution jusqu'à présent. Toute autre nouvelle solution sera comparée avec la meilleure solution en mémoire.

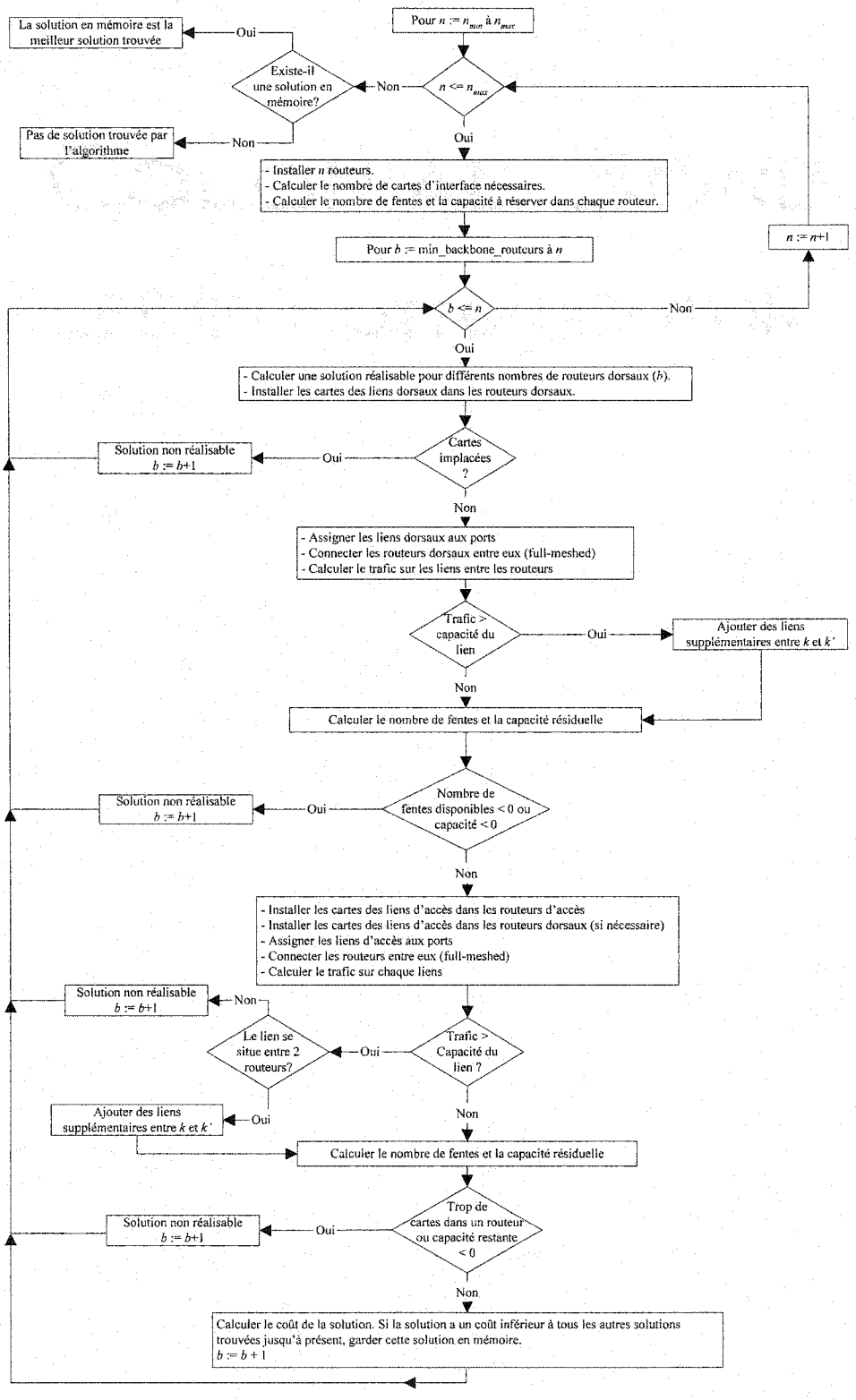


Figure 2.8 Organigramme de la méthode heuristique

CHAPITRE 3

IMPLANTATION ET RÉSULTATS

Après avoir proposé et analysé un modèle et une méthode heuristique, nous allons maintenant expliquer comment nous les avons implantés. Cette étape nous permettra d'appliquer concrètement nos modèles à l'aide de technologies déjà existantes. En effet, nous commencerons par présenter l'implantation du modèle de programmation mathématique et nous suivrons avec celle de la méthode heuristique. Après avoir détaillé l'environnement d'expérimentation, nous démontrerons le fonctionnement de la méthode heuristique à l'aide d'un exemple détaillé. Ensuite, nous exposerons le plan d'expérience et nous terminerons avec une présentation et une analyse de nos résultats.

3.1 Détails de l'implantation

Dans cette section, nous détaillerons comment nous avons implanté nos différents modèles. Nous commencerons tout d'abord avec le modèle de programmation mathématique et nous enchaînerons avec la méthode heuristique.

3.1.1 Implantation du modèle de programmation mathématique

Comme nous l'avons démontré au chapitre précédent, ce modèle est composé d'une fonction objective et de plusieurs contraintes. Nous pourrions le classifier comme étant un modèle de programmation mixte car certaines variables sont réelles ($\in \mathbb{R}$) tandis que d'autres sont entières ($\in \mathbb{N}$).

Plusieurs logiciels d'optimisation existent sur le marché. Cependant, certains sont plus faciles à utiliser que d'autres. Pour notre part, nous avons utilisé la suite ILOG OPL STUDIO 3.5 (2000). OPL STUDIO est un environnement de développement pour la programmation mathématique ainsi que pour des applications d'optimisation combinatoire. Il réduit considérablement le temps de développement car il utilise du

code plus simple que les langages de programmation traditionnels (C ou C++). Par exemple, l'utilisation des mots clés comme *forall* et *sum* facilitent de beaucoup l'écriture d'équations. Comme nous pouvons le constater à la Figure 3.1, les expressions mathématiques sont plus faciles à programmer.

L'équation mathématique suivante :

$$\sum_{k \in K} x_{ik}^{r} = q_{r}^{i} \quad (r \in R^{A}, i \in I)$$

peut se traduire avec OPL par :

```
forall(r in Ra, I in Clients)
  sum(k in Location)
    x [r, I, k] = client[I, r];
```

Figure 3.1 Expression en langage OPL d'une équation mathématique

Afin de résoudre notre modèle, nous avons utilisé le solutionneur CPLEX Mixed Integer Optimizer qui utilise un algorithme de séparation et évaluation progressive (*Branch and Bound*) pour résoudre les modèles à variables mixtes. Ces problèmes consistent à choisir la meilleure combinaison parmi toutes les combinaisons possibles. La difficulté principale dans ce type de problème est qu'il n'y a pas de conditions à évaluer afin de déterminer si une solution réalisable est optimale ou non. Le seul moyen de garantir l'optimalité est de comparer toutes les solutions entre elles. Étant donné la NP-complétude de cette tâche, nous devons trouver un moyen afin de détecter que la solution est optimale sans avoir à évaluer toutes les possibilités. Pour ce faire, nous pouvons effectuer une énumération partielle de tous les cas possibles.

Chaque problème peut être représenté par un arbre d'énumération total. Par exemple, la Figure 3.2 représente un arbre d'énumération total ayant 2^3 solutions possibles. L'arbre comporte un nœud de départ (nœud 0) appelé racine. Tous les nœuds consécutifs qui sont créés sont des solutions partielles (nœuds 1 à 6). Ces solutions ont seulement un sous-ensemble des variables qui sont assignées. Seuls les derniers nœuds

(nœuds 7 à 14), appelés feuilles, représentent une solution finale. Ces nœuds représentent l'ensemble de toutes les solutions possibles.

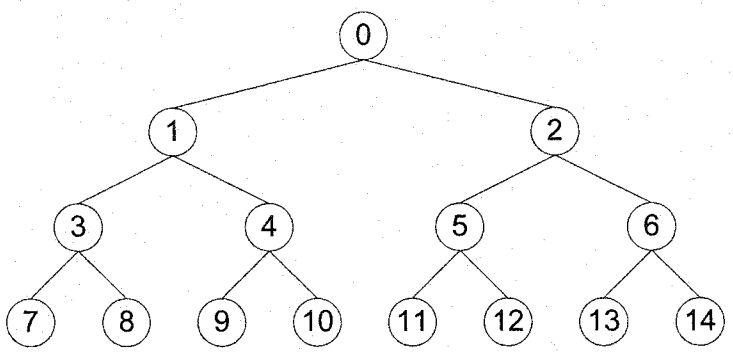


Figure 3.2 Arbre d'énumération totale

De façon générale, le principe d'évaluation et séparation progressive est le suivant : à chaque nœud, si nous pouvons démontrer que la solution optimale ne peut pas être trouvée dans les descendants de celui-ci, alors nous pouvons ignorer tous ces descendants. Cela aura pour effet de réduire la taille de l'arbre à parcourir. Si nous réussissons à réduire suffisamment l'arbre, le temps d'exécution en sera réduit de beaucoup. Cependant, comment peut-on s'assurer que la solution optimale ne se trouve pas dans les descendants d'un nœud? Si un nœud a un coût supérieur à la meilleure solution trouvée jusqu'à présent, alors nous pouvons éliminer avec certitude tous les descendants de ce nœud. En effet, toutes les solutions trouvées à partir de ce nœud auront un coût égal ou supérieur à celui-ci. Il est possible d'évaluer le coût d'un nœud en calculant une borne inférieure. Cette borne est calculée en relâchant un certain nombre de contraintes. Par exemple, dans un problème de programmation entière, la borne inférieure peut être obtenue en relâchant la contrainte d'intégralité sur un certain nombre de variables. Il est évident que plus la borne inférieure est élevée, plus il sera possible d'éliminer des nœuds et ainsi réduire l'arbre. Dans le pire des cas, tous les nœuds de l'arbre seront évalués (Nemhauser et Wolsey, 1988).

3.1.2 Implantation de la méthode heuristique

La programmation de notre heuristique a été réalisée à l'aide du langage de programmation C++. Le logiciel qui a été utilisé est Microsoft Visual C++ 6.0.

Le code programmé se répartit sur trois fichiers principaux : *heuristic.cpp*, *dijkstra.cpp* et *list.cpp*. Le fichier *heuristic.cpp* est le fichier principal du programme. Il commence par vérifier si le fichier de données est présent dans le répertoire courant. Dans l'affirmative, il lit ce dernier et garde en mémoire toutes les informations nécessaires. Après avoir mémorisé ces informations, il exécute toutes les routines nécessaires pour appliquer les différentes étapes de l'heuristique que nous avons décrites au chapitre trois.

Le fichier *dijkstra.cpp* simule la politique de routage qui est basée sur les plus courts chemins (PCC). La majorité des protocoles de routage utilisent les PCCs pour trouver la route de chaque origine à chaque destination dans le réseau. Par exemple, le protocole OSPF (Open Shortest Path First) emploie les PCCs pour acheminer les paquets. Généralement, chaque PCC est calculé sur la base de métriques de liens. Ces métriques sont généralement configurées par le planificateur du réseau ou calculées par le routeur. Pour réaliser cette tâche, nous avons implanté un tableau de deux dimensions contenant les métriques des nœuds adjacents. Nous commençons par initialiser ce tableau avec une très grande valeur indiquant que le coût pour relier deux nœuds est infiniment grand. Ensuite, si un client ou un autre POP est relié à un routeur, alors une métrique de 100 est insérée. Une métrique de 1 est introduite entre les routeurs. Pour terminer, l'algorithme de Dijkstra est exécuté afin de trouver les PCCs. Cette notation est représentée à la Figure 3.3. Par exemple, la valeur du PCC entre le client C1 et le client C4 est de 201, tandis que celle entre le client C2 et l'autre POP O1 est de 200.

C'est le fichier *list.cpp* que nous avons implanté la structure de données. En effet, les renseignements sur les différents routeurs, les cartes d'interfaces et les liens sont tous mémorisés dans des listes chaînées. Ces listes sont utilisées pour garder l'information du fichier de données en mémoire. Plusieurs fonctions ont été implantées afin d'ajouter de nouveaux éléments à une liste, de naviguer dans celle-ci

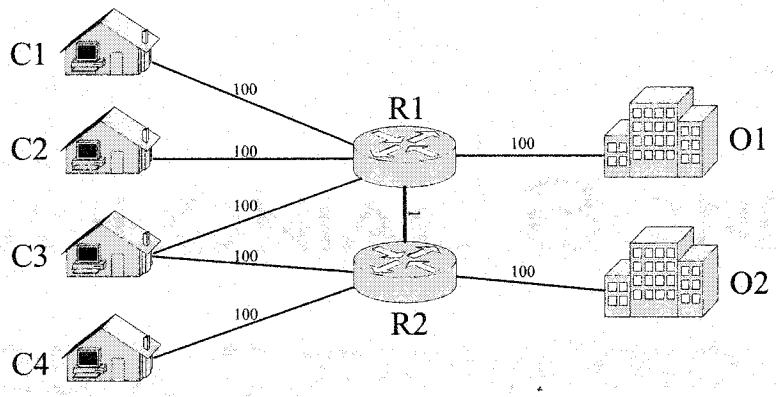


Figure 3.3 Métrique des liens

et de trouver l'information voulue. Les fonctions qui nous permettent d'ajouter de nouveaux éléments à la liste sont les suivantes :

- insert_router() : permet d'insérer les caractéristiques d'un nouveau routeur dans la liste des routeurs;
- insert_card() : permet d'ajouter les caractéristiques d'une nouvelle carte d'interface dans la liste des cartes;
- insert_link() : permet d'insérer les caractéristiques d'un nouveau type de lien dans la liste des liens.

Les fonctions qui nous permettent de naviguer à travers une liste sont les suivantes :

- set_first() : permet de retourner au début de la liste;
- next() : permet d'aller à l'élément suivant de la liste;
- prev() : permet de revenir à l'élément précédent de la liste;
- is_empty() : permet de savoir si la liste est vide;
- length() : permet de savoir combien d'éléments sont dans la liste.

Finalement, les fonctions qui nous permettent de lire de l'information dans un élément de la liste sont les suivantes :

- max_nl_over_hl(int option, char* s) : si la variable *option* est égale à zéro, alors cette fonction retourne le nom de la carte ayant le plus grand ratio n^l/h^l pour la technologie de type *s*. Sinon, elle retourne la valeur de ce ratio;
- max_mt() : cette fonction retourne le nombre de fentes du routeur qui peut accueillir le plus grand nombre de cartes d'interfaces;
- get_max_router() : cette fonction retourne le nom du routeur qui peut accueillir le plus grand nombre de cartes d'interfaces;
- get_max_bit_rate(char* s) : cette fonction retourne le nom du lien appartenant à l'ensemble *s* ayant la plus grande capacité;
- get_bit_rate(char* s) : cette fonction retourne la capacité du lien nommé *s*;
- get_capacity(int m_t, char* n) : cette fonction retourne la capacité de transfert simultané du routeur nommé *n* ayant *m_t* fentes;
- get_port(int l) : cette fonction retourne le nombre de ports de la carte d'interface nommée *l*;
- get_slot(int l) : cette fonction retourne le nombre de fentes que la carte appelée *l* utilise dans un routeur;
- get_price(int type) : cette fonction retourne le coût du routeur, de la carte d'interface ou du lien de nom *type*.

3.2 Environnement d'expérimentation

Avant de présenter un exemple détaillé, nous présenterons l'environnement d'expérimentation. Tout d'abord, nos tests ont été effectués sur un ordinateur ayant un processeur à 1700 MHz, un disque dur de 40 GB ainsi qu'une mémoire vive de 512 MB. Afin d'effectuer ces tests, nous avons eu besoin d'un fichier dans lequel toutes les données nécessaires étaient présentes. Nous présentons dans cette section le format de ce fichier.

Afin de résoudre le problème de planification des POPs, nos programmes doivent d'abord acquérir les spécifications du problème. Celles-ci sont fournies dans un

fichier de données. Nos deux méthodes utilisent le même fichier de données appelé *optimization_pop.dat*. Puisque OPL utilise une syntaxe particulière, nous avons donc décidé d'en faire notre standard et d'adapter notre heuristique afin qu'elle puisse utiliser le même fichier.

Comme plusieurs tests ont dû être effectués, nous avons implanté un programme dans le but de générer ces fichiers de façon automatique. Le procédé est très simple. Nous devons seulement indiquer le nombre de clients, le nombre d'autres POPs ainsi que le nombre de sites possibles dans notre POP pour installer des routeurs, comme illustré à la Figure 3.4. L'information qui est contenue dans ces fichiers est décrite dans les paragraphes suivants.

```
The file "optimization_pop.dat" will be created

Enter the number of clients (I): 30
Enter the number of O-POPs (J): 20
Enter the number of locations (K): 5
```

Figure 3.4 Générateur de fichier de données

Deux types de liens (R^A = {OC-3, OC-12}) sont utilisés dans le réseau d'accès, deux dans le réseau dorsal (R^B = {OC-12, OC-48}) et un seul dans le réseau entre les routeurs (R^P = {OC-192}). Les coûts des types de routeurs sont présentés au Tableau 3.1 et les coûts des cartes d'interface sont présentés au Tableau 3.2. Ces coûts incluent l'achat ainsi que le coût d'installation (comprenant l'espace, le support, les câbles, les connecteurs et la main d'œuvre). Il est à noter que toutes les cartes d'interface utilisent une seule fente. Les coûts pour interconnecter deux routeurs dans un POP sont présentés au Tableau 3.3. Ce coût inclut les câbles et la main d'œuvre.

Pour chaque client, le nombre de liens d'accès est généré aléatoirement entre un et quatre avec un maximum de deux liens OC-3 et de deux liens OC-12. Pour chaque autre POP, le nombre de liens dorsaux est aussi sélectionné aléatoirement entre un et quatre, avec un maximum de deux liens OC-12 et de deux liens OC-48. Il est important

de noter que les liens fonctionnent en mode *full duplex*, c'est-à-dire qu'ils peuvent transmettre de l'information en même temps qu'ils en reçoivent.

Le trafic dans un réseau peut prendre plusieurs formes. Par exemple, les modèles avec exactement un emplacement central utilisent souvent une demande de type *many-to-one*. Dans ce cas, chaque nœud communique seulement avec l'emplacement central. Pour notre problème, nous avons modélisé le trafic comme étant de type *many-to-many*. Cela signifie que chaque nœud dans le réseau peut échanger des données avec chaque autre nœud. Cette demande de trafic est générée aléatoirement en utilisant une distribution uniforme dans un intervalle de 0 à 1 Mbps entre chaque paire de clients, de 0 à 40 Mbps entre chaque paire d'autres POPs et de 0 à 1 Mbps entre chaque client et un autre POP.

Tableau 3.1 Caractéristiques des différents types de routeur

Type	Nombre de fentes	Capacité (Gbps)	Coût ($)
A	7	20	10 000
B	14	80	20 000

Tableau 3.2 Caractéristiques des différentes cartes d'interface

Type	Technologie	Nombre de ports	Coût ($)
A	OC-3	4	3 000
B	OC-3	16	16 000
C	OC-12	4	10 000
D	OC-48	4	30 000
E	OC-192	1	50 000

Tableau 3.3 Caractéristiques des différents types de liens d'interconnexion

Type	Capacité (Mbps)	Coût ($)
OC-3	155	500
OC-12	622	500
OC-48	2 500	500
OC-192	10 000	500

3.3 Mise en œuvre de la méthode

Dans cette section, nous nous proposons de détailler un exemple dans le but de montrer le fonctionnement général de notre méthode heuristique. Nous commencerons par présenter les données du problème et ensuite nous appliquerons les différentes étapes de la méthode heuristique. Cette section utilise la notation ainsi que les équations du chapitre précédent.

Pour cet exemple, le nombre de clients dans I est 30 ($|I| = 30$), le nombre d'autres POPs dans J est 20 ($|J| = 20$) et le nombre maximum de routeurs qui peut être installé dans le POP est cinq ($|K| = 5$). Pour chaque autre POP, le nombre et le type de liens dorsaux sont présentés au Tableau 3.4. Le Tableau 3.5 présente le nombre et le type de liens d'accès pour chacun des clients.

Tableau 3.4 Nombre de liens dorsaux (et leur type) pour chaque autre POP

O-POP	Nombre de liens	
	OC-12	OC-48
1	2	0
2	0	2
3	1	2
4	2	1
5	1	0
6	1	0
7	2	0
8	2	2
9	0	1
10	1	0

O-POP	Nombre de liens	
	OC-12	OC-48
11	2	0
12	1	0
13	2	0
14	1	1
15	2	2
16	1	1
17	1	2
18	0	2
19	2	2
20	0	1

Tableau 3.5 Nombre de liens d'accès (et leur type) pour chaque client

Client	Nombre de liens	
	OC-3	OC-12
1	1	1
2	2	0
3	1	0
4	0	2
5	2	2
6	1	2
7	0	1
8	0	2
9	2	1
10	1	0
11	1	1
12	1	0
13	0	2
14	2	1
15	1	2

Client	Nombre de liens	
	OC-3	OC-12
16	2	0
17	2	1
18	1	2
19	2	1
20	2	0
21	1	2
22	2	0
23	1	1
24	0	1
25	2	0
26	2	0
27	2	2
28	2	1
29	1	2
30	0	1

La première étape consiste à calculer les valeurs de A et de B. Ces valeurs peuvent être calculées à l'aide des équations (2.35) et (2.36) présentées au chapitre deux.

$$A = \max_{l \in L_r : r \in R^P} \left\{ \frac{1}{1} \right\} = 1 .$$

$$B = \left(\frac{37}{\max\limits_{l \in L_r} \left\{ \frac{16}{1} \right\}} + \frac{31}{\max\limits_{l \in L_r} \left\{ \frac{4}{1} \right\}} \right) + \left(\frac{24}{\max\limits_{l \in L_r} \left\{ \frac{4}{1} \right\}} + \frac{19}{\max\limits_{l \in L_r} \left\{ \frac{4}{1} \right\}} \right)$$

$$= \left(\frac{37}{16} + \frac{31}{4} \right) + \left(\frac{24}{4} + \frac{19}{4} \right) = 10.0625 + 10.75 = 20.8125.$$

Après avoir calculé A et B, nous pouvons vérifier la faisabilité de notre problème en utilisant la proposition 4. Ainsi, l'inéquation (2.44) nous donne :

$$20.8125 \le \lceil (1 \bullet 14 + 1)/2 \rceil \lfloor (1 \bullet 14 + 1)/2 \rfloor \Leftrightarrow 20.8125 \le 56$$

et l'inéquation (2.45) nous donne :

$$5 \ge \left\lceil \frac{1}{2}\left((1 \bullet 14 + 1) - \left((1 \bullet 14 + 1)^2 - 4 \bullet 1 \bullet 20.8125 \right)^{1/2} \right) \right\rceil \Leftrightarrow 5 \ge 2 .$$

Puisque les deux inéquations sont respectées, notre problème est donc réalisable. La deuxième étape consiste à trouver n_{min} et n_{max} afin de savoir le nombre minimum et le nombre maximum de routeurs que nous pouvons installer dans notre POP. Ces deux valeurs peuvent être trouvées à l'aide de la partie droite des équations (2.33) et (2.34) du chapitre deux.

$$n_{\min} = \left\lceil \frac{1}{2}\left((1 \cdot 14 + 1) - \left((1 \cdot 14 + 1)^2 - 4 \cdot 1 \cdot 20.8125 \right)^{1/2} \right) \right\rceil = 2.$$

$$n_{\max} = \min\left\{ \left\lfloor \frac{1}{2}\left((1 \cdot 14 + 1) + \left((1 \cdot 14 + 1)^2 - 4 \cdot 1 \cdot 20.8125 \right)^{1/2} \right) \right\rfloor, 5 \right\} = 5.$$

Nous pouvons maintenant commencer le processus d'itération. Nous commençons donc par installer deux routeurs ($n = 2$) aux deux premiers emplacements ($k = 0$ et $k = 1$). Ensuite, nous calculons le nombre de cartes d'interface nécessaires afin de satisfaire tous les clients et tous les autres POPs. Pour trouver le nombre de cartes nécessaires, nous effectuons le calcul suivant en traitant séparément les clients et les autres POPs :

$$\text{Pour les clients : } \frac{1}{n^{l(r)}} \sum_{i \in I} q_i^r \quad \text{pour tous } r \in R^A$$

$$\text{Pour les O-POPs : } \frac{1}{n^{l(r)}} \sum_{j \in J} q_j^r \quad \text{pour tous } r \in R^B$$

où $n^{l(r)}$ représente le nombre de ports de la carte de type r ayant le plus grand ratio n^l/h^l. Avec ces calculs, nous obtenons le nombre de cartes représentées au Tableau 3.6.

Tableau 3.6 Nombre de cartes nécessaires

Type	Technologie	n^l	h^l	Nombre de cartes	
				clients	O-POP
A	OC-3	4	1	0	0
B	OC-3	16	1	3	0
C	OC-12	4	1	8	6
D	OC-48	4	1	0	5
E	OC-192	1	1	0	0

Cependant, si $R^A \cap R^B \neq \varnothing$, nous devons vérifier que nous n'avons pas installé trop de cartes. Cette vérification est nécessaire puisque nous avons calculé le nombre de cartes nécessaires pour les clients et pour les autres POPs séparément. Dans notre exemple, puisque la technologie OC-12 est utilisée pour les liens d'accès et pour les liens dorsaux, nous devons vérifier l'équation suivante.

$$\left\lceil \frac{1}{n^{l(r)}} \left(\sum_{i \in I} q_i^r + \sum_{j \in J} q_j^r \right) \right\rceil = \frac{1}{n^{l(r)}} \sum_{j \in J} q_j^r + \frac{1}{n^{l(r)}} \sum_{i \in I} q_i^r \, .$$

Si l'équation est vérifiée, alors le nombre de cartes est minimum. Sinon, nous devons diminuer de un le nombre de cartes d'accès pour cette technologie.

Après avoir déterminé le nombre de cartes nécessaires pour les clients et pour les autres POPs, nous devons maintenant calculer la capacité à réserver dans chaque routeur pour les liens inter-routeurs. Puisque nous avons installé deux routeurs (n = 2), nous aurons besoin d'au moins un lien pour les relier. Nous devrons donc réserver une fente ainsi que 10 000 Mbps dans chacun des deux routeurs.

Ensuite, nous déterminons le nombre de routeurs (parmi les n installés) qui seront des routeurs dorsaux. Le but d'utiliser un certain nombre de routeurs dorsaux est

de pouvoir regrouper tous les liens dorsaux au même endroit. Ceci peut faciliter de beaucoup la gestion du point de présence. Le nombre minimum de routeurs dorsaux (*b*) est calculé à partir du nombre de cartes nécessaires afin de pouvoir connecter tous les autres POPs. Puisque nous avons besoin de onze cartes (6 + 5), un seul routeur dorsal sera nécessaire.

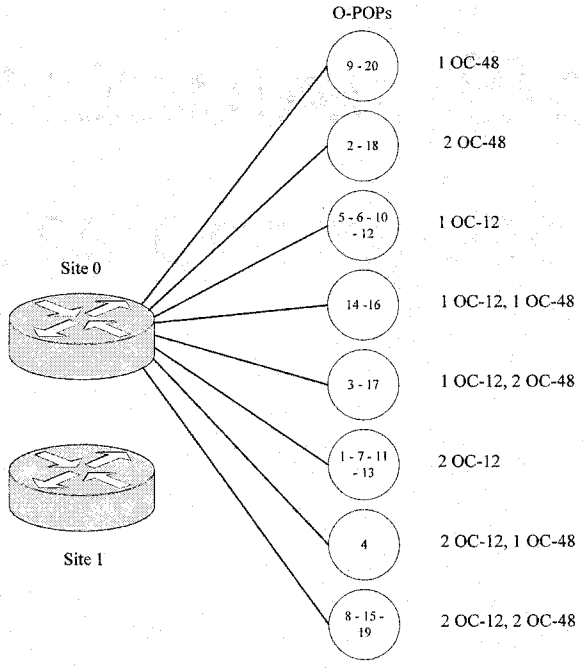


Figure 3.5 Topologie du réseau après assignation des autres POPs

Nous allons donc faire varier le nombre de routeurs dorsaux *b* entre un et le nombre de routeurs installés dans le POP, c'est-à-dire *n*. Commençons par $b = 1$.

Après avoir déterminé la valeur de *b*, nous installons les cartes du réseau dorsal dans les *b* routeurs dorsaux. Si toutes les cartes ont été placées, nous assignons les liens dorsaux aux ports. Ensuite, nous connectons les routeurs dorsaux entre eux et calculons le flot passant sur chaque lien. Puisque notre réseau ne contient qu'un seul routeur dorsal, nous sautons cette étape. Jusqu'à présent, la topologie de notre réseau ressemble à celle de la Figure 3.5 où seuls les autres POPs ont été assignés. À des fins de

compréhension, nous avons représenté tous les autres POPs ayant le même nombre de liens dans un cercle. Les chiffres dans les cercles représentent le numéro du O-POP.

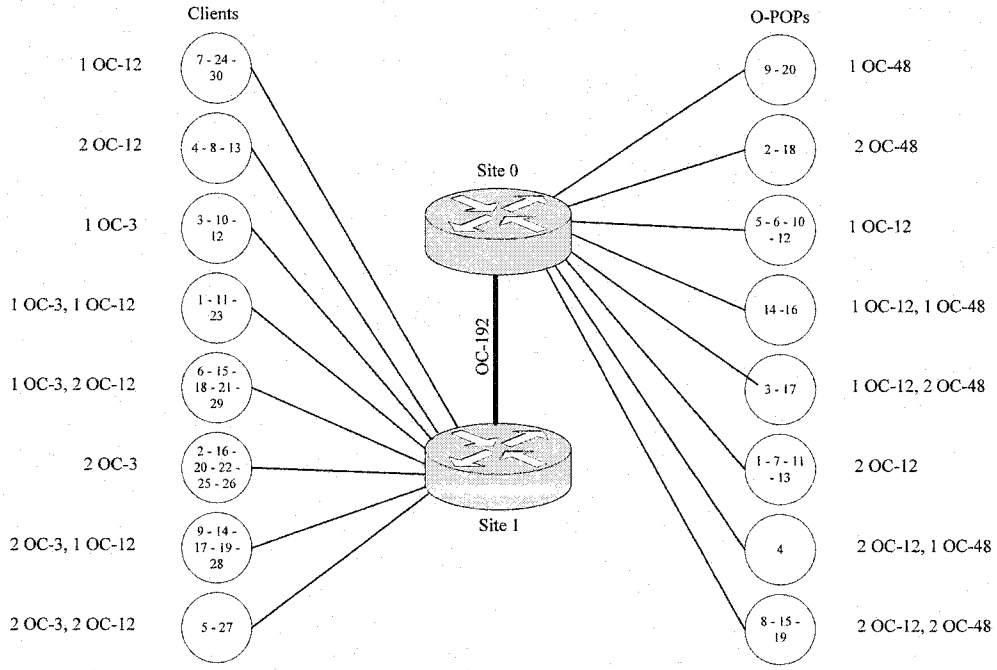


Figure 3.6 Topologie du réseau après assignation des autres POPs et des clients

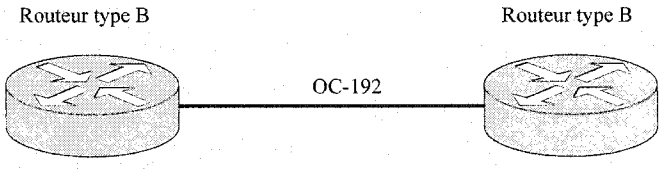
C'est maintenant le temps d'installer les cartes des liens d'accès dans les routeurs d'accès. En fait, il existe $(n - b)$ routeurs d'accès disponibles. Cependant, il se peut qu'il n'y ait pas assez de fentes disponibles dans ces routeurs afin de répondre à la demande de tous les clients. Nous devrons alors utiliser les fentes restantes disponibles dans les routeurs dorsaux. Une fois les cartes installées, nous assignons les liens d'accès aux ports. Ensuite, nous connectons les n routeurs entre eux à l'aide de liens de capacité maximale (OC-192). Maintenant que tous les nœuds sont connectés (voir Figure 3.6), nous pouvons calculer le flot sur chaque lien en appliquant la politique de routage. Cette politique est basée sur les plus courts chemins. Puisque les matrices de trafic sont très grandes (30 x 30 pour le trafic entre les clients, 20 x 20 pour le trafic entre les autres POPs et 30 x 20 pour le trafic entre les clients et les autres POPs), nous ne présenterons

pas les calculs afin de trouver le flot sur chaque lien. En fait, le flot est simplement trouvé en effectuant la sommation des trafics circulant sur un lien.

Si la capacité d'un lien inter-routeur est insuffisante, alors on ajoute des liens OC-192 jusqu'à ce que le flot sur ce lien soit plus petit ou égal à la capacité totale des liens. Par contre, si la capacité d'un lien d'accès ou d'un lien dorsal est plus petite que le flot passant sur ce lien, alors cette configuration n'est pas réalisable. Si b est plus petit que n, alors $b := b + 1$ et on recommence une nouvelle itération. Si b est égal à n et que n est plus petit que n_{max}, alors $n := n + 1$ et on recommence une nouvelle itération. Sinon, il n'existe pas de solution à ce problème.

En ce qui concerne notre problème, aucun flot ne dépasse la capacité des liens installés. L'étape suivante consiste à vérifier le nombre de fentes ainsi que la capacité résiduelle à chaque emplacement. S'il y a trop de cartes installées dans un routeur ou que la capacité résiduelle est plus petite que zéro, alors nous sommes en présence d'une solution qui n'est pas réalisable. Si tout est conforme, alors nous calculons le coût de la configuration actuelle. Notre configuration actuelle a un coût de 478 500 $. Si ce coût est plus petit que le coût trouvé jusqu'à présent, alors nous gardons cette configuration en mémoire et nous remplaçons la valeur du meilleur coût trouvé par la valeur du coût actuel. Une première itération est maintenant terminée. Nous effectuons toutes ces étapes pour les différentes valeurs de b et de n. La solution finale est celle dont le coût est minimum. La Figure 3.7 représente la meilleure solution trouvée par notre heuristique. Le routeur installé au site 0 est un routeur dorsal, tandis que celui installé au site 1 est un routeur d'accès. Cette solution a un coût de 478 500 $.

À titre de comparaison, le coût de la solution optimale trouvé avec OPL est de 464 500 $. Ceci représente une différence de 3.01%



Type de cartes	Nombre
A	0
B	0
C	6
D	5
E	1

Type de cartes	Nombre
A	0
B	3
C	8
D	0
E	1

Figure 3.7 Solution de l'exemple à l'aide de l'heuristique

Maintenant que nous avons démontré le fonctionnement général de notre heuristique, nous allons présenter la méthodologie qui sera utilisée afin d'évaluer la performance de notre heuristique.

3.4 Plan d'expérience

Le plan d'expérience propose la configuration des tests qui ont été effectués afin d'évaluer la performance de notre heuristique. Cinq tests de 40 configurations différentes ont été expérimentés. Pour ce faire, nous avons fait varier le nombre de clients entre 10 et 100 et le nombre d'autres POPs entre 10 et 25. Les données utilisées (technologie, type de carte, type de routeur, liens, etc.) pour effectuer les tests sont celles qui ont été présentées à la section 3.2.

Après avoir effectué les tests, nous analyserons les résultats. Une première analyse sera effectuée sur la différence de coût entre notre méthode heuristique et une borne inférieure. La détermination de cette borne fera l'objet de la prochaine section. Une deuxième analyse sera réalisée afin de comparer les temps d'exécution de l'heuristique et de la méthode exacte. Puisque notre modèle de programmation mathématique est de type NP-complet, nous pouvons nous attendre à obtenir un temps d'exécution extrêmement long pour des problèmes de grande taille. Ces deux analyses sont effectuées à la section 3.5

Afin d'évaluer la qualité de notre méthode heuristique, il est essentiel de la comparer avec une valeur de référence. Il est préférable de comparer nos résultats avec une méthode exacte car nous pouvons mieux nous situer par rapport à la solution optimale. Cependant, ce n'est pas toujours possible. Dans ce cas, nous devons comparer nos résultats avec une borne inférieure. Plus cette borne est élevée, mieux elle se situe par rapport à la solution optimale. Il est alors plus facile d'avoir une bonne estimation de la qualité de la solution fournie par l'heuristique. Cependant, le temps de calcul d'une bonne borne est plus important que celui d'une borne grossière. La Figure 3.8 montre les écarts respectifs entre une solution heuristique et une solution optimale puis une borne inférieure. Puisque notre modèle de programmation mathématique a un temps d'exécution long (plus de 28 000 secondes pour un problème ayant 40 clients et 10 autres POPs), nous avons décidé de relâcher toutes les contraintes de trafic (contraintes (2.18) à (2.28)). Le temps d'exécution passe alors à moins de quatre secondes. Nous utiliserons donc le modèle sans trafic comme borne inférieure.

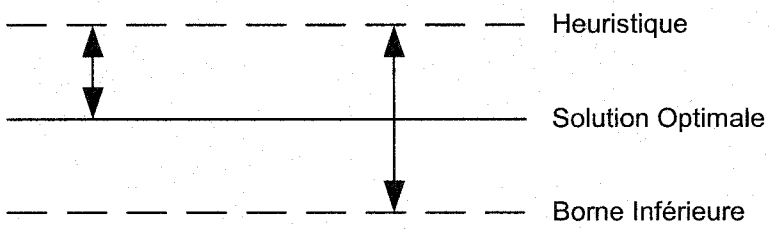


Figure 3.8 Heuristique vs solution optimale et borne inférieure

3.5 Analyse des résultats

Comme nous l'avons mentionné, l'analyse des résultats se divisera en trois volets différents. Le premier sera la génération des résultats. Ensuite, nous évaluerons la qualité des solutions obtenues et nous terminerons par l'analyse du temps d'exécution de nos deux méthodes.

3.5.1 Génération des résultats

Les Tableaux 3.8 à 3.11 contiennent un résumé des cinq tests que nous avons effectués. Les deux premières colonnes représentent le nombre de clients ($|I|$) et le nombre de POPs ($|J|$) connectés au POP que nous avons optimisé. Les trois colonnes suivantes représentent respectivement l'écart minimum, l'écart maximum et la moyenne des écarts entre le coût trouvé par l'heuristique et celui trouvé à l'aide de la borne inférieure. Les deux dernières colonnes représentent le temps moyen requis par la méthode heuristique et par la borne inférieure avant de trouver la solution. Pour avoir plus de détails sur les résultats obtenus, consulter l'annexe.

Tableau 3.8 Résultats des tests pour $|J| = 10$

I	J	Écart			Temps moyen	
		Min (%)	Max (%)	Moy (%)	Heuristique (s)	OPL (s)
10	10	1.7	9.2	4.2	0.2	0.3
20	10	0.0	6.7	4.3	0.2	0.3
30	10	2.7	6.3	4.9	0.8	6.9
40	10	1.5	3.2	2.6	1.4	8.1
50	10	0.0	3.1	0.9	2.2	5.0
60	10	1.1	2.6	2.2	3.0	471.4
70	10	0.5	2.3	1.2	4.8	353.3
80	10	1.2	1.2	1.2	10.4	11108.0
90	10	0.8	1.9	1.2	21.0	10395.4
100	10	0.3	1.1	0.6	39.8	9127.8

Tableau 3.9 Résultats des tests pour $|J| = 15$

I	J	Écart			Temps moyen	
		Min (%)	Max (%)	Moy (%)	Heuristique (s)	OPL (s)
10	15	0.0	4.6	1.8	0.6	0.3
20	15	3.7	7.5	5.0	0.6	3.2
30	15	0.9	4.3	2.8	1.0	3.2
40	15	0.0	3.5	1.9	1.2	289.0
50	15	1.1	2.9	2.1	2.4	3273.4
60	15	0.5	2.0	1.0	3.0	494.1
70	15	0.0	1.7	0.9	4.4	1524.2
80	15	0.0	1.6	1.0	8.6	4345.2
90	15	0.3	0.9	0.6	15.8	20136.4
100	15	0.0	0.8	0.4	28.0	21233.2

Tableau 3.10 Résultats des tests pour $|J| = 20$

I	J	Écart			Temps moyen	
		Min (%)	Max (%)	Moy (%)	Heuristique (s)	OPL (s)
10	20	0.0	3.7	1.4	0.8	3.5
20	20	1.9	3.3	2.8	1.0	9.6
30	20	0.8	3.0	1.4	1.0	6.5
40	20	0.0	2.7	1.7	1.2	720.7
50	20	0.9	1.9	1.3	2.4	678.8
60	20	0.0	2.2	0.9	4.0	1986.1
70	20	0.0	1.2	0.8	4.4	12085.3
80	20	0.6	1.8	1.2	11.4	8477.7
90	20	0.0	1.0	0.6	15.4	14779.2
100	20	0.6	1.0	0.8	11.6	3312.2

Tableau 3.11 Résultats des tests pour $|J| = 25$

I	J	Écart			Temps moyen	
		Min (%)	Max (%)	Moy (%)	Heuristique (s)	OPL (s)
10	25	0.7	3.5	1.5	0.6	6.1
20	25	0.8	3.3	1.6	1.0	4.8
30	25	0.0	2.3	1.1	1.0	688.4
40	25	0.0	2.3	1.3	2.0	402.1
50	25	0.0	1.8	0.4	2.8	429.9
60	25	0.0	1.8	0.9	3.2	8457.3
70	25	0.3	1.1	0.8	4.4	29184.8
80	25	0.3	1.0	0.7	8.0	13573.5
90	25	0.0	1.0	0.5	4.2	6038.2
100	25	0.0	0.8	0.3	11.4	932.8

Puisque ces tableaux sont une moyenne de tous les tests effectués, ils ne représentent pas tous les faits. En effet, si nous regardons les tableaux des résultats situé à l'annexe A, nous pouvons remarquer que la borne inférieure n'a pas été capable de résoudre tous les problèmes. Pour neuf problèmes différents, le programme OPL a tout simplement manqué de mémoire et n'a pas été en mesure de retourner des résultats. Cependant, notre heuristique a été capable de générer une solution en un temps très raisonnable (quelques secondes).

Pour certains problèmes, nos deux méthodes n'ont pas été capables de trouver une solution réalisable. Cette situation s'explique par le fait que les fichiers de données sont générés aléatoirement et que nous avons limité le nombre de routeurs à l'intérieur d'un POP à un maximum de cinq ($|K| = 5$). Par exemple, si notre programme génère un fichier qui contient un trop grand nombre de liens, alors cinq routeurs ne seront pas suffisants pour répondre à toutes ces demandes.

Puisque notre borne inférieure ne tient pas compte du trafic, il se peut qu'elle génère des solutions qui ne sont pas réalisables. Cette situation ne s'est présentée qu'une seule fois. En effet, OPL a généré un résultat, tandis que notre heuristique a indiqué qu'il n'y avait pas de solutions possibles car le flot d'un lien dorsal dépassait la capacité de ce dernier.

3.5.2 Analyse de la qualité des solutions

Dans cette section, nous nous proposons d'analyser la différence de coût entre la méthode heuristique et la borne inférieure. La Figure 3.9 représente la différence (en pourcentage) entre le coût trouvé par l'heuristique et celui trouvé par la borne inférieure. La première constatation que nous pouvons faire est que la différence de coût ne dépasse jamais 5%. En effet, à quatre exceptions près, nous aurions pu avoir une différence de moins de 3%. Si nous effectuons la moyenne des écarts, nous obtenons une différence de 1.6%.

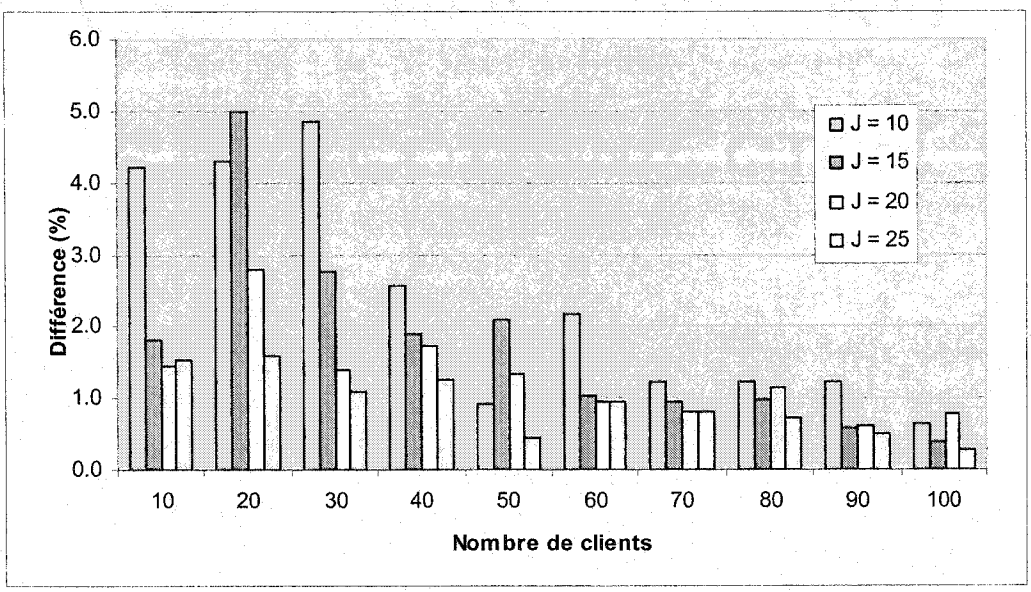


Figure 3.9 Différence de coût entre l'heuristique et la borne inférieure

Une deuxième constatation peut être effectuée en fixant le nombre de clients (par exemple $|I| = 30$) et en faisant varier le nombre d'autres POPs. En effet, nous pouvons remarquer que la différence de coût est inversement proportionnelle au nombre d'autres POPs. Cette même constatation peut être effectuée en fixant le nombre d'autres POPs tout en faisant varier le nombre de clients. Plus le nombre de clients augmente, plus la différence de coût diminue. De façon générale, nous pouvons affirmer que plus le nombre de nœuds connectés au POP est grand, plus la différence de coût sera petite.

Ceci s'explique par le fait que le coût total du POP est proportionnel au nombre de nœuds connectés à celui-ci. En effet, une différence de 20 000$ est peu significative si le coût du POP est élevé. Par contre, si le coût du POP est faible, une différence de 20 000$ peut devenir très significative.

Pour terminer cette analyse de coût, nous pouvons prévoir que plus le nombre de nœuds sera élevé, meilleure se comportera notre heuristique.

3.5.3 Analyse du temps d'exécution

Nous avons implanté une méthode heuristique car nous savions que la méthode exacte allait prendre beaucoup trop de temps avant de générer des résultats et ce, même pour des problèmes de taille « normale ». Il est donc intéressant de comparer le temps d'exécution de ces deux méthodes. Encore une fois, nous comparerons notre heuristique avec la borne inférieure puisque c'est avec celle-ci que nous avons généré nos résultats.

Comme nous pouvons le constater en analysant les Tableaux 3.8 à 3.11, le temps d'exécution de la borne inférieure augmente très rapidement. En effet, nous avons attendu plus de 71 000 secondes pour un problème qui a fini par manquer de mémoire. En regardant les temps d'exécution à l'annexe A, nous pouvons remarquer que le temps ne varie pas toujours en fonction de la taille du problème. Cela s'explique par le fonctionnement même de l'algorithme de séparation et évaluation progressive. Dans cet algorithme, tout dépend de la solution initiale trouvée par OPL. Si cette solution est proche de l'optimum, alors le temps sera moins long que si la solution est éloignée de l'optimum. Par exemple, un problème de petite taille ayant une solution initiale éloignée de l'optimum peut prendre plus de temps à résoudre qu'un problème de grande taille ayant une solution initiale proche de l'optimum.

En ce qui concerne notre heuristique, nous pouvons affirmer qu'elle génère des résultats en un temps plus que raisonnable. Le temps d'exécution du plus long problème a été de 42 secondes.

CHAPITRE 4

CONCLUSION

Après avoir testé les algorithmes et analysé nos résultats, il ne reste plus qu'à conclure ce mémoire. Dans ce dernier chapitre, nous effectuerons un bref rappel des points que nous avons abordés. Par la suite, nous exposerons les limitations de nos algorithmes et nous terminerons avec des indications pour des travaux futurs.

4.1 Synthèse des travaux

Dans ce mémoire, nous avons présenté deux méthodes afin de résoudre le problème de planification des points de présence dans les réseaux IP. La première méthode consiste en un modèle de programmation mathématique que nous avons traduit en langage de programmation OPL et résolue avec CPLEX. Le modèle a pour but de minimiser une fonction objectif (le coût du POP) tout en respectant un certain nombre de contraintes. Elle a l'avantage de générer des solutions optimales au détriment d'un temps d'exécution très long. Puisque nous avons démontré que ce problème était NP-complet, il est peu probable que cette méthode soit capable de résoudre des problèmes de grande taille. En effet, pour des problèmes de grande taille, le temps d'exécution serait extrêmement long et même la mémoire de l'ordinateur pourrait être insuffisante. Pour ces raisons, nous avons relâché toutes les contraintes de trafic afin d'obtenir une borne inférieure. Cette borne nous a servi de référence afin d'évaluer notre heuristique.

Notre deuxième approche est une méthode heuristique que nous avons implantée en langage C++. Le but de cette dernière est de générer des résultats quasi-optimaux en temps raisonnables. C'est-à-dire que nous avons fait un compromis entre la qualité de la solution et le temps d'exécution.

Afin de répondre à la première question que nous nous étions posé dans l'introduction, nous pouvons affirmer que nos deux modèles représentent très bien la réalité puisque les technologies d'aujourd'hui et le trafic ont été considérés.

Après avoir implanté les deux modèles, nous les avons comparés afin d'évaluer la qualité de notre heuristique. Pour ce faire, nous avons effectué cinq tests de quarante configurations différentes. Chaque configuration a été générée aléatoirement selon une distribution uniforme par un programme que nous avons implémenté. En comparant les résultats des deux modèles, nous pouvons conclure que les solutions trouvées par notre heuristique sont excellentes. En effet, la moyenne des écarts est de 1.6% et la plus grande différence est de seulement 4.9%. À plusieurs reprises, notre heuristique a même trouvé la solution optimale. En ce qui concerne le temps d'exécution, nous pouvons affirmer que notre heuristique est beaucoup plus performante. En effet, le temps maximum que notre heuristique a mis pour trouver une solution est de 42 secondes. Ce temps est pratiquement négligeable comparé aux 71000 secondes de la borne inférieure.

4.2 Limitations des travaux

Comme nous l'avons déjà mentionné, la méthode optimale n'est plus efficace pour des problèmes de grande taille. Même la borne inférieure prend trop de temps avant de générer des solutions et parfois la mémoire de l'ordinateur devient insuffisante. Il est donc difficile d'évaluer la performance de notre heuristique pour ces problèmes puisque nous ne sommes pas en mesure de générer une valeur de comparaison. Cependant, dans le chapitre portant sur les résultats (chapitre 3), nous avons remarqué que plus la taille du problème augmente, plus notre heuristique est performante. Ce comportement peut donc être supposé pour les problèmes de grande taille. Le seul moyen de vérifier cette hypothèse serait de trouver une façon plus efficace de calculer la borne inférieure.

4.3 Indications des travaux futurs

En ce qui concerne les travaux futurs, quelques pistes peuvent être intéressantes. Premièrement, dans ce mémoire, nous considérons seulement la conception de nouveaux

points de présence. C'est-à-dire que tout est construit à partir de zéro. Il serait fort pertinent de pouvoir considérer l'expansion d'un POP. En effet, si de nouveaux clients veulent se connecter à un POP déjà existant, comment pourrait-on optimiser ce dernier en tenant compte des équipements qui sont déjà installés?

Une deuxième piste réside dans le développement de méthodes exactes plus performantes. Avec l'avancement de la programmation mathématique et l'augmentation de la puissance des ordinateurs, de nouveaux algorithmes seront peut-être en mesure de résoudre des problèmes de grande taille en un temps raisonnable.

BIBLIOGRAPHIE

BRASSARD, G. et BRATLEY, P., "Algorithmique conception et analyse", Masson, 1987.

CHAMBERLAND, S., SANSO, B. et MARCOTTE O. "Topological design of two-level telecommunication networks with modular switches", Operations Research, vol.48 pp. 745-760, 2000.

CHAMBERLAND, S. et SANSO, B., "Topological expansion of multiple-ring metropolitain area networks", Networks, vol. 36, pp. 210-224, 2000.

CHAMBERLAND, S. et SANSO, B., "On the design problem of multitechnology networks", INFORMS Journal on Computing, vol. 13, pp. 245-256, 2001.

GAREY, M.R., et JOHNSON, D.S., "Computers and Intractability – A guide to the theory of NP-Completeness", Freeman, 1979.

HENTENRYCK, P.V., "ILOG OPL Studio 3.5 – The Optimization Language", MIT, 2001.

ILOG Inc., "Using the CPLEX Callable Library and CPLEX Mixed Integer Library", Incline Village: NV, 2000.

KLINCEWICZ, J.G., "Hub location in backbone/tributary network design: A review", Location Science, vol. 6, pp. 307-335, 1998.

MANN-RUBINSON, T.C. et TERPLAN, K., "Network Design – Management and Technical Perspectives", CRC Press, 1998.

NEMHAUSER, G.L. et WOLSEY, L.A, "Integer and Combinatorial Optimization", Wiley,1988.

PIERRE, S., ELGIBAOUI, A., "A Tabu-Search Approach for Designing Computer-Network Topologies with Unreliable Components", IEEE Transactions on Reliability, vol. 46, pp. 350 – 359, 1997.

PIERRE, S., LEGAULT, G., "A Genetic Algorithm for Designing Distributed Computer Network Topologies" IEEE Transactions on Reliability, vol. 28, pp. 249 – 258, 1998.

STALLINGS, W., "Data & Computer communications", Prentice Hall, 6th edition, 1996.

TANENBAUM, A., "Réseaux", Prentice Hall, 3e édition, 1996.

ANNEXE A

Résultats détaillés des tests

Tableau A.1 Résultats du test 1

			TEST 1				
Problème	I	J	Heuristique	Temps	OPL*	Temps	Écart
			$	(s)	$	(s)	%
1	10	10	202000	<1	185000	0.7	9.2
2	20	10	242000	<1	232000	0.33	4.3
3	30	10	382500	1	372500	10.88	2.7
4	40	10	412500	2	401500	3.88	2.7
5	50	10	478500	3	478500	2.83	0.0
6	60	10	745500	3	737500	353.88	1.1
7	70	10	781500	5	764500	203.44	2.2
8	80	10	1179000	12	(OM)1165000	14048.66	n/a
9	90	10	1155000	26	1134000	26879.31	1.9
10	100	10	1221000	40	1208000	5116.05	1.1
11	10	15	236000	<1	236000	0.75	0.0
12	20	15	422500	<1	402500	1.59	5.0
13	30	15	392500	1	381500	1.17	2.9
14	40	15	504500	1	487500	13.84	3.5
15	50	15	735500	2	(OM)721500	12341.03	n/a
16	60	15	771500	3	756500	50.92	2.0
17	70	15	871500	4	871500	288.63	0.0
18	80	15	813500	13	800500	119	1.6
19	90	15	1195000	19	1184000	39059.69	0.9
20	100	15	1683000	41	(OM)	71769.19	n/a
21	10	20	396500	1	382500	1.48	3.7
22	20	20	432500	1	418500	16.56	3.3
23	30	20	478500	1	464500	13.03	3.0
24	40	20	544500	1	531500	14.63	2.4
25	50	20	785500	3	774500	326.81	1.4
26	60	20	911500	4	911500	50.5	0.0
27	70	20	1203000	5	(OM)1195000	24425.77	n/a
28	80	20	1249000	10	1242000	2470.55	0.6
29	90	20	1747000	5	1733000	1360.63	0.8
30	100	20	1703000	15	1686000	9157.7	1.0
31	10	25	502500	<1	485500	1.88	3.5
32	20	25	502500	1	498500	5.31	0.8
33	30	25	813500	1	805500	505.78	1.0
34	40	25	518500	2	518500	7.28	0.0
35	50	25	865500	3	865500	67.61	0.0
36	60	25	1203000	4	1182000	24833.38	1.8
37	70	25	1263000	4	1253000	3124.39	0.8
38	80	25	1761000	4	1744000	8686.7	1.0
39	90	25	1767000	4	1750000	17075.22	1.0
40	100	25	1733000	20	1719000	4080.97	0.8

(OM) : Out of Memory

Tableau A.2 Résultats du test 2

			TEST 2				
Problème	I	J	Heuristique	Temps	OPL*	Temps	Écart
			$	(s)	$	(s)	%
1	10	10	196000	<1	192000	0.06	2.1
2	20	10	182000	<1	172000	0.26	5.8
3	30	10	388500	<1	365500	7.5	6.3
4	40	10	478500	1	471500	10.11	1.5
5	50	10	464500	2	450500	13.11	3.1
6	60	10	721500	3	703500	507.36	2.6
7	70	10	757500	5	740500	202.69	2.3
8	80	10	813500	17	803500	321.09	1.2
9	90	10	1175000	25	1161000	6507.72	1.2
10	100	10	1201000	37	1197000	138.69	0.3
11	10	15	236000	<1	226000	0.27	4.4
12	20	15	342500	1	318500	5.01	7.5
13	30	15	462500	1	458500	5.63	0.9
14	40	15	478500	1	478500	9.16	0.0
15	50	15	765500	2	747500	261.36	2.4
16	60	15	775500	3	771500	76.09	0.5
17	70	15	781500	5	777500	3002.47	0.5
18	80	15	1213000	5	1199000	5973	1.2
19	90	15	1255000	15	1245000	1683.45	0.8
20	100	15	1733000	11	1720000	9666.92	0.8
21	10	20	266000	<1	266000	0.41	0.0
22	20	20	432500	1	424500	4.19	1.9
23	30	20	502500	1	498500	4.94	0.8
24	40	20	785500	1	764500	3577.72	2.7
25	50	20	825500	3	818500	871.55	0.9
26	60	20	1197000	4	1189000	7263.31	0.7
27	70	20	1259000	4	1245000	11691.09	1.1
28	80	20	1225000	9	1204000	20953.42	1.7
29	90	20	1697000	30	(OM)	37332.08	n/a
30	100	20	1757000	14	1747000	6071.27	0.6
31	10	25	426500	<1	422500	1.64	0.9
32	20	25	532500	1	515500	3.2	3.3
33	30	25	542500	1	542500	10.02	0.0
34	40	25	839500	2	831500	109.77	1.0
35	50	25	915500	2	915500	381.58	0.0
36	60	25	891500	3	884500	382.69	0.8
37	70	25	1219000	5	(OM)	66172.7	n/a
38	80	25	1289000	9	1279000	25186.2	0.8
39	90	25	1787000	4	1773000	1451.67	0.8
40	100	25	(NS)	21	(NS)	132.66	n/a

(OM) : Out of Memory
(NS) : No solution found

Tableau A.3 Résultats du test 3

			TEST 3				
Problème	I	J	Heuristique $	Temps (s)	OPL* $	Temps (s)	Écart %
1	10	10	236000	<1	232000	0.19	1.7
2	20	10	222000	<1	212000	0.17	4.7
3	30	10	428500	1	407500	3.38	5.2
4	40	10	428500	1	417500	2.13	2.6
5	50	10	464500	2	457500	4.06	1.5
6	60	10	725500	3	710500	15.25	2.1
7	70	10	761500	4	757500	1163.91	0.5
8	80	10	1169000	5	(OM)	13338.44	n/a
9	90	10	1215000	14	1205000	13805.75	0.8
10	100	10	1211000	41	1201000	6134.38	0.8
11	10	15	266000	1	266000	0.22	0.0
12	20	15	422500	<1	405500	1.39	4.2
13	30	15	458500	1	444500	4.58	3.1
14	40	15	755500	2	730500	1406.44	3.4
15	50	15	765500	2	757500	226.64	1.1
16	60	15	805500	3	801500	547.06	0.5
17	70	15	827500	4	813500	3451.7	1.7
18	80	15	1189000	5	1178000	483.89	0.9
19	90	15	1209000	18	1205000	8870.61	0.3
20	100	15	1687000	11	1680000	15479.09	0.4
21	10	20	416500	1	412500	6.17	1.0
22	20	20	462500	1	448500	9.59	3.1
23	30	20	462500	1	458500	4.47	0.9
24	40	20	522500	2	522500	3.33	0.0
25	50	20	775500	2	764500	162.16	1.4
26	60	20	831500	4	824500	1380.91	0.8
27	70	20	865500	5	865500	58.77	0.0
28	80	20	1229000	10	1218000	6921.22	0.9
29	90	20	1289000	19	1289000	765.86	0.0
30	100	20	1733000	14	1720000	74.5	0.8
31	10	25	546500	1	542500	2.45	0.7
32	20	25	526500	1	522500	1.89	0.8
33	30	25	839500	1	821500	8.64	2.2
34	40	25	855500	2	841500	168.39	1.7
35	50	25	805500	3	791500	1293.41	1.8
36	60	25	1237000	3	1229000	8725.5	0.7
37	70	25	1253000	4	1249000	56746.67	0.3
38	80	25	1249000	8	1245000	7306.7	0.3
39	90	25	1751000	4	1747000	515.92	0.2
40	100	25	1813000	9	1813000	436.27	0.0

(OM) : Out of Memory

Tableau A.4 Résultats du test 4

Problème	I	J	TEST 4				
			Heuristique	Temps	OPL*	Temps	Écart
			$	(s)	$	(s)	%
1	10	10	186000	1	179000	0.41	3.9
2	20	10	382500	<1	358500	0.5	6.7
3	30	10	372500	1	358500	3.39	3.9
4	40	10	448500	1	434500	7.06	3.2
5	50	10	484500	2	484500	3.19	0.0
6	60	10	701500	3	683500	21.11	2.6
7	70	10	761500	5	757500	19.7	0.5
8	80	10	1169000	6	1155000	27690.33	1.2
9	90	10	1149000	20	1138000	2430.31	1.0
10	100	10	1201000	40	1194000	23371.94	0.6
11	10	15	246000	1	246000	0.36	0.0
12	20	15	392500	1	375500	6.86	4.5
13	30	15	402500	1	392500	1.03	2.5
14	40	15	472500	1	472500	2.11	0.0
15	50	15	745500	3	724500	3207.42	2.9
16	60	15	821500	3	814500	128.97	0.9
17	70	15	807500	5	797500	710.45	1.3
18	80	15	1199000	5	1185000	14713.75	1.2
19	90	15	1215000	14	1208000	31212.03	0.6
20	100	15	1723000	35	(OM)	7679.48	n/a
21	10	20	446500	1	442500	1.38	0.9
22	20	20	462500	1	448500	5.86	3.1
23	30	20	498500	1	491500	6.44	1.4
24	40	20	769500	1	757500	5.8	1.6
25	50	20	765500	2	757500	674.83	1.1
26	60	20	801500	4	784500	1132.8	2.2
27	70	20	1199000	4	1185000	14629.69	1.2
28	80	20	1219000	11	1198000	5537.72	1.8
29	90	20	1305000	18	1292000	712.75	1.0
30	100	20	(NS)	1	(NS)	0.14	n/a
31	10	25	456500	1	449500	1.36	1.6
32	20	25	542500	1	535500	3.39	1.3
33	30	25	799500	1	781500	2914.23	2.3
34	40	25	785500	2	767500	1561.14	2.3
35	50	25	869500	3	865500	350.58	0.5
36	60	25	885500	3	885500	107.5	0.0
37	70	25	1289000	4	1276000	5548.42	1.0
38	80	25	(NS)	8	1299000	6.08	n/a
39	90	25	1791000	4	1791000	277.14	0.0
40	100	25	1777000	7	1777000	13.44	0.0

(OM) : Out of Memory
(NS) : No solution Found

Tableau A.5 Résultats du test 5

Problème	I	J	TEST 5				
			Heuristique	Temps	OPL*	Temps	Écart
			$	(s)	$	(s)	%
1	10	10	176000	<1	169000	0.17	4.1
2	20	10	252000	1	252000	0.34	0.0
3	30	10	388500	1	365500	9.28	6.3
4	40	10	418500	2	407500	17.17	2.7
5	50	10	464500	2	464500	2	0.0
6	60	10	741500	3	723500	1459.3	2.5
7	70	10	771500	5	767500	176.63	0.5
8	80	10	827500	12	817500	141.7	1.2
9	90	10	1155000	20	1141000	2353.8	1.2
10	100	10	1191000	41	1187000	10878.05	0.3
11	10	15	226000	1	216000	0.13	4.6
12	20	15	392500	1	378500	1.33	3.7
13	30	15	408500	1	391500	3.66	4.3
14	40	15	504500	1	491500	13.31	2.6
15	50	15	745500	3	730500	330.72	2.1
16	60	15	801500	3	791500	1667.69	1.3
17	70	15	811500	4	801500	167.92	1.2
18	80	15	837500	15	837500	436.24	0.0
19	90	15	1219000	13	1215000	19856.19	0.3
20	100	15	1251000	42	1251000	1571.25	0.0
21	10	20	426500	1	419500	7.88	1.7
22	20	20	442500	1	431500	11.66	2.5
23	30	20	482500	1	478500	3.66	0.8
24	40	20	526500	1	518500	2.13	1.9
25	50	20	805500	2	790500	1358.88	1.9
26	60	20	805500	4	797500	102.95	1.0
27	70	20	1243000	4	1232000	9621.08	0.9
28	80	20	1249000	17	1239000	6505.45	0.8
29	90	20	1717000	5	(OM)	33724.7	n/a
30	100	20	(NS)	14	(NS)	1257.45	n/a
31	10	25	466500	1	462500	23.27	0.9
32	20	25	572500	1	562500	10.22	1.8
33	30	25	562500	1	562500	3.28	0.0
34	40	25	849500	2	838500	163.91	1.3
35	50	25	905500	3	905500	56.08	0.0
36	60	25	1223000	3	1205000	8237.28	1.5
37	70	25	1249000	5	1235000	14331.72	1.1
38	80	25	1289000	11	1279000	26681.59	0.8
39	90	25	1797000	5	1787000	10871.28	0.6
40	100	25	(NS)	<1	(NS)	0.56	n/a

(OM) : Out of Memory
(NS) : No solution Found