Titre: Title:	Modélisation des commutateurs tandems dans un logiciel de planification de réseaux de télécommunications
Auteur: Author:	Christian Foisy
Date:	1990
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Foisy, C. (1990). Modélisation des commutateurs tandems dans un logiciel de planification de réseaux de télécommunications [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/59268/

Université de Montréal

Modélisation des commutateurs tandems dans un logiciel de planification de réseaux de télécommunications

par

Christian Foisy
Département de Génie Électrique et de Génie Informatique
École Polytechnique de Montréal

Mémoire présenté en vue de l'obtention
du grade de Maître ès Sciences Appliquées (M.Sc.A.)

Décembre 1990

ISBN 0-315-77651-X

Université de Montréal

École Polytechnique

Ce mémoire intitulé:

Modélisation des commutateurs tandems dans un logiciel de planification de réseaux de télécommunications

présenté par: Christian Foisy

en vue de l'obtention du grade de: M.Sc.A..

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

M. Dinkar MUKHEDKAR, D.Sc., président

M. Pierre BLONDEAU, Ph.D.

M. Jean-Louis HOULE, Ph.D.

Sommaire

La planification des réseaux de télécommunications est une activité complexe qui, au fil des années, a été partiellement automatisée à l'aide d'outils logiciels. Ces outils, qui ont graduellement remplacé les anciennes méthodes manuelles, font usage d'algorithmes spéciaux soit pour la recherche de la charge maximale que peut absorber un réseau, soit pour l'optimisation d'un ensemble de ressources atteignant un objectif de performance donné. Le logiciel pour lequel la nouvelle méthodologie a été développée s'appelle SNAP (Switch Network Analysis Program). Ce logiciel a été conçu pour la planification des réseaux locaux publics de télécommunications. Dans sa version actuelle, ce logiciel ne peut que modéliser un sous-ensemble des commutateurs du réseau local, soit les commutateurs de Classe 5. Si le logiciel modélise bien les commutateurs de Classe 5, il est impossible de modéliser les commutateurs situés juste après dans la hiérarchie: les commutateurs tandems. On devait donc adapter les algorithmes à ce nouveau problème. Il a fallu tout d'abord analyser les algorithmes utilisés. Ainsi, on devait trouver les conditions de convergence de ces algorithmes afin de les respecter dans la méthodologie projetée. De plus, le problème de modélisation dans la base de données du logiciel devait être aussi abordé. Les résultats ont montré qu'il est effectivement possible de modéliser les tandems, si l'on accepte certaines contraintes pour la précision des résultats. Ces contraintes sont jugées raisonnables dans le contexte de la planification.

Abstract

Planification of telecommunication networks is a complex task that has been over the years partially automated, with the advent of software tools. These tools gradually replaced manual methods by using special algorithms either for the research of the maximal load of a given network or the optimization of resources for a given performance objective. SNAP, which stands for *Switch Network Analysis Program*, is such a program. SNAP was designed specifically for the planification of local public telecommunication networks. If Class 5 switches are handled correctly, the planner is unable with SNAP to tackle the modelization of the switches located one step higher in hierarchy, namely the tandem switch. This thesis presents a new methodology, compatible with those already in SNAP, to solve this problem. Firstly, a thorough analysis of the methodology in place and conditions of convergence was done. Those conditions of convergence must of course be satisfied by the proposed methodology. Moreover, the problem of the modelization of tandem switches in SNAP database has also to be solved. Results obtained with a working program with the new methodology showed that is possible to do so, if planners accept some deviations from optimal solution, deviations judged to be acceptable in a planification context.

Remerciements

J'aimerais ici remercier les gens qui ont contribué de près ou de loin au contenu de ce mémoire. Je voudrais tout d'abord remercier Pierre Blondeau qui a bien voulu me suivre dans les dédales (administratifs et mathématiques) de ce projet. Je remercie aussi Recherches Bell-Northern ainsi que les gens qui ont gentiment répondu aux questions du néophyte que j'étais, dont entre autres Ken Rosenfeld (mon "parrain"), Gilbert Benoît (philosophie SNAP), Marc Chatel (CMS et Datapac), René-Alain Hébert (XENIX, Informix, SNAP), Lionel Moser (RCMB), Michel Boyer (planification), Alain Caron (Compressor), Sergei Michailov et Tadeusz Drwiega (pour des documents "hautement" confidentiels), et enfin Francine Vermette (pour mes incalculables téléphones et autres services). Je profite aussi de l'occasion pour remercier de façon particulière Doug Lowther et Gilles Lapierre sans qui ce projet n'aurait sûrement jamais vu le jour.

Le travail de ce mémoire a été en partie subventionné par une bourse du Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG). Enfin, je terminerai en remerciant les gens qui m'ont aidé à relever d'un cran (ou deux) le français et l'orthographe de ce mémoire, Lorraine Caron et Martine Foisy.

Table des matières

Liste des tableaux

Liste des figures

Liste des figures (suite)

Liste des symboles

$\{a, ..., b\}$	ensemble discret
$[a, b]$	intervalle fermé sur l'axe des réels
B	ensemble binaire $\{0, 1\}$
B^n	ensemble des vecteurs de dimension n à variables binaires
R	ensemble des réels
R^n	ensemble des vecteurs de dimension n à variables réelles
Z	ensemble des entiers
Z_+	ensemble des entiers positifs, 0 y compris
$\bar{x}$	complément de $x \in B, \bar{x} = 1 - x$
$\lceil x$	plafond de x, c'est-à-dire l'entier n le plus petit pour lequel $n \geq x$
$\|S\|$	cardinalité de l'ensemble S

Liste des acronymes et sigles

100AmpH:	Système d'alimentation
AHP:	Appels à l'Heure de Pointe
BIP:	Binary Integer Program
BUSNS:	Service commercial
CCS:	Centaine de Communications par Seconde
CD:	Commutateur Distant
CENTRX:	Service Centrex
CMA:	Charge Maximale Admissible
CMS:	Conversational Monitoring System
CP:	Commutateur(s) Principal(aux)
CPM:	Critical Path Method
CPU:	Central Processing Unit
DCR:	Dynamically Controlled Routing
DF:	Direct Final
DMS[1]:	Digital Mutiplexing System
DRL:	Dimensionneur de R.éseaux de Liaisons (algorithme)
DS:	Digital Stream
DS30A:	Digital Stream de haute capacité (interne)
DSN:	Data for Switch Network
DTC:	Digital Trunk Controller
EAS:	Extended Area of Service
FHR:	Fixed Hierarchical Routing
GERM:	General Entity Relationship Model
HU:	High Usage
IBM:	International Buisness Machines
IHU:	Intermediary High Usage
Kb/s:	kilo bits par seconde
LCM:	Line Concentrating Module
LGC:	Line Group Controller
LTC:	Large Trunk Controller
MG:	Modèle Gravitationnel
NM:	Network Module
NRB:	Nombre de Recherches Binaires
PC:	Point de Congestion
POTS:	Plain Ordinary Telephone Service
RBN:	Recherches Bell-Northern
RCMB:	Resource and Cost ModelBase
RLCM[1]:	Remote Line Concentrating Module
RSC[1]:	Remote Switching Center
SNA:	Standard Network Architecture
SNAP:	Switch Network Analysis Program
SXS:	Step-by-Step
VF:	Voice Frequency
VM:	Virtual Machine
XBAR:	Crossbar

1. Marque déposée de Northern Telecom Ltée.

Liste des appendices

Introduction

Il n'y pas si longtemps, la planification des réseaux de télécommunications se faisait encore de façon manuelle. Avec l'avènement des ordinateurs, les anciennes méthodes manuelles ont été graduellement automatisées. Plusieurs années de recherche dans le domaine de la planification à long terme des réseaux ont abouti à l'élaboration d'algorithmes couvrant l'ensemble des ressources que l'on retrouve dans ceux-ci. Que ce soit pour l'optimisation du réseau d'accès, le calcul de la charge maximale admissible pour les commutateurs et les réseaux de transmission optimaux, ces algorithmes sont à la base de plusieurs logiciels qui permettent aux planificateurs de passer moins de temps à exécuter des tâches fastidieuses ou impossible à solutionner dans un temps raisonnable. Ces outils permettent de considérer plus de scénarios dans le même laps de temps, d'où l'amélioration de la productivité.

Le problème qui fait l'objet de ce document consiste à résoudre, à partir d'un logiciel existant, un problème de modélisation qui ne pouvait être abordé avec les algorithmes en place. Ce logiciel est utilisé pour la planification à long terme des ressources de commutation dans les réseaux publics locaux de télécommunications. Dans sa forme actuelle, le logiciel ne permet que la modélisation des commutateurs de Classe 5. Le problème consistait à modéliser les commutateurs tandems, c'est-à-dire les commutateurs hiérarchiquement supérieurs aux commutateurs de Classe 5.

Ce mémoire est divisé en cinq chapitres. Le premier chapitre décrit tous les éléments de la problématique. Le second chapitre contient la méthodologie développée. Le troisième chapitre donne de l'information supplémentaire concernant la méthodologie; cette information est complémentaire et non essentielle à la compréhension de la méthodologie du deuxième chapitre. Le quatrième chapitre contient l'ensemble des résultats obtenus. Enfin, le cinquième et dernier chapitre contient la discussion ainsi que la conclusion de ce mémoire.

1. Problématique

La modélisation des commutateurs tandems se situe dans le contexte général de la planification des réseaux de télécommunications. De plus, ce problème est intimement lié aux algorithmes relatifs à la planification des réseaux. Le présent chapitre débute donc par la mise en situation du problème dans son cadre général. La seconde section contient l'ensemble des définitions ainsi que la description des algorithmes et des modèles pertinents au problème. La troisième section décrit spécifiquement le problème et présente les étapes de solution. Enfin, la quatrième et dernière section conclut ce chapitre avec une liste des travaux antérieurs.

1.1. Situation

Les sous-sections 1.1.1 et 1.1.2 décrivent le contexte général dans lequel s'inscrit le problème à résoudre. La première décrit globalement le processus de la planification à long terme des réseaux de télécommunications, alors que la deuxième sous-section explique comment ce processus est partiellement automatisé par logiciel. Cette deuxième partie introduit le logiciel d'aide à la planification pour lequel la nouvelle modélisation a été développée.

1.1.1 La planification des réseaux de télécommunications

La planification à long terme des réseaux publics de télécommunications est une activité importante au même titre que la maintenance et la gestion des opérations. Le planificateur d'une compagnie offrant ce type de service se doit d'assurer un développement continu du réseau, ceci afin de satisfaire la demande prévue. Le processus de planification peut être scindé en quatre phases distinctes, soit:

1. *L'estimation de la future demande dans le réseau.* Cette demande est fonction de plusieurs paramètres. Parmis les plus importants on retrouve les données de trafic recueillies dans le réseau existant, la croissance démographique, le contexte économique, les nouveaux services qui seront offerts aux abonnés et les résultats d'études de marché pour ces services.

2. *L'élaboration du scénario de développement.* Le scénario comprend l'ensemble des décisions qui déterminent le choix des éléments suivants: l'emplacement des commutateurs, le type de commutateur à utiliser, l'architecture du réseau, le type d'équipement à installer à l'intérieur d'un commutateur pour un type donné d'abonné. Bref, le scénario décrit l'ensemble des décisions dans l'installation de futurs équipements dans le réseau. Le planificateur, qui élabore un scénario, doit aussi tenir compte des politiques établies par la compagnie. La modernisation du réseau en est un exemple.

3. *Le calcul de la quantité d'équipements requis.* À partir de ce calcul, on détermine les coûts d'équipement, les coûts d'exploitation, les revenus générés par les abonnés que le scénario a permis de desservir ainsi que tout autre indicateur économique pertinent.

4. *Comparaison des différents scénarios envisagés.* On compare les différents scénarios sur la base des informations produites à la phase 3. D'autres facteurs comme la robustesse du réseau, la facilité avec laquelle on peut adapter le réseau si la demande réelle dévie des prévisions, les types de services que l'ont peut offrir à l'usager ainsi que d'autres critères sont aussi considérés.

Une fois la phase 1 complétée pour un sous-réseau donné, les prévisions de demande restent les mêmes pour tous les scénarios (si l'on veut comparer de façon valable différents scénarios). Les phases 2 et 4 requièrent l'expérience et la créativité du planificateur; c'est le "coeur" du travail de planification. Lorsque les procédures d'approvisionnement de l'équipement et le calcul des coûts et des indicateurs économiques sont déterminées, la phase 3 s'avère être la partie fastidieuse et répétitive du processus. C'est principalement pour automatiser cette étape qu'un logiciel d'aide à la planification a été conçu.

1.1.2 Automatisation de la planification

Cette troisième phase est maintenant automatisée grâce à un logiciel appelé SNAP, pour *Switch Network Analysis Program*. SNAP est un logiciel d'aide à la planification des réseaux locaux publics de télécommunications. Il a été conçu par le groupe Développement des Outils de Planification de Recherches Bell-Northern (RBN), au laboratoire de Montréal. Du point de vue de l'information d'entrée/sortie, on peut considérer le logiciel SNAP comme une boîte noire. La Figure 1, avec les flèches, montre le flot de données. Voici le fonctionnement général de ce logiciel.

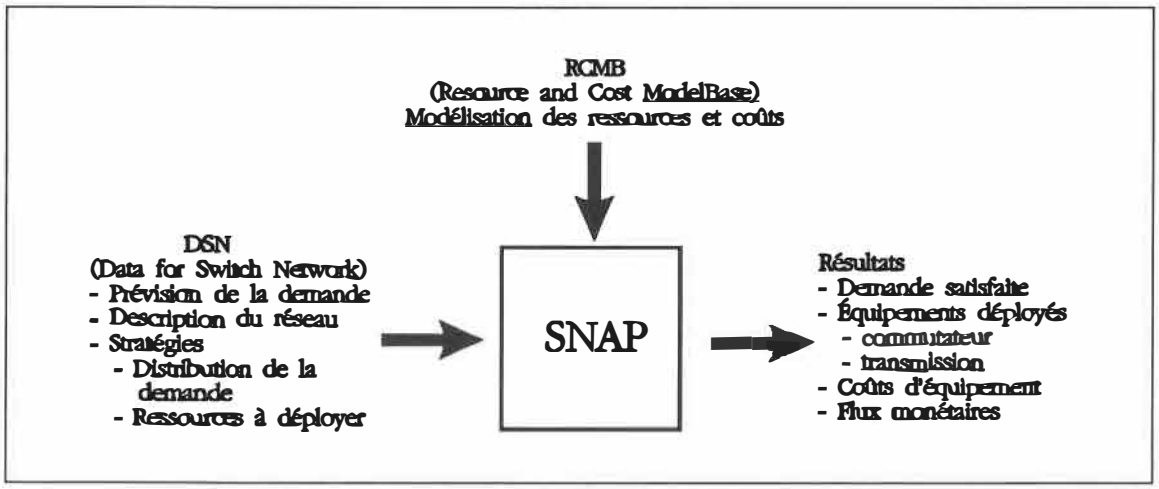


Figure 1. **SNAP: schéma entrée/sortie**

L'information d'entrée est contenue dans deux bases de données: la base DSN (Data for Switch Network) et la base RCMB (Resource and Cost ModelBase). La base DSN contient l'information spécifique au réseau étudié: les prévisions annuelles de demande couvrant la période de l'étude, les stratégies pour la distribution de la demande et les stratégies pour l'installation des ressources. Les stratégies seront décrites plus loin dans cette section. La base RCMB modélise les relations logiques entre les ressources et les flux monétaires (unit cash flow) associés à celles-ci. La base RCMB comprend aussi les procédures de calcul des quantités d'équipements à déployer. Bref, la base DSN contient l'information des phases 1 et 2, c'est-à-dire l'information spécifique au sous-réseau étudié. On appelle scénario l'information contenue dans la base DSN. La base RCMB décrit la modélisation des ressources (phase 3) commune à tous les scénarios, donc l'information indépendante. À partir de l'information d'entrée (DSN + RCMB), le logiciel SNAP produit les rapports suivants: la demande satisfaite, les équipements requis, les revenus générés et les flux monétaires (entre autres: dépréciation, valeur nette). Ces rapports sont produits pour chacune des années de la période couverte par l'étude.

Le logiciel SNAP incorpore deux mécanismes importants, ce sont l'algorithme d'approvisionnement et les stratégies. L'algorithme d'approvisionnement est utilisé dans SNAP pour installer les ressources (équipements) nécessaires dans le réseau simulé. La base RCMB contient des points de congestion qui permettent au logiciel, à l'aide de l'algorithme d'approvisionnement, de trouver la charge maximale admissible (CMA) pour un commutateur. L'algorithme d'approvisionnement se termine lorsque la CMA trouvée respecte la tolérance demandée par le planificateur. Les rapports produits par SNAP garantissent que de nouvelles ressources ne sont introduites qu'au moment nécesssaire et qu'aucun commutateur n'est saturé par la prévision de charge calculée pour celui-ci.

L'algorithme d'approvisionnement et les points de congestion forment un mécanisme rétroactif qui sera décrit en détail à la section 1.2.3 (p. 23). En plus des points de congestion, le planificateur contrôle le déploiement des ressources dans le réseau à l'aide des stratégies.

Il existe deux types de stratégies: celles pour les commutateurs et celles pour les ressources. Les prévisions de demande, dans SNAP, sont données par immeuble. Les stratégies pour les commutateurs spécifient la séquence utilisée par l'algorithme d'approvisionnement pour distribuer la charge parmis les commutateurs d'un immeuble (si un commutateur congestionne, le prochain indiqué par la séquence est utilisé). Les stratégies pour les ressources permettent de choisir entre différents types d'équipements possibles pour satisfaire un type donné d'abonné.

Il est important de mentionner que SNAP est un outil de planification et non d'approvisionnement (même si l'algorithme de SNAP est appelé ainsi). Un outil d'approvisionnement sert à calculer précisément la quantité de ressources à installer dans le réseau actuel. Ce calcul dépend des données récoltées dans le réseau et des fluctations prévues à court terme. Un outil de planification à long terme donne une prévision des ressources nécessaires pour atteindre les objectifs de performance prévus. L'outil de planification doit fournir au planificateur des résultats lui permettant de comparer deux scénarios de façon fiable. Le but de tout outil de planification est *de permettre la comparaison valable entre différents scénarios*. Ce critère implique que deux scénarios similaires traités par SNAP doivent résulter en deux réseaux "voisins". Mathématiquement parlant, SNAP doit se comporter comme une fonction continue de son entrée. Une description plus poussée de la base RCMB et de l'algorithme d'approvisionnement est donnée dans les prochaines sous-sections.

Pour conclure la section, voici l'architecture du système (illustrée à la Figure 2) retenue pour l'opération du logiciel SNAP. L'architecture est composé deux éléments: l'environnement et le noyau. Voici la description de chacun des éléments.

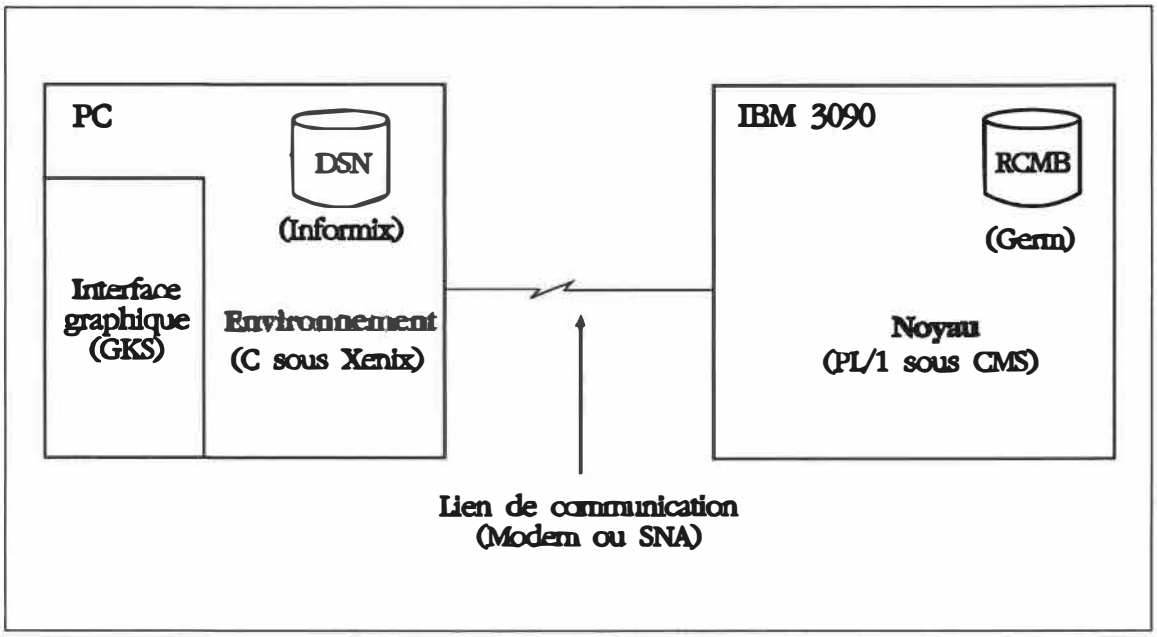


Figure 2. **SNAP: architecture du système**

L'**environnement** est composé de l'ensemble des programmes qui permettent au planificateur d'interagir avec SNAP. Les programmes de l'environnement gèrent les interactions suivantes: la gestion des menus, le questionnement des données d'entrée/sortie, la visualisation des données d'entrée/sortie, la manipulation des données d'entrée et la validation de celles-ci. L'ensemble des programmes qui forment l'environnement sont exécutés par une station de travail de type PC AT. Le PC est relié par un lien de communication (par modem ou lien SNA-3270) à un ordinateur IBM 3090. L'ensemble des programmes exécutés par l'IBM 3090 forment le noyau.

Le **noyau** est la partie du système qui, à partir du scénario associé à un réseau, produit les rapports suivants: la demande satisfaite, les ressources déployées, les coûts, les revenus et les indicateurs économiques. Le flot de données entre les éléments du système est le suivant:

- le planificateur entre les données de l'étude dans le PC,
- lorsque la fonction **évaluer** est invoquée, une copie du scénario est envoyée à l'IBM 3090 qui exécute le noyau,
- quand l'exécution du noyau est complétée, les résultats sont retournés au PC où le planificateur peut les consulter.

Dans le reste du mémoire, le mot SNAP fait référence aux programmes qui forment le noyau ou à l'algorithme d'approvisionement et sont interaction avec la base RCMB. Le problème de modélisation se situe essentiellement à cet endroit.

1.2. Définitions et descriptions

La modélisation des tandems recoupe plusieurs élements. Cette section comprend la définition des termes utilisés et la description des éléments touchés ou qui doivent être modifiés. Les réseaux de télécommunications publics sont divisés en deux sous-réseaux: le réseau local et le réseau interurbain. C'est le réseau local qui retiendra notre attention puisque c'est celui qui est modélisé avec la version actuelle de SNAP. La description du réseau local sera suivie de celle de la base de données RCMB. Ensuite, le fonctionnement de l'algorithme d'approvisionnement et son interaction avec la base RCMB est donné. Finalement, une courte description des algorithmes utilisés pour le dimensionnement des réseaux de liaisons concluera cette section.

1.2.1 Le réseau local

Le réseau local est le sous-réseau responsable de la commutation des appels non tarifiés. Cette division, en réseau local et réseau interurbain, n'est pas simplement logique: ces deux réseaux sont composés d'équipements différents et sont habituellement administrés, gérés et planifiés par des groupes différents. Dans la hiérarchie de commutation utilisée en Amérique du Nord, le réseau local est composé de tous les commutateurs de Classe 5 et des tandems. La Figure 3 schématise le réseau local avec les principaux composants que l'on y retrouve. Voici la description de ces composants.

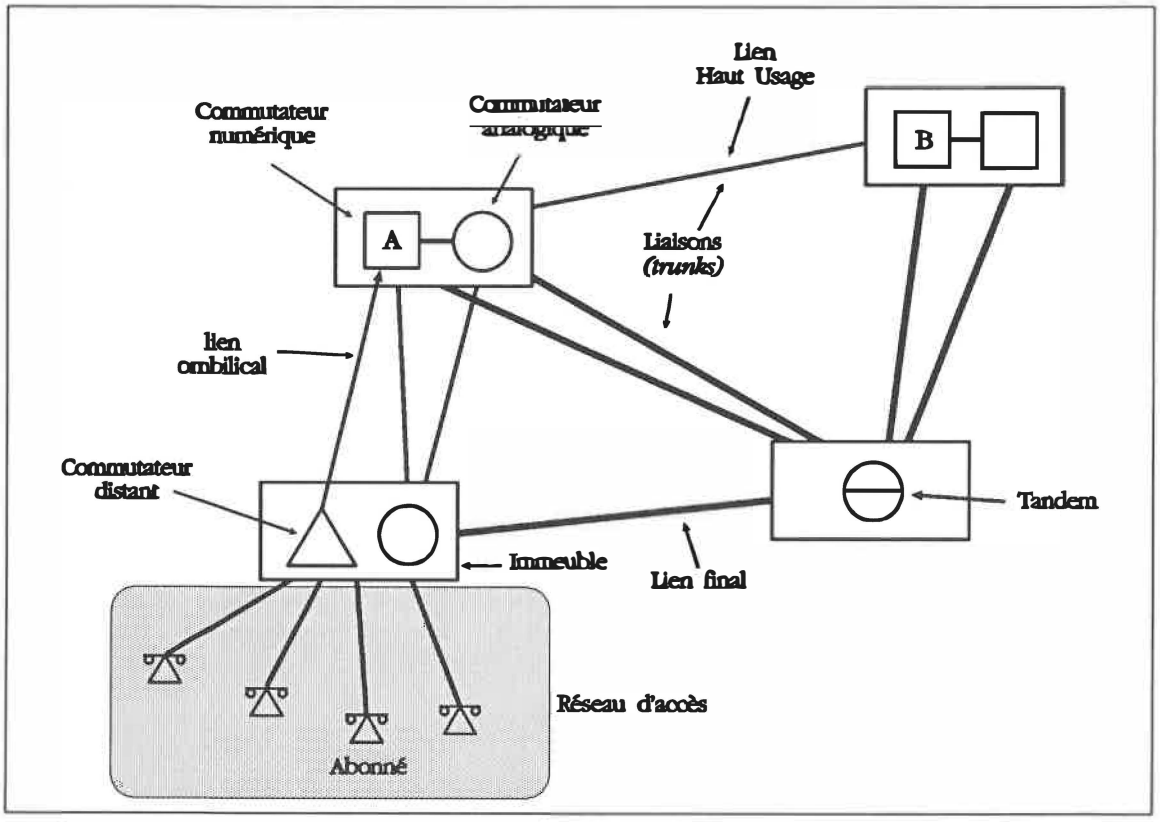


Figure 3. Le réseau local

Le **réseau d'accès** englobe tout l'équipement partant de l'usager jusqu'au commutateur auquel il est rattaché, c'est-à-dire les câbles, multiplexeurs, conduits et potaux. On ne peut modéliser le réseau d'accès avec SNAP; seule la charge présente dans ce dernier est utilisée. Le réseau d'accès desservi par une compagnie de téléphone est partitionné géographiquement en une série de sous-territoires appelés **échanges**. L'ensemble des échanges avec qui l'abonné d'un échange donné peut rejoindre, sans être tarifié, forme l'aire étendue de service (*Extended Area of Service*, EAS) de cet échange. Toutes les terminaisons du réseau d'accès aboutissent à un immeuble du réseau.

Un **immeuble** désigne le bâtiment ou voûte abritant le ou les commutateurs, les systèmes de transmission, les terminaisons du réseau d'accès, les systèmes de climatisation et d'alimentation d'urgence. Un immeuble peut abriter plus d'un commutateur (surtout dans les centres densemment peuplés).

Le **commutateur** désigne le système électro-mécanique ou électronique qui connecte le réseau d'accès au réseau des liaisons (décrit plus loin). La fonction du commutateur consiste à établir et maintenir un canal de communication entre deux abonnés. Si l'abonné demandé est rattaché au même commutateur, la communication est dite de type intracommutateur. Dans le cas où l'abonné demandé n'est pas rattaché au même commutateur, on parle de communication intercommutateur. Les communications de type intercommutateur requièrent le réseau de liaisons. Les plus vieux commutateurs sont de type analogique, comme les commutateur *pas-à-pas*[1] ou les commutateurs *à barres croisées*[2]. Les plus récents sont de type numérique, par exemple le DMS-100 et le DMS-1Urbain de Northern Telecom.

1. *step-by-step*, SXS dans le reste du mémoire.
2. *crossbar*, XBAR dans le reste du mémoire.

Il existe un autre type de commutateur: le **commutateur distant** (*remote switch*). Le commutateur distant dépend d'un autre commutateur, appelé commutateur récepteur, pour accéder au réseau des liaisons. Certains peuvent commuter leur trafic intracommutateur, s'ils sont équipés en conséquence. Pour ceux ne disposant pas de cet équipement, une communication intracommutateur doit premièrement être acheminée au commutateur recevant le lien ombilical, pour être commuté. Une fois commutée, une autre liaison est requise pour le retour. Le commutateur distant est relié au commutateur récepteur par un **lien ombilical**. Le lien ombilical est composé d'équipements similaires aux liaisons mais n'est pas considéré par SNAP comme faisant partie du réseau de liaisons.

Le dernier type de commutateur que l'on retrouve dans le réseau local est le **tandem**. Le tandem se distingue des autres commutateurs du fait qu'il effectue des connections de type liaison à liaison. Les principales fonctions du tandem dans le réseau local sont:

- la multiplication les routes possibles entre deux commutateurs, d'où l'augmentation de la robustesse du réseau,
- la diminution du nombre de liaisons en concentrant le trafic[1] qui ne justifie pas un lien direct entre deux commutateurs et le trafic de débordement (définition à la p. 12),
- la formation d'un "squelette" dans le réseau. Tous les commutateurs du réseau, à l'aide des tandems et des liens finaux, peuvent communiquer avec n'importe quel autre commutateur du réseau.

Les tandems qui désservent en plus des abonnés sont appelés tandems-locaux.

1. Le trafic moyen par liaison augmente avec la charge.

Les **liaisons** (*trunks*) permettent la connection, directe ou indirecte (via un tandem) entre tous les commutateurs du réseau. Une liaison est essentiellement un canal de communication, entre deux commutateurs, qui peut être utilisée lors de l'établissement d'une communication, si elle n'est pas déjà occupée. Les liaisons peuvent être de technologie analogique (VF pour *Voice Frequency*) ou numérique (de type T1 avec fils de cuivre ou fibre optique). Lorsqu'une communication intercommutateur est demandée, le commutateur d'où origine l'appel accède au réseau de liaisons selon un ordre préétabli, dicté par des tables appelées **table de routage**. Voici les deux principaux types de fonction des liaisons.

Le **lien Haut Usage** (HU) est un groupe de liaisons entre deux commutateurs dont le trafic de débordement est redirigé vers un autre lien. Le **trafic de débordement** provient des appels dirigés vers le lien et qui y sont bloqués; l'établissement d'un appel est bloqué si le lien ne contient aucune liaison libre. Un lien de ce type qui ne reçoit aucun trafic de débordement est dit de type Primaire à Haut Usage (PHU), sinon il est dit de type Intermédiaire à Haut Usage (IHU).

Le **lien final** est un groupe de liaisons, entre un commutateur et un tandem, pour lequel le trafic de débordement n'est pas réacheminé vers un autre lien. Le lien final est le dernier lien possible pour l'acheminement des communications intercommutateurs. Un appel bloqué au lien final est signalé à l'usager par une tonalité modulée plus rapidement que lorsque le numéro demandé est déjà en communication. Chaque commutateur possède un tel lien avec un tandem.

L'activité du commutateur consistant à trouver une séquence de liaisons libres lors de l'établissement d'une communication est appelée **routage**. Par exemple, si un appel origine au commutateur A (Figure 3, p. 9) et doit rejoindre un abonné du commutateur B, le commutateur A vérifie s'il reste des liaisons (des canaux) libres dans le lien PHU les réunissant. Si toutes les liaisons sont utilisées, l'appel est dirigé vers le lien final. L'appel est bloqué si toute les liaisons de ce lien sont occupées ou si le tandem ne dispose d'aucune liaison libre avec B.

Deux paramètres sont utilisés dans la mesure de l'amplitude du trafic, le CCS (Centaine de Communication par Seconde) et les AHP (Appels à l'Heure de Pointe). Le **CCS** sert à mesurer l'intensité du trafic. 36 CCS équivalent 1 Erlang. 1 Erlang correspond à l'occupation maximale d'un canal de communication. Par exemple, si un canal est utilisé pendant une demi-heure sur une période d'une heure, le trafic généré est de 0.5 Erlang. Toutes les mesures de trafic dans le mémoire sont données en CCS. C'est l'unité la plus importante ici puisque le nombre de liaisons entre les commutateurs est fonction de l'intensité du trafic, et non pas du taux d'appel. Les **AHP** est l'unité de mesure du taux d'appels. Les réseaux de télécommunications sont dimensionnés pour une période bien définie appelée heure de pointe.

L'**heure de pointe** est l'heure pendant laquelle le trafic est le plus élevé. L'heure de pointe ne comprend pas le trafic généré pendant les périodes où le trafic est anormalement élevé, par exemple lors de la fête des mères ou lors d'une tempête de neige. On dimensionne les réseaux de façon à ce que le blocage perçu par l'usager ne dépasse pas une certaine limite, limite déterminée par la compagnie opérant le réseau et les organismes de réglementation. Pour terminer, la capacité des canaux de l'équipement numérique de télécommunications est classée selon une hiérarchie appelée hiérarchie numérique. Les 4 premiers niveaux de la hiéarchie utilisée en Amérique du Nord sont les suivants (1 VF = *Voice Frequency* = 1 canal de voix numérisé):

Signal	Vitesse de transmission	Capacité
DS0	64 Kb/s	1 VF
DS1	1.544 Mb/s	24 VF
DS2	6.312 Mb/s	4 DS1; 96 VF
DS3	44.736 Mb/s	7 DS2; 28 DS1; 672 VF

1.2.2 La base de données RCMB

La base de données RCMB (Resource and Cost ModelBase) est composée de plusieurs fichiers bidimensionnels classiques, c'est-à-dire des fichiers formés de champs et d'entrées. C'est la base RCMB qui contient l'ensemble des relations qui existent entre les différentes ressources et les flux monétaires modélisés. La première étape, dans l'élaboration de la base RCMB, consiste à représenter sous forme d'un graphe orienté les implications logiques entre les ressources. Cette première étape "graphique" de modélisation est régie par des règles précises et possède sa propre terminologie, terminologie en partie empruntée aux graphes orientés. La description des règles de modélisation sort du cadre de ce mémoire et n'est pas nécessaire à la compréhension du problème. Le reste de cette sous-section expose donc les éléments essentiels. Des analogies, techniquement incorrectes ou incomplètes, sont utilisées pour clarifier des éléments autrement trop complexes.

La Figure 4, à la page 15, donne un exemple de représentation graphique de la base RCMB; ici c'est la modélisation d'un commutateur numérique DMS-100. Il est conseillé ici de se familiariser avec la terminologie employée (toujours à la Figure 4) dans la description de la base RCMB avant de continuer.

Par définition, on appelle *père* le noeud d'où émane un arc orienté et *fils* le noeud pointé. Avant de définir les différentes classes de noeuds qui composent la base RCMB, les deux points suivants méritent d'être précisés.

Premièrement, les arcs indiquent une relation *logique* et non pas physique. Par exemple, si des câbles sont requis entre un équipement A et un équipement B, la modélisation correcte est celle illustrée à la Figure 5 (b) de la page 16, non celle illustrée en (a). Les arcs indiquent une relation causale entre le père et le fils. Cette forme d'implication est identique à celle des diagrammes CPM pour la gestion de projet. Par exemple, à la Figure 5 (a), l'arc entre les deux noeuds peut être interprété comme le besoin de compléter l'activité du calcul de la quantité d'équipement A avant de pouvoir compléter l'activité du calcul de la quantité d'équipement B.

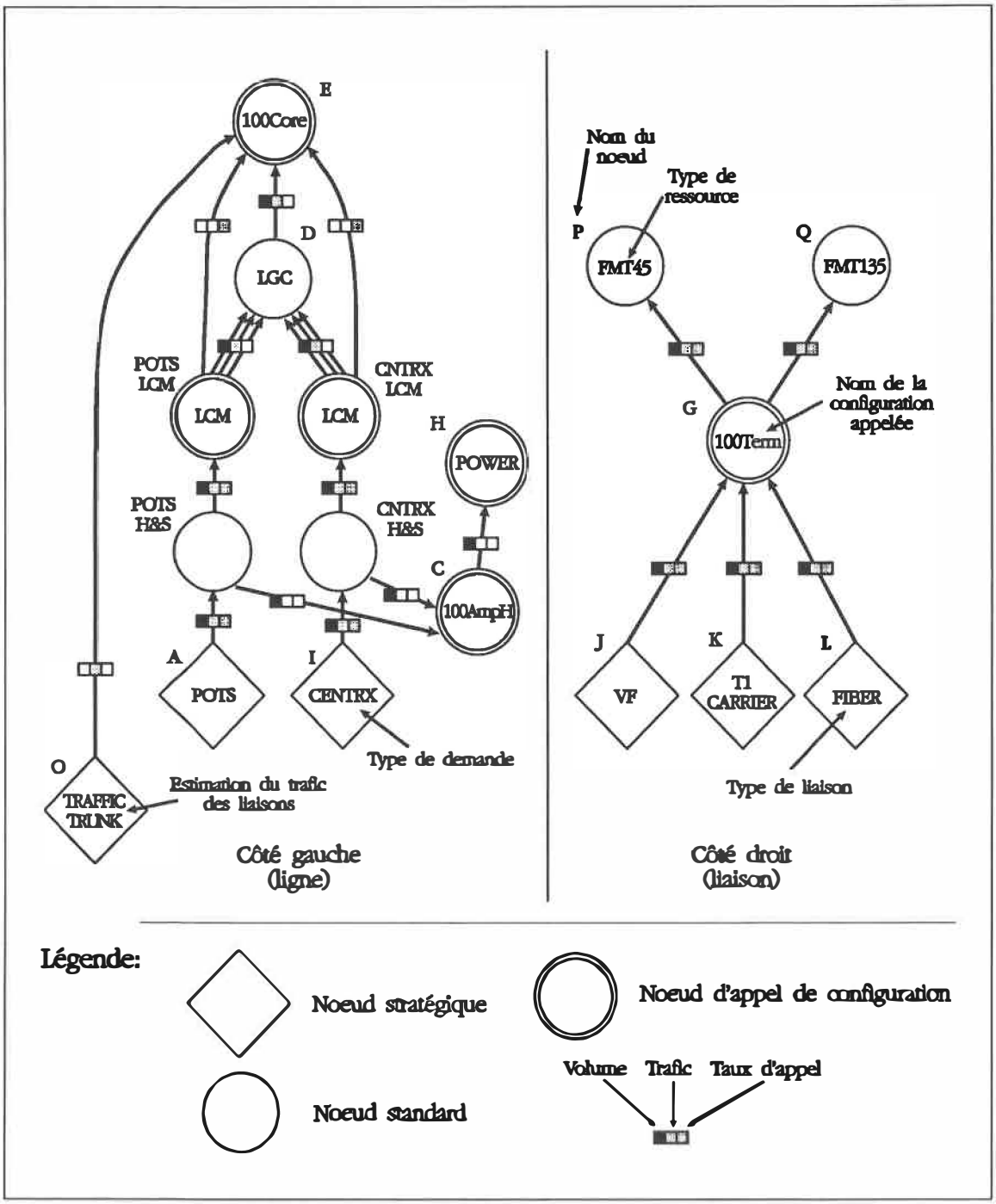


Figure 4. **Représentation graphique de la base RCMB**

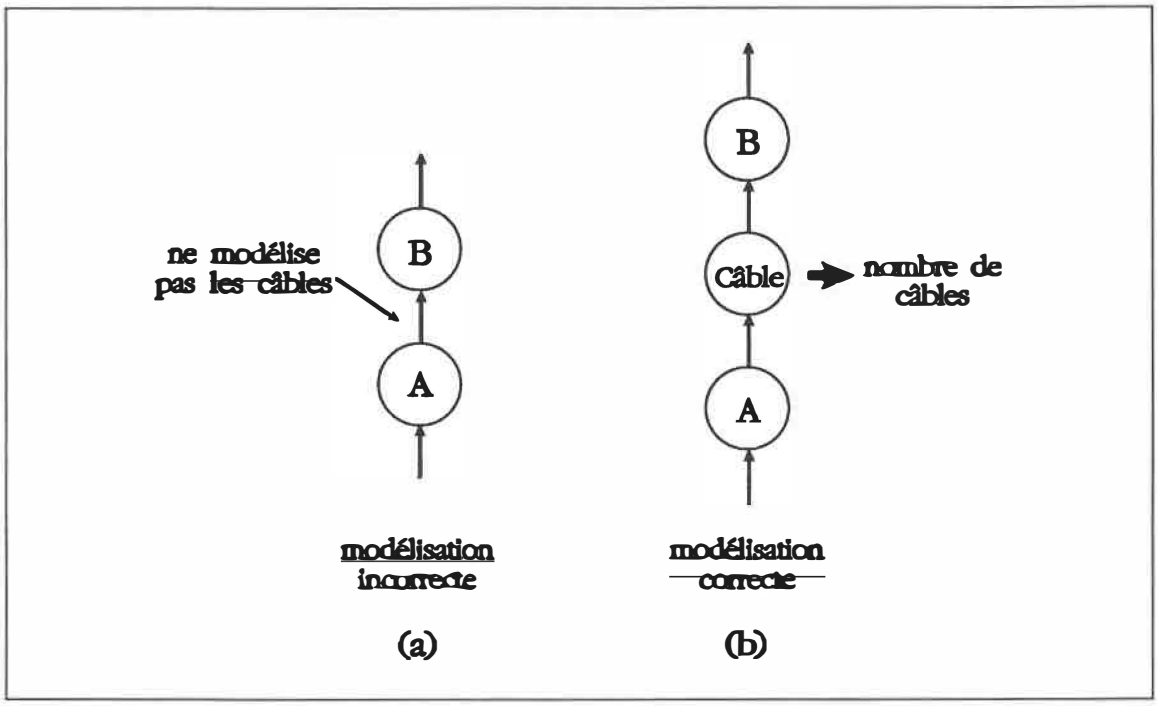


Figure 5. **Implication logique des arcs**

Deuxièmement, à chaque noeud est rattaché une quantité par commutateur et un numéro de type de ressource. De cette façon, SNAP n'a qu'à sommer par numéro de ressource, pour un commutateur donné, afin de produire le rapport des ressources à déployer du commutateur. Afin de déterminer l'équipement à déployer dans le réseau, selon le type de commutateur, la base RCMB contient des points de changement de commutateur. La Figure 6 illustre comment SNAP résout les points de changement de commutateur en les combinant avec la topologie du réseau. À l'aide de ce mécanisme, SNAP ne "copie"[1] que les parties pertinentes de la base RCMB. Voici maintenant la description des quatre classes de noeuds qui composent la base RCMB.

1. Techniquement, le processus d'interprétation de la base RCMB par SNAP est beaucoup plus complexe. L'analogie de la copie ci-haut est tout à fait valable pour le niveau de détail nécessaire à la compréhension de ce qui suit.

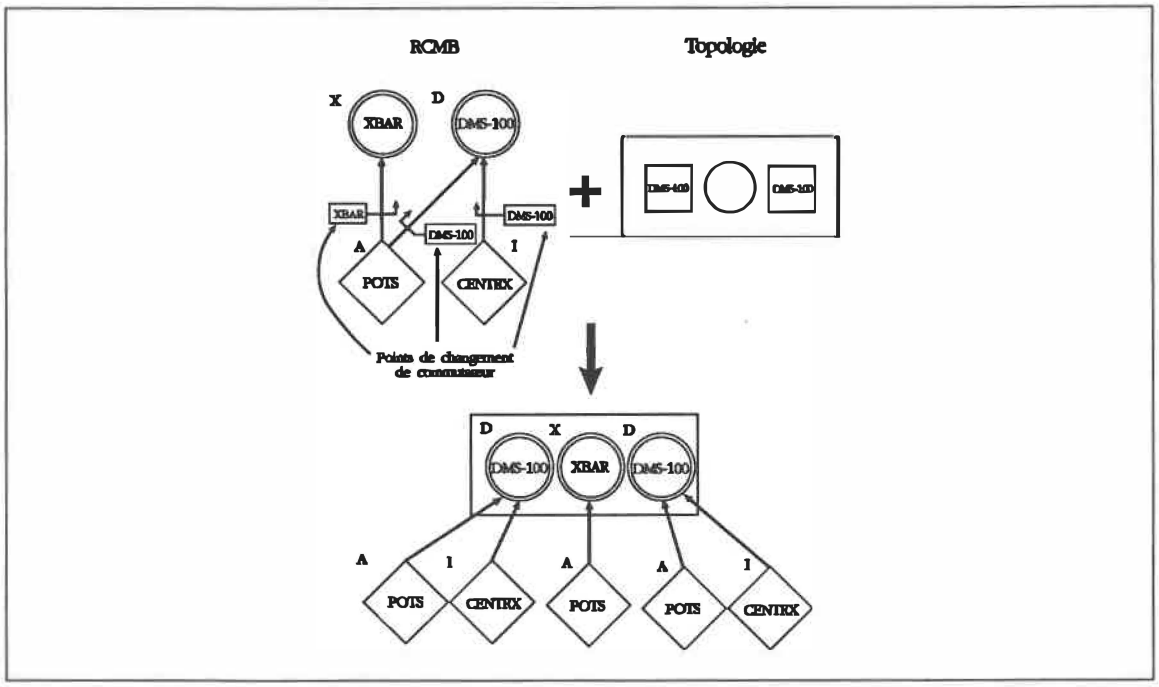


Figure 6. Points de changement de commutateur

Les **noeuds principaux d'entrée** constituent la première classe. Il y a six de ces noeuds à la Figure 4. Tous les noeuds en forme de losange, situés au bas de la figure, sont des noeuds principaux d'entrée. Tous les noeuds au bas du côté gauche, mis à part le noeud TRAFFICTRUNK, modélisent des types de demande. Ces noeuds représentent, pour le planificateur, des types de demande desservies par le réseau de la compagnie. La charge à ces noeuds est définie en terme d'unité de demande. Pour chaque type de demande, la base RCMB contient le trafic et le taux d'appel par unité de demande. De cette façon, SNAP peut calculer la charge en terme de trafic et de taux d'appel, pour un commutateur, en multipliant le nombre total d'unité de demande par le trafic ou taux d'appel par unité de demande correspondant. Tous les noeuds principaux d'entrée du côté droit modélisent des types de liaisons. Ils représentent les différentes technologies des liens entre les commutateurs du réseau. Nous décrirons plus loin le noeud TRAFFICTRUNK et la dichotomie de la base RCMB.

La deuxième classe est constituée des **noeuds de type ressource**. C'est le noeud le plus commun. Le noeud POTS LCM (à la Figure 4), ayant comme ressource LCM (Line Concentrating Module), en est un exemple. La fonction des cartes LCM dans un DMS-100 est de concentrer les canaux DS0 en provenance du réseau d'accès. Le calcul de la quantité d'équipement requis est effectué par la fonction STANDARD. Nous verrons à la page 21 la description de cette fonction.

La troisième classe de noeuds est caractérisée pas sa forme en losange: ce sont les **noeuds stratégiques**[1]. La différence entre ceux-ci et les noeuds de type ressource vient du fait que des décisions doivent être prise, soit par le planificateur, soit par la topologie du réseau. La décision provient de la topologie du réseau si l'arc émanant du noeud possède un point de changement de commutateur (voir l'exemple donné précédemment, p. 16). Dans le graphe, la décision indique dans quelle direction l'information présente à l'entrée du noeud doit être distribuée entre les différents types de serveurs en aval de celui-ci.

La Figure 7 montre ce que l'on entend par type de serveur. Chacun des trois ensembles de noeud {B,H,E}, {C,I,F} et {D,J,G} définissent un type de serveur pour le noeud stratégique B de la configuration LCMCtnts. Une configuration est, par définition, un ensemble de noeuds regroupés par le même nom de configuration[2]. Les trois types de serveurs modélisent les trois grosseurs de baies dans lesquelles on peut installer les cartes LCM. Un type de serveur est défini, strictement parlant, comme l'enfant d'un noeud stratégique. Le planificateur peut décider lui-même du type de serveur à utiliser, sinon SNAP utilise le type de serveur par défaut défini dans la base RCMB[3].

1. Les noeuds principaux d'entrée sont aussi des noeuds stratégiques.
2. La configuration à laquelle appartient un noeud est un attribut de celui-ci.
3. Le programme de validation de la base RCMB ne permettra la compilation de la base que si tous les noeuds stratégiques possèdent un type de serveur par défaut.

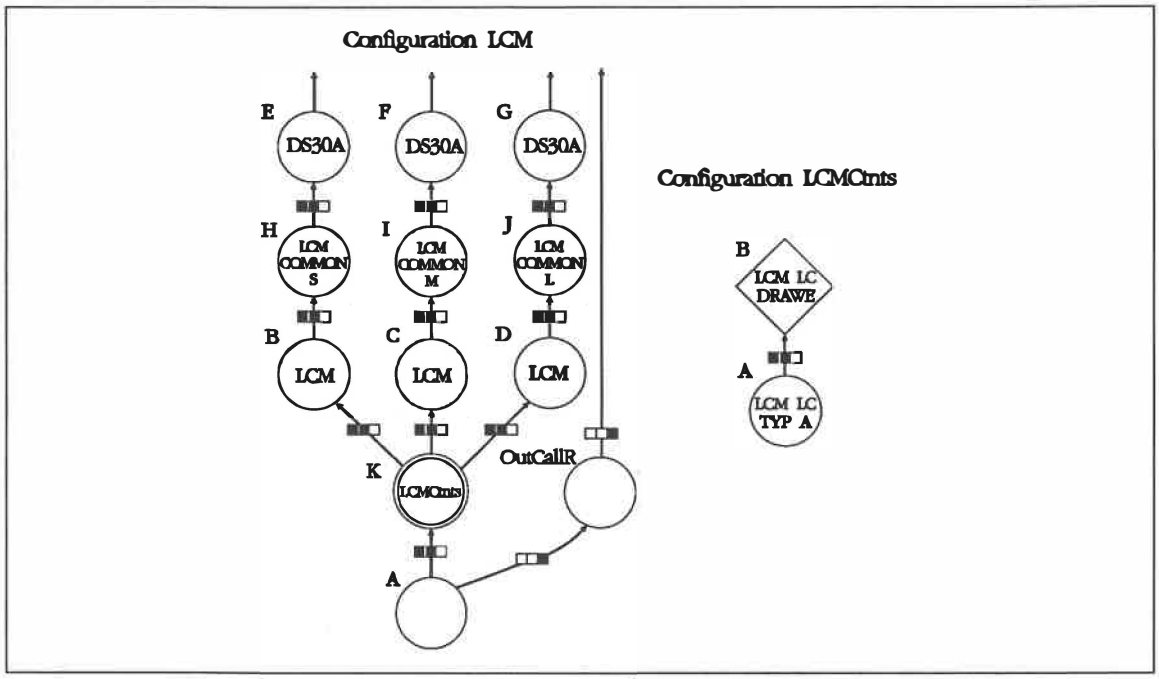


Figure 7. Type de serveur et appel de configuration

Le dernière classe est composée des **noeuds d'appel de configuration.** Les noeuds d'appel de configuration agissent comme des macros dans un langage de programmation. Lorsque la base RCMB est compilée[1], l'ensemble des noeuds qui forment la configuration appelée, au centre du cercle double, sera copié à tous les endroits où la configuration est appelée. Par exemple, à la Figure 7, nous voyons la configuration LCM qui doit remplacer chacun des deux noeuds d'appel POTS LCM et CENTRX LCM de la Figure 4. De la même façon, la configuration LCMCtnts remplace le noeud K de la configuration LCM. Ce mécanisme permet de ne définir qu'à un seul endroit un ensemble de ressources et de regrouper les ressources qui forment une unité logique ou physique. La modélisation est ainsi facilitée puisque l'on peut définir à l'aide de ce mécanisme des niveaux différents de détails.

1. Le but de la compilation est de convertir l'information de la base RCMB dans un format accessible à SNAP. La base RCMB est gérée par un programme de gestion de base de données relationnelles appelé GERM (General Entity Relationship Model), développé par RBN.

Dichotomie de la base RCMB

À la Figure 4, on constate que la base RCMB est scindée en deux parties: le côté gauche et le côté droit. Le côté gauche est aussi appelé côté ligne; les quantités calculées de ce côté dépendent de la charge du réseau d'accès. Le côté droit est appelé côté liaison puisque tous les noeuds principaux d'entrée, sans exception, qui sont au bas de celui-ci, modélisent des types de liaisons. Comme nous l'avons vu précédemment, tous les noeuds d'entrée du côté gauche représentent un type de demande, à l'exception du noeud TRAFFICTRUNK.

Voici comment le noeud TRAFFICTRUNK fait le lien entre les deux côtés. La dichotomie de la base RCMB est artificielle dans le sens suivant: par exemple, la configuration 100 Core (noeud E, Figure 4) possède un noeud qui modélise le CPU (Central Processing Unit) d'un DMS-100[1]. Contrairement aux cartes LCM, le CPU d'un DMS-100 n'est pas une ressource dédiée aux lignes. Le CPU d'un DMS-100 doit gérer le trafic des liaisons. La tâche du noeud TRAFFICTRUNK consiste à fournir une approximation du trafic des liaisons. De cette façon le planificateur peut déterminer, sans sous-estimation, la quantité de ressources utilisées simultanément par les liaisons et les lignes. La dichotomie de la base RCMB et l'approximation du noeud TRAFFICTRUNK sont des conditions nécessaires à la convergence de l'algorithme d'approvisionnement[2]. C'est la raison principale de l'existence de la dichotomie ligne/liaison. Nous verrons, à la section 1.2.3 (p. 23), comment l'algorithme d'approvisionnent traite chacun des côtés.

Les racines de cette dichotomie sont l'architecture des réseaux et la nature statistique du trafic. Le trafic du réseau d'accès est considéré comme aléatoire[3], hypothèse qui est vérifiée dans les réseaux actuels[4]. Par contre le trafic des liaisons,

1. La configuration 100 Core est illustrée au Chapitre 4.
2. On peut trouver la preuve de la convergence à l'Appendice A.
3. La distribution de l'arrivée est Poisson, le temps de service est exponentiellement distribué.
4. Du moins pour les conversations téléphoniques.

particulièrement le trafic de débordement, ne possède plus la même nature statistique. Les paramètres décrivant le trafic des liaisons dépendent de facteurs comme le routage, le nombre de liaisons ainsi que de l'amplitude du trafic. De plus, l'architecture du réseau de liaisons crée une interdépendance entre les commutateurs puisque tous les commutateurs y sont ratachés. Une décision prise à un commutateur influence donc la CMA de d'autres commutateurs.

L'idéal serait de disposer d'un algorithme parallèle qui solutionne le problème d'une façon globale. Cette solution est difficilement envisageable puisque pour le même scénario SNAP doit produire les mêmes résultats: les algorithmes utilisés pour SNAP doivent être déterministes. Pour contourner cet obstacle, la solution est de *séquentialiser* le problème, c'est-à-dire isoler les commutateurs pour trouver leur point de saturation. L'algorithme d'approvisionnement utilise cette méthode. La séquentialisation et l'isolation rend sans conséquence l'ordre dans lequel les congestions sont résolues. Ce procédé nous assure aussi de la convergence de l'algorithme d'approvisionnement. Ce critère est très important: il doit être respecté lors des modifications. On peut trouver les conditions nécessaires à la convergence de l'algorithme d'approvisionnement à l'Appendice A.

La fonction STANDARD

Chaque arc de la base RCMB conduit au moins un des trois types d'information suivants: volume, trafic et taux d'appels. La Figure 4 illustre quel type d'information est conduit à l'aide du "domino", placé sur l'arc, et la présence ou non d'un motif dans la case appropriée. Tous les noeuds de la base RCMB, mis à part les noeuds d'entrée, font appel à la fonction STANDARD pour le calcul de la quantité de ressources. Le nom de la fonction utilisée n'est qu'un attribut (comme la configuration) du noeud, il est donc possible d'en définir d'autres au besoin. Le processus de calcul de la quantité de ressource à un noeud donné, avec la fonction STANDARD, procède comme suit:

- les quantités à l'entrée sont sommées par type d'information,
- ensuite, pour chacun des types d'information présents à l'entrée, la fonction effectue la conversion voulue (de volume, trafic, taux d'appel à volume de ressources) à l'aide de la table de capacité désignée par le noeud,
- à partir des volumes calculés à l'étape précédente, le plus grand volume est retenu comme celui nécessaire pour rencontrer les objectifs de performance.

Le choix du plus grand volume, à la dernière étape, indique que les résultats de SNAP proviennent d'une analyse de type pire cas. Pour le calcul des ressources aux noeuds fils, les totaux du trafic et du taux d'appel sont transmis inchangés à la sortie du noeud. Avec le même volume, SNAP détermine s'il y a eu augmentation ou diminution avec la dernière année et utilise les flux monétaires correspondants pour la production des rapports (Figure 8).

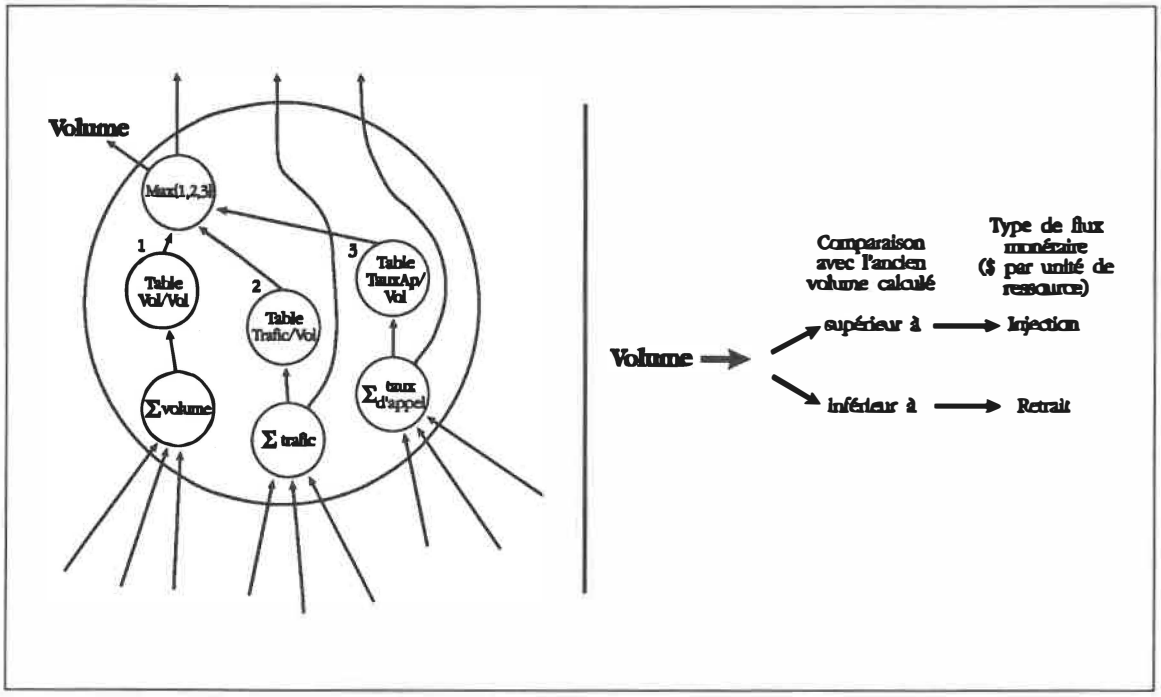


Figure 8. **Le noeud STANDARD**

1.2.3 Algorithmes et modèles

Cette sous-section décrit l'algorithme d'approvisionnement ainsi que les modèles et les algorithmes qui lui sont rattachés. La première partie donne une description fonctionnelle de l'algorithme d'approvisionnement. La deuxième partie décrit comment on calcule le trafic intercommutateur total d'un commutateur. Enfin, la troisième et dernière partie montre comment le trafic intercommutateur est distribué entre les paires de commutateurs à l'aide du modèle gravitationnel.

L'algorithme d'approvisionnement

L'algorithme d'approvisionnement de SNAP est composé de quatre modules nommés Assigner, Compressor, Installer et Trunker. Chacun des modules exécute une fonction bien précise que nous décrirons bientôt. Ces quatres modules partagent une structure de données commune appelée Resources. La structure de données Resources contient l'état entier du réseau simulé. Les modules communiquent entre eux via l'information indiquée au-dessus et au-dessous des flèches de la Figure 9. La Figure 9 schématise l'algorithme avec les quatres modules ainsi que leur relation avec la base DSN et RCMB. Pour la description de l'algorithme, nous utiliserons un modèle par bloc avec un échange de commandes d'installation[1]. Voici la description des modules et leurs interactions.

L'algorithme d'approvisionnement débute avec le module Assigner. Le module Assigner interprète les stratégies du scénario. L'interprétation des stratégies résulte en une série d'actions dont les plus usuelles sont: l'ouverture ou la fermeture de commutateurs, le transfert partiel ou total de la demande d'un type de serveur à un autre ou d'un commutateur à un autre, le choix d'un nouveau serveur pour la croissance d'un type de demande. Une fois les stratégies interprétées, le module Assigner génère des commandes d'installation. Les commandes d'installation

1. Le mécanisme de commande d'installation ne décrit pas exactement l'interaction entre les modules. La description est toutefois valable pour le niveau de détail nécessaire.

peuvent être visualisés comme des fiches de demande pour l'introduction (positive ou négative) d'équipements dans le réseau. Cet ensemble de commandes d'installation constitue la première tentative du module Assigner dans la recherche du réseau optimal. Cet ensemble de commandes d'installation est transmis au module Compressor.

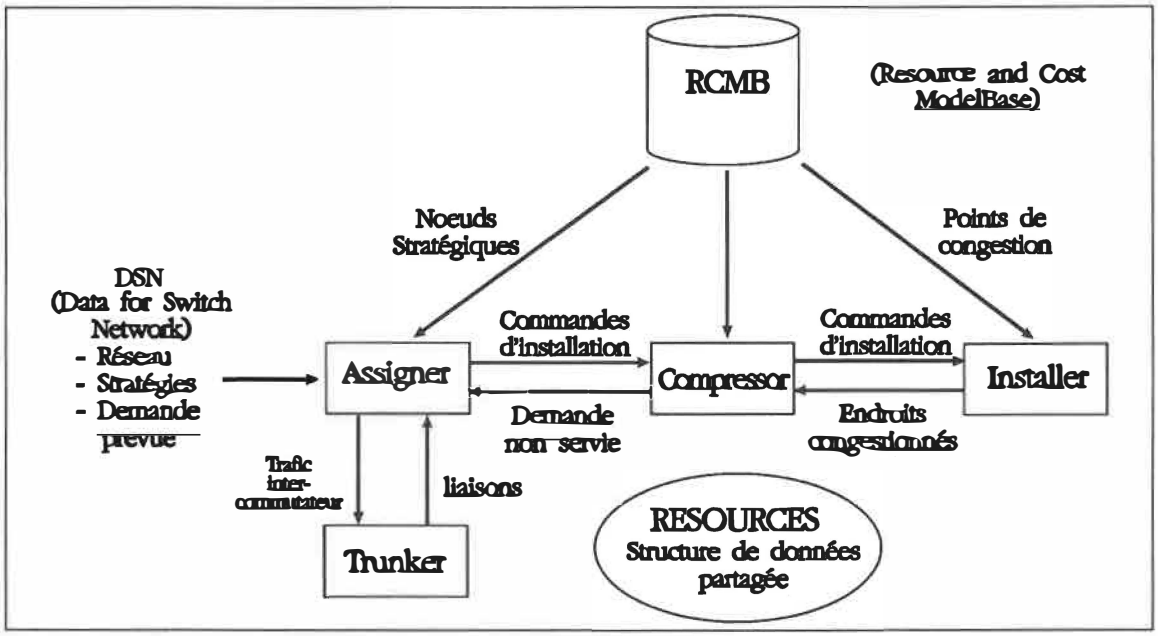


Figure 9. Schéma bloc de l'algorithme d'approvisionnement

Le module Compressor assume un rôle d'intermédiaire entre le module Assigner et le module Installer. La fonction du module Compressor consiste à trouver la charge maximum que le réseau peut absorber, sans causer de congestion. Contrairement au module Assigner, le module Compressor ne peut ouvrir un commutateur pour absorber l'excédant de demande. Le module Compressor ne peut que réduire (compresser) les commandes d'installation. Afin de détecter les congestions dans le réseau, les commandes d'installation sont transmis au module Installer.

Le module Installer reçoit les commandes d'installation et les exécute. Une fois les commandes exécutés, le module Installer détecte les congestions. Les congestions sont détectées à certains noeuds de la base RCMB. Ces noeuds se retrouvent dans la liste des points de congestion[1]. Ces noeuds sont constamment surveillés par le module Installer: ils ne peuvent produire plus d'une unité de leur ressource. On dira qu'un point de congestion est activé s'il génère plus d'une unité.

Par exemple, la configuration 100 Core de la Figure 4 (p. 15) contient le noeud C qui est défini comme un point de congestion (la configuration 100 Core est illustrée à la Figure 23, p. 92). Le noeud C modélise le CPU[2]. Si le noeud C génère un volume supérieur à un, on sait que la puissance de calcul du commutateur est excédée par la charge simulée. Le module Compressor doit alors réduire la demande pour le commutateur jusqu'à ce que la quantité générée retombe à un.

Lorsque l'exécution du module Installer est terminée, une liste de tous les noeuds qui causent les congestions dans le réseau est renvoyée au module Compressor[3]. Afin de trouver la charge maximale admissible, le module Compressor utilise un algorithme de recherche binaire. L'algorithme de recherche binaire génère une série de commandes d'installation en fonction de la congestion ou non de la dernière tentative. Si la dernière tentative du module Compressor ne cause aucune congestion, le programme vérifie si l'écart entre cette tentative et la précédente est en deça de la tolérance (en pourcentage) demandée par le planificateur. Si c'est le cas, alors l'exécution du module Compressor se termine. Finalement, le module Compressor renvoit au module Assigner la liste de la demande qui n'a pu être satisfaite par le réseau.

1. Cette liste fait partie de la base RCMB.
2. Ici le CPU fait référence à tous les modules et cartes qui forment le processeur central d'un commutateur DMS-100.
3. Cette liste comprend tous les parents des points de congestion activés qui sont des types de demande.

Avec les nouvelles données provenant du module Compressor, le module Assigner vérifie si le planificateur a défini des stratégies suceptibles de résoudre les congestions. Le module Assigner génère des tentatives jusqu'à ce qu'il ne dispose plus de stratégies de décongestion ou que la totalité de la demande ait pu être absorbée. La demande qui n'a pu être satisfaite apparaît dans le rapport d'activité produit par SNAP.

La boucle, qui résulte des appels au module Comressor par le module Assigner, est appelée boucle Assigner. La boucle qui résulte des appels du module Installer par le module Compressor est appelé boucle Compressor. La convergence de la boucle Assigner est assurée par le nombre fini de stratégies et de commutateurs. Par contre, la convergence de la boucle Compressor est plus subtile (Appendice A).

L'algorithme partiel précédemment décrit (Assigner-Compressor-Installer), approvisionne dans une première phase le côté gauche de la base RCMB. Lorsque la boucle Assigner du côté gauche est terminée, le module Trunker transforme le trafic intercommutateur total des commutateurs en nombre de liaisons (la description du module Trunker sera donnée à la section 1.2.4 qui décrit les algorithmes dimensionneurs de réseaux de liaisons). Le module Trunker se trouve à calculer le nombre de liaisons et le trafic pour les noeuds d'entrée du côté droit. La Figure 10 illustre le diagramme fonctionnel et la relation avec chacun des côtés de la base RCMB.

L'algorithme, décrit plus haut pour le côté gauche, est alors répété pour le côté droit de la base RCMB, à la différence près qu'aucun point de congestion ne peut se trouver de ce côté. La raison de cette contrainte est évidente lorsque l'on observe la Figure 10. Si on reprend l'exemple du noeud CPU de la configuration 100 Core, on voit que SNAP peut retracer les noeuds de types de demande à l'aide d'une recherche de type profondeur d'abord. Cette recherche débute au point de congestion activé et explore les parents de ce noeud. Cette exploration est impossible à partir des noeuds du côté droit.

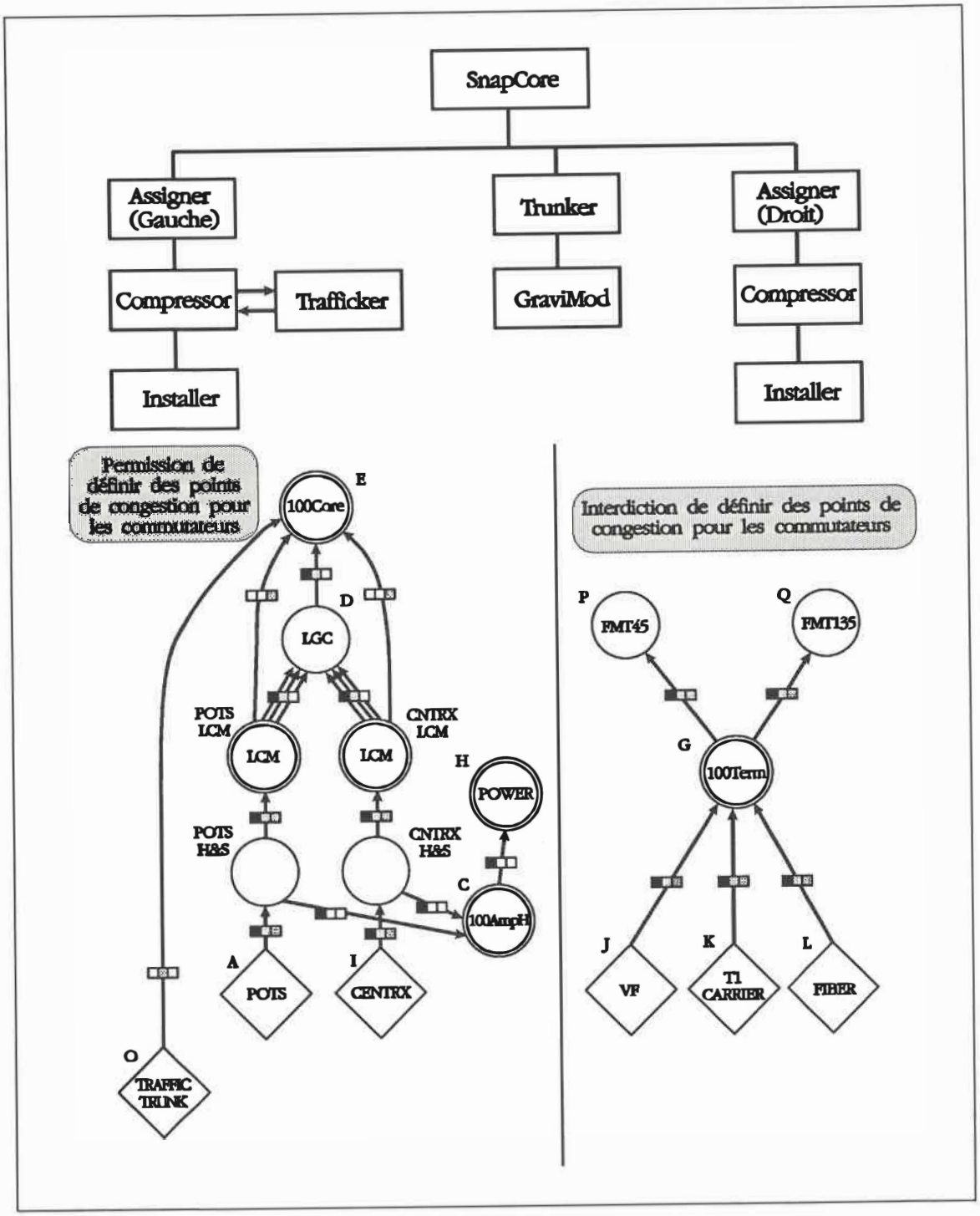


Figure 10 Organigramme fonctionnel et la base RCMB

On pourrait penser qu'il suffit d'exécuter le module Trunker, une fois le trafic intercommutateur connu, et d'appliquer cette information aux noeuds du côté droit simultanément avec les noeuds du côté gauche pour résoudre le problème. Malheureusement, la convergence de la boucle Compressor (donc de l'algorithme d'approvisionnement) n'est assurée que si et seulement si les commutateurs n'interagissent pas entre eux pendant la boucle. Nous verrons plus loin comment le changement de charge à un commutateur influence tous les autres, avec le modèle gravitationnel utilisé par le module Trunker. Mais auparavant, voici comment est calculé le trafic intercommutateur.

Le module Trafficker

Le module Trafficker a comme fonction d'interpréter les relations commutateurs distants-récepteurs, de propager et de calculer le trafic intercommutateur. Le modèle derrière le module Trafficker consiste à scinder le trafic provenant du réseau d'accès en deux trafics distincts: le trafic intracommutateur et le trafic intercommutateur. Pour ce faire, une table de la base RCMB donne le pourcentage de trafic intracommutateur par type de demande.

La Figure 11 illustre un exemple d'arbre formé par deux commutateurs RLCM (Remote Line Concentrating Module) qui possèdent un lien ombilical avec un commutateur RSC (Remote Switching Center). Ce même RSC possède un lien ombilical avec un DMS-100. Dans cet exemple, le RLCM A est équipé pour commuter son trafic intracommutateur. Par contre, le RLCM B ne possède pas cet équipement et dépend donc du RSC C pour commuter son trafic intracommutateur.

Voici la définition des variables utilisées à la Figure 11: TOTCCS(S) est égal au nombre d'unités de demande du commutateur S multiplié par le trafic par unité de demande, IntraCCS%(S) est égal au pourcentage de trafic intracommutateur, Ombilical(S) est égal au trafic du lien ombilical du cummutateur S. La Figure 11 montre comment le trafic est propagé à partir des commutateurs distants vers le haut de l'arbre.

Dans le modèle du réseau de liaisons utilisé par SNAP, les seuls commutateurs connectés au réseau de liaisons sont les racines. On les appelle commutateurs principaux (CP[1]). Conséquemment, seuls les CP possèdent un noeud TRAFFICTRUNK[2]. Le trafic calculé pour ce noeud est égal au trafic intercommutateur total trouvé pour l'arbre. Dans l'exemple cité plus haut, le trafic émanant du noeud TRAFFICTRUNK est égal à (1 - IntraCCS%(D))*TOTCCS(D).

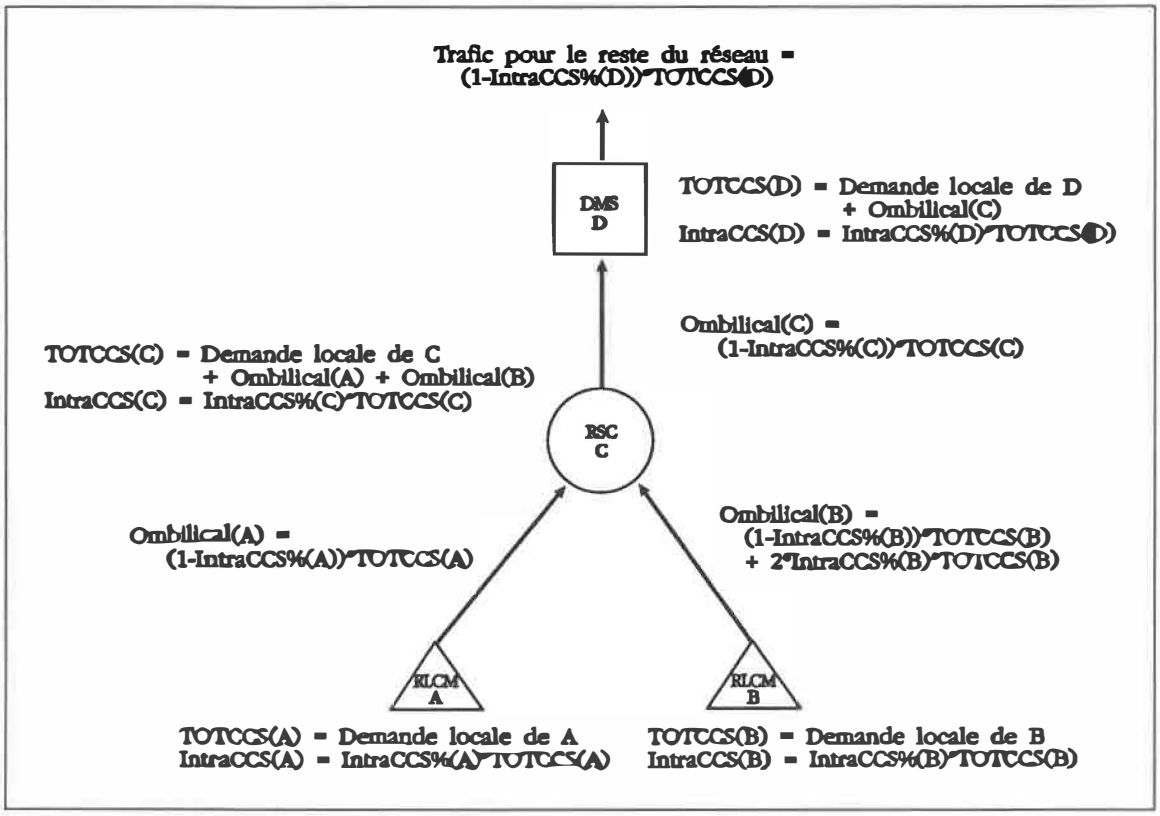


Figure 11. Trafficker: propagation du trafic

1. En fait, nous ne distinguerons plus l'arbre complet et le commutateur principal puisque cet ensemble forme une seule entité dans le calcul du trafic intercommutateur. À noter que chaque arbre ne possède qu'une seule racine et qu'un commutateur ne peut être membre que d'un seul arbre. La preuve de la convergence de l'algorithme d'approvisionnement dépend de cette caractéristique.
2. Nous avons vu que ce noeud a pour but de fournir une approximation du trafic intercommutateur pendant que l'algorithme d'approvisionnement traite le côté gauche de la base RCMB.

Le module Trafficker est appelé par le module Compressor à chaque fois que la demande change aux noeuds principaux d'entrée du côté gauche. Afin d'estimer le nombre de liaisons entre les commutateurs, on doit disposer d'un modèle qui distribue le trafic intercommutateur total entre les paires de commutateurs. Cette étape est nécessaire puisque, comme nous le verrons à la section 1.2.4, le dimensionneur de réseau de liaisons de SNAP (DRL/SNAP) nécessite une matrice de trafics. Cette matrice est générée par un modèle de distribution appelé modèle gravitationnel.

Le modèle gravitationnel

Le modèle gravitionnel estime le trafic d'un CP vers un autre dans le réseau local à l'aide d'une fonction basée sur deux axiomes. Le premier axiome indique que le trafic provenant d'un CP vers l'autre CP est directement proportionnel au produit des trafics intercommutateurs. Le deuxième axiome suppose que le trafic est inversement proportionnel au carré de la distance entre les deux CP[1]. Les données nécessaires au modèle gravitionnel pour le calcul incluent: le trafic intercommutateur total des CP, la matrice des distances entre les CP et la description des EAS (voir la page 10 pour la description) du réseau. À partir de ces données, le modèle gravitionnel produit une matrice de trafics. La formule du modèle gravitationnel apparaît à la page 43. Les algorithmes qui transforment les matrices de trafics en nombre de liaisons sont appelés dimensionneurs de réseaux de liaisons (DRL). La sous-section qui suit présente un survol de ce type d'algorithme (dont DRL/SNAP).

En résumé, tout l'algorithme d'approvisionnement peut être vu comme un processus itératif qui construit une série de réseau, ayant comme origine l'état du réseau de l'année précédente, jusqu'au nouvel état. La transition entre les deux états, pour le calcul des flux monétaires, ne s'effectue qu'à la toute fin de l'algorithme d'aprovisionnement.

1. Ces deux axiomes sont identiques à ceux de la formule pour calculer la force gravitationnelle entre deux corps, sauf qu'ici, la masse est remplacée par le trafic.

1.2.4 Dimensionneur de réseaux de liaisons

Un dimensionneur de réseaux de liaisons (DRL) désigne un algorithme qui, à partir de la matrice de trafic entre les commutateurs, génère une matrice du nombre de liaisons à installer. Les DRL génèrent la matrice de liaisons en minimisant la fonction écomomique du coût du réseau. De plus, si le réseau de liaisons est dimensionné selon cette matrice, on est assuré que le blocage ne dépasse pas le blocage maximum spécifié par le planificateur. Les DRL sont tous des algorithmes heuristiques: la solution exacte de ce type de problème est irréalisable (même pour de petits réseaux).

Pour produire la matrice de liaisons, un DRL nécessite les informations suivantes:

- les tables de routages,
- une ou plusieurs matrices de trafic entre les commutateurs (plusieurs pour ceux permettant une analyse multi-charges),
- la description du réseau (position des commutateurs),
- la fonction du coût des liaisons,
- la matrice de blocage maximum permis entre deux commutateurs,
- la modularité (terme défini plus loin) entre les commutateurs,
- les liaisons existantes (pour ceux qui le supportent).

Il existe présentement, chez RBN, plusieurs DRL. Ces DRL se différencient les uns des autres par le type de réseaux de liaisons modélisables. Les deux grandes catégories sont le système de routage et le sens de la signalisation des liaisons. Voici une courte description de chacune de ces catégories.

Deux systèmes de routage sont présentement en usage: le système FHR (Fixed Hierarchical Routing) et le système DCR (Dynamically Controlled Routing). Le système FHR produit la séquence de liens possibles lors de l'établissement d'une communication à partir de table fixe, d'où le nom de cette méthode. Le système FHR n'exploite pas complètement les capacités des nouveaux commutateurs numériques puisque ce système à été développé à l'époque pour l'ancienne génération de commutateurs électro-mécanique. Ce système, avec la numérisation croissante des réseaux plublics, est en voie d'être remplacé par le système DCR.

Le système DCR est composé de noeuds intelligents (NI). Les NI sont des commutateurs numériques spécialement équipés (logiciel et matériel). Après une période fixe (10 secondes environ), chaque NI envoit à un processeur central (PC) l'occupation des liens qui lui sont rattachés. Le PC dispose alors, presque en temps réel, de l'état global du réseau de liaisons. Le PC, à l'aide d'algorithmes spéciaux, remet à jour les tables de routages des NI. Plusieurs avantages découlent de l'emploi de ce système. Voici la liste des plus importants:

- une meilleure utilisation de la capacité du réseau de liaisons en place. Le système DCR permet de retarder l'installation de nouvelles liaisons, par exemple en exploitant le fait que les périodes de pointes commerciales et résidentielles ne sont pas les mêmes,
- une récupération plus rapide du réseau en cas de panne ou de congestion en changeant les tables pour éviter les commutateurs inutilisables ou engorgés,
- facilite la gestion pour les opérateurs de réseau puisque le système dispose de l'état global du réseau et intervient plus rapidement, ce qui est important dans les cas où la charge du réseau dépasse la charge maximale de conception (les performances se dégradent très rapidement dépassé cette limite).

La deuxième catégorie comprend le sens de signalisation des liaisons. Les liaisons peuvent être de type simple sens ou double sens. Simple et double sens fait référence au sens dans lequel peut s'effectuer la signalisation, et non pas dans quelle sens le signal peut se propager (une fois la communication établie le sens de la signalisation est non pertinent). Par exemple, une liaison simple sens entre un CP A et un CP B avec comme sens de signalisation A→B ne peut être utilisée pour l'établissement d'une communication émanant du CP B vers le CP A. Avec une liaison double sens, la signalisation peut s'effectuer dans les deux sens, à l'aide un mécanisme de prévention ou de détection de collision. Pour le même trafic, le nombre de liaisons bidirectionnelles dans un lien est toujours plus petit ou égal au nombre de liaisons unidirectionnelles que nécessiterait le lien pour assurer le même service. Les liaisons unidirectionnelles sont utilisées dans les réseaux locaux urbains, où il y a peu à gagner avec des liaisons bidirectionnelles. Les liaisons bidirectionnelles sont utilisées principalement pour les réseaux interurbains.

Lorsque des liaisons sont installées entre deux CP, on rajoute un ou des modules de transmission à chaque extrémité. Chaque paire de modules ajoute n liaisons, où n est appelé modularité. Dans les réseaux urbains, la majorité des systèmes installés sont de type T1, via un câble de cuivre ou une fibre optique. Le système T1 possède une modularité de 24 liaisons. Dans un module composé de liaisons simple sens, le sens de signalisation de chacune des liaisons peut être établi de façon indépendante.

Le DRL de SNAP, qui est codé dans le module Trunker, suppose le réseau suivant: toutes les paires de commutateurs avec un trafic intercommutateur non nul possèdent un lien de type direct final, composé de liaisons double sens, avec une modularité de un. Un lien direct final se comporte comme un lien final, c'est-à-dire que ce type de lien ne génère aucun trafic de débordement. À la différence du lien

final, il relie deux CP au lieu d'un CP et son tandem. Avant de trouver le nombre de liaisons entre deux CP du réseau, le module Trunker calcule la moyenne arithmétique du trafic entre les deux CP. Si $CCS(A,B)$ est égal au trafic émanant du CP A vers le CP B, alors le nombre de liaisons du lien est déterminé pour le trafic $\frac{1}{2}[CCS(A,B) + CCS(B,A)]$.

Le nombre de liaisons est donné par la table TRUNKERPARAMETER de la base RCMB. Lorsque le nombre de liaisons est connu, Trunker les distribue parmis les différents types de liaisons en utilisant les pourcentages définis dans la base RCMB. Ce réseau modélise *logiquement* le réseau de liaisons. Ce modèle repose sur l'hypothèse que le nombre de liaisons des commutateurs de Classe 5 est peu inluencé par la topologie et le routage du réseau physique.

Une autre particularité de DRL/SNAP vient du fait qu'on ne planifie que des partie de réseaux seulement: le temps de calcul prohibitif empêche la modélisation d'un réseau complet comme le Québec par exemple. Le temps de calcul augmente encore lorsque le degré de détail de la base RCMB est grand. Puisque le nombre liaisons dépend aussi du trafic provenant des autres sous-réseaux rattachés au sous-réseau étudié, SNAP utilise le concept d'immeuble EAS. Un immeuble EAS est un immeuble rattaché au sous-réseau étudié mais donc les commutateurs ne sont pas modélisés: SNAP ne fait que calculer leur trafic intercommutateur comme si l'immmeuble ne possédait qu'un seul et unique commutateur. De cette façon, toujours à l'aide du MG, DRL/SNAP peut mieux estimer le trafic des liaisons, et conséquemment, le nombre de liaisons.

1.3. Définition du problème

Le logiciel SNAP, dans sa version actuelle, ne peut modéliser les commutateurs de type tandem. Deux éléments, soit la dichotomie de la base RCMB et DRL/SNAP, empèchent la modélisation. Cette section débute par la description des deux éléments et la raison de leur imcompatibilité avec la modélisation des tandems. Ensuite, nous verrons les problèmes soulevés par l'introduction d'un nouveau DRL et la solution retenue.

1.3.1 Les obstacles à la modélisation

Le premier obsctacle à la modélisation des tandems réside dans la dichotomie de la base RCMB. Nous avons vu, à la section 1.2.1, que les tandems effectuaient des connections de type liaison à liaison. La congestion ou non d'un tandem dépend du nombre de liaisons qui lui sont rattachées et du trafic acheminé par celles-ci. Si on modélise un tandem avec la version actuelle de SNAP, l'ensemble des ressources se retrouve du côté droit puisque le tandem n'est pas connecté directement au réseau d'accès. Mais puisqu'il est impossible de définir des points de congestion du côté droit (voir explication p. 20), on ne pourrait trouver la charge maximale du tandem à l'aide de l'algorithme d'approvisionnement. Le deuxième obstacle découle aussi de cette dichotomie mais concerne le DRL de SNAP.

Comme nous l'avons vu, le DRL utilisé pour SNAP modélise un réseau logique de liaisons. Toutefois, la congestion d'un tandem dépend du réseau physique. Avec la version actuelle, un tandem ne posséderait aucune liaison puisque seul les commutateurs possédant un trafic intercommutateur non nul se voient attribuer des liaisons. Dans le réseau, le tandem commute le trafic intercommuateur, mais ce trafic n'est pas généré par son réseau d'accès, il provient des CP qui lui sont rattachés. De plus, l'amplitude de ce trafic dépend du nombre de liaisons des liens HU entre les CP. Il faut donc remplacer DRL/SNAP par un algorithme adéquat.

Il existe présentement, chez RBN, toute une famille de DRL développés au cours des années. Il suffirait donc d'intégrer SNAP et le DRL choisi pour résoudre la deuxième étape. Cette solution, qui relève du genie des logiciels, soulève plusieurs questions concernant le temps de calcul nécessaire, la convergence de l'algorithme d'approvisionnement, les données supplémentaires requises, et enfin l'espace mémoire nécessaire pour les données supplémentaire et le programme résultant de l'intégration. Voici une courte description de chacun des problèmes soulevés.

Le temps de calcul supplémentaire vient du fait que SNAP requiert déjà un temps CPU important. Si un DRL est incorporé, il devra être utilisé de façon itérative à cause du fonctionnement de l'algorithme d'approvisionnement. Si le temps CPU résultant de l'intégration est trop grand, les planificateurs, qui généralement doivent respecter un budget en terme de temps CPU, devront faire des compromis sur d'autres aspects comme la grandeur du sous-réseau, le degré de détail ou la période couverte par l'étude.

La base de données DSN, qui contient l'ensemble des données d'entrée pour le logiciel SNAP, demande déjà beaucoup de travail et d'information. L'adjonction d'un nombre important de données peut rendre l'outil moins "attrayant", surtout si l'information supplémentaire obtenue est marginale.

Certaines études exigent une machine virtuelle[1] avec 12 Moctets de mémoire et plus, ce qui est beaucoup. Le nouveau programme et les données supplémentaires introduiraient probablement de nouvelles contraintes du type décrit pour le temps de calcul supplémentaire.

1. Le noyau est éxécuté avec le système d'exploitation VM/CMS.

Finalement, le problème le plus important concerne la convergence de l'algorithme d'approvisionnement. Tout DRL implique nécessairement tous les commutateurs du réseau. Mais l'algorithme d'approvisionnent ne converge que si un mécanisme d'isolement existe lors de la boucle Compressor. Qu'importe le DRL choisi, il faut donc trouver un tel mécanisme.

1.3.2 Solution retenue

La solution retenue, pour la modélisation des tandems, attaque le problème d'un autre angle. Au lieu d'intégrer les deux logiciels (SNAP et le DRL), on trouve un algorithme qui donne une approximation du nombre de liaisons. Cet algorithme doit générer un réseau de liaisons sub-optimal de façon à ce que, lorsque le logiciel DRL choisi dimensionnera à son tour les liaisons, on est assuré qu'il n'y ait aucune congestion dans le réseau. L'algorithme doit aussi minimiser les nouvelles données requises et, si possible, être performant et compact. Cette solution, si elle est viable, permettra de garder les deux logiciels indépendants l'un de l'autre. Le seul élément requis serait un interface entre les logiciels. La solution sera élaborée en deux étapes, soit une étape par obstacle.

La première étape consiste à intégrer le calcul des ressources en une seule et même phase. À la fin de cette première étape, le logiciel SNAP et la base RCMB ne devront comprendre qu'un seul côté. De cette façon, toute les ressources pourront participer à la recherche de la charge maximale.

La deuxième étape consiste à développer un "petit" DRL afin de remplacer DRL/SNAP. Ce DRL devrait être rapide, compact et nécessiter peu d'information supplémentaire. Les résultats produits seront nécessairement sub-optimaux puisque des contraintes devront être utilisées pour réduire le problème abordé par les vrais DRL. Il faut bien noter que cette deuxième étape n'a pas pour but de remplacer un vrai logiciel DRL, mais seulement d'approximer la solution optimale de celui-ci pendant que SNAP recherche le réseau optimal.

Dans le reste du mémoire, le terme solution optimale fera référence à la solution, fournie par le logiciel SNAP, donc l'écart en pourcentage avec la charge maximale adimissible théorique ne dépasse pas une tolérance fixée a priori par le planificateur. C'est à ce type de solution qu'est rattaché le terme "solution optimale".

1.4. Travaux antérieurs

Il n'existe pas de travaux antérieurs à propos du problème spécifique qui fait l'objet de ce mémoire, c'est-à-dire la modélisation des tandems par le logiciel SNAP. Par contre, voici une liste de livres, articles et rapports qui contiennent des descriptions et des théories de différents éléments utilisés dans la réalisation de la méthodologie. Les références complètes sont données dans la bibliographie.

Pour une description des hypothèses et savoir d'où provient le modèle gravitationnel, on peut consulter les deux ouvrages suivants:

Principles of telecommunication-traffic engineering, de D. Bear, à la page 184, et

Telecommunications networks, de J.E. Flood, aux pp.304-308 pour une description plus détaillée.

Presque tous les livres traitant de l'ingénierie des réseaux de liaisons possèdent quelques pages sur la méthode dite du CCS économique (ECCS pour Economic CCS). Cette méthode est utilisée par tous les DRL pour trouver le réseau le plus économique. La méthode ECCS procède à l'optimisation du réseau de liaisons en approvisionnant les liens haut usage jusqu'au point où, il devient plus économique de rediriger le trafic vers un autre lien, lien dont le coût par unité de trafic est plus avantageux. Pour connaître la théorie de la méthode ECCS, on peut lire

Engineering and Operations in the Bell System, des Laboratoires AT&T Bell, aux pp. 169-172.

L'information concernant les DRL provient de deux rapports confidentiels, soit

Dimensioning for a FHR/HPR toll network, par A.S. Le Nir, R.M. Huberman et T. Drwiega (Recherches Bell-Northern, juillet 1984).

où HPR veut dire High Performance Routing (l'ancien nom pour DCR). Plusieurs procédures de calcul du trafic et du nombre de liaisons, décrites en pseudo-code à la fin de ce rapport, on été utilisées dans les programmes tests. Deuxièment,

Consolidation of Existing and New Methodology for a Mixed FHR/HPR Dimensioner, par S. Michailov et P. Higham (Recherches Bell-Northern, juillet 1987)

fait un survol des DRL développés au cours des années et présente le design, de niveau moyen, d'un algorithme DRL pour les réseaux mixtes FHR/DCR.

Et enfin, l'article à la base de la théorie du trafic de débordement,

Theories for Toll Traffic Engineering, par R.I. Wilkinson, paru dans Bell System Technical Journal, pp. 421-5124, en 1956.

La description du trafic de débordement décrite dans cet article, par ses deux premiers moments (moyenne et variance), est encore utilisée aujoud'hui par les DRL.

2. Méthodologie

Le présent chapitre est divisé en quatres sections. La première section décrit l'intégration du calcul de la charge maximale admissible (CMA) en une seule phase. La seconde section décrit la méthodologie du nouveau DRL. La troisième section explique les changements apportés au logiciel SNAP et à la base RCMB. Enfin, la dernière section montre comment la fonction Lambda prototype, développée à la première section, a été modifiée dans le nouveau contexte de la modélisation des tandems.

2.1. Intégration du calcul de la CMA

Cette section contient la méthodologie utilisée dans l'élaboration de la fonction permettant l'intégration du calcul de la charge maximale admissible (CMA). Cette fonction est appelée Lambda. La première sous-section décrit les hypothèses posées pour la solution du problème d'intégration. La deuxième sous-section explique la dérivation de la fonction Lambda. Cette sous-section est suivie de la description des caractéristiques de la fonction Lambda. La quatrième sous-section présente l'ancienne procédure utilisée pour la transformation du trafic en nombre de liaisons. Les deux dernières sous-sections présentent respectivement les changements à la base RCMB et les changements à l'algorithme d'approvisionnement.

2.1.1 Hypothèses

Le but visé avec la fonction Lambda est l'intégration du calcul de la CMA en une seule phase. Il ne s'agit donc pas de remplacer le trafic calculé par le modèle gravitationnel (MG), qui est la meilleure approximation du trafic intercommutateur que l'on peut obtenir à partir des données disponibles, mais bien de permettre au côté gauche d'avoir accès à ce trafic. Puisque l'on veut modéliser les ressources présentement du côté liaison avec celles du côté ligne, on doit trouver en plus une

fonction pour transformer le trafic en nombre de liaisons. Voici la liste des deux hypothèses, chacune avec une courte justification, qui ont permis la dérivation:

- ***le réseau est stable,***
  il n'y a pas de changement brusque de demande dans le réseau. Par exemple, il ne peut y avoir une augmentation soudaine de 40% de la demande dans un immeuble. Cette hypothèse rejoint l'objectif de développement continu visé par la planification et le comportement de la croissance des réseaux réels. De cette façon, on est assuré que le point de saturation "n'est pas loin" de la tentative de charge,

- ***une approximation linéaire du MG est suffisamment précise*** pour le calcul de la CMA. Cette approximation doit dépendre du trafic intercommutateur du seul CP congestionné, ceci dans le but d'assurer la convergence de l'algorithme d'approvisionnement. Cette hypothèse découle un peu de la première étant donné que la CMA est près de la tentative de charge. On suppose qu'une approximation linéaire devrait être suffisante.

La fonction linéaire de la deuxième hypothèse doit dépendre, comme mentionné, du seul trafic intercommutateur du CP congestionné. Cette condition découle directement de la preuve de convergence de l'algorithme d'approvisionnement. Cette preuve est donnée à l'Appendice A. En fait, c'est une condition *sine qua non*: il est impossible d'assurer la convergence si les CP interagissent entre eux *pendant* la boucle de Compressor. Avant de passer à la dérivation de la fonction Lambda, notons qu'aucune des deux hypothèses n'introduit de nouvelles contraintes à la modélisation.

2.1.2 La fonction Lambda prototype

La fonction Lambda est la fonction qui, à partir de la charge d'un CP, estime le trafic dans les deux sens de ses liaisons. Le trafic dans les deux sens est celui donné par le MG. Bref, cette fonction est appelée à remplacer le noeud TRAFFICTRUNK du côté gauche de la base RCMB. La fonction Lambda ci-bas a été développée pour la version actuelle du logiciel SNAP. Une version modifiée de cette fonction[1], décrite à la section 2.4., est utilisée par le nouveau DRL. Voici la liste des définitions nécessaires.

Définitions/p[;;;;;;;;;;

SNAP reconnaît deux types de technologie pour un commutateur: analogique et numérique. Pour simplifier la notation, une définition plus générale des types de technologie est utilisée. Cette définition inclut les deux premières catégories plus une troisième appelée EAS. Les commutateurs, déclarés de technologie EAS, sont les "commutateurs uniques" dans les immeubles de l'EAS du réseau étudié (voir p. 34).

$CCS(S_i,n)$	trafic intercommutateur provenant du commutateur principal S_i généré par l'échange n,
$CCS(S_i \mid n)$	trafic intercommutateur provenant de S_i généré par l'EAS de l 'échange n,
S	l'ensemble des index de tous les commutateurs,
S_t	l'ensemble des index de tous les commutateurs de technologie t ($S_t \subset S$),
E	l'ensemble des index de tous les échanges,
EAS_n	l'ensemble des index de tous les échanges de l'EAS de l'échange n,
K	exposant du modèle gravitationnel, plus K est grand, plus la distance entre les commutateurs a d'impact sur la distribution du trafic,

1. Dans le reste du mémoire, l'appellation fonction Lambda prototype fera référence à la fonction présentée dans cette section, pour la différencier de la version utilisée par le nouveau DRL.

TD "distance" entre les commutateurs d'un même immeuble. Cette valeur strictement positive (>0) est requise puisque que les distances entre les commutateurs sont déterminées à partir des coordonnées des immeubles. La distance entre deux commutateurs d'un même immeuble est donc de 0. Pour éviter une division par zéro, lors du calcul du trafic à l'aide du MG, cette constante a été rajoutée. Elle permet aussi d'influencer la distribution du trafic par le MG. Si *TD* est grand, moins le trafic "reste" à l'intérieur de l'immeuble et vice-versa. Le modèle qui résulte de l'addition des deux paramètres *TD* et *K* est appelé modèle gravitationnel généralisé,

et les fonctions

$$d(i,j) = \begin{cases} 0 & \text{si } i = j \\ (distance(S_i,S_j)+TD)^{-K} & \text{si } i \neq j \end{cases}$$

$$den(i,n) = \sum_{j \in S} CCS(S_j|n)\, d(i,j)$$

Dérivation

À l'aide des définitions ci-haut, le trafic de S_i vers S_j s'écrit, selon le modèle gravitationnel:

$$CCS(S_i,S_j) = \sum_{n \in E} \frac{CCS(S_i,n)\, CCS(S_j|n)\, d(i,j)}{den(i,n)}$$

Le trafic dans les deux entre S_i et S_j est donné par la moyenne arithmétique de $CCS(S_i,S_j)$ et $CCS(S_j,S_i)$, c'est-à-dire:

$$TWCCS(S_i,S_j) = \frac{1}{2}\left[\sum_{n \in E} \frac{CCS(S_i,n)\, CCS(S_j|n)\, d(i,j)}{den(i,n)} + \sum_{n \in E} \frac{CCS(S_j,n)\, CCS(S_i|n)\, d(j,i)}{den(j,n)} \right]$$

Le trafic dans les deux sens entre S_i et tous les commutateurs de technologie t est donné par la somme sur tout les commutateurs S_j tels que $j \in S_t$,

$$TWCCS(S_i|S_t) = \frac{1}{2}\sum_{j\in S_t}\left[\sum_{n\in E}\frac{CCS(S_i,n)\,CCS(S_j|n)\,d(i,j)}{den(i,n)} + \sum_{n\in E}\frac{CCS(S_j,n)\,CCS(S_t|n)\,d(j,i)}{den(j,n)}\right] \quad (2.1)$$

Si on définit

$$\lambda^i_{nt} = \sum_{j\in S_t}\frac{CCS(S_j|n)\,d(i,j)}{den(i,n)} \quad (2.2)$$

et

$$\lambda'^i_{nt} = \sum_{j\in S_t}\frac{CCS(S_j,n)\,d(j,i)}{den(j,n)} \quad (2.3)$$

à l'aide de (2.2) et (2.3), (2.1) peut s'écrire

$$TWCCS(S_i|S_t) = \frac{1}{2}\left[\sum_{n\in E}CCS(S_i,n)\,\lambda^i_{nt} + \sum_{n\in E}CCS(S_i|n)\,\lambda'^i_{nt}\right]$$

mais puisque $CCS(S_i|n) = \sum_{j\in EAS_n}CCS(S_i,j)$

$$TWCCS(S_i|S_t) = \frac{1}{2}\left[\sum_{n\in E}CCS(S_i,n)\,\lambda^i_{nt} + \sum_{n\in E}\sum_{j\in EAS_n}CCS(S_i,j)\,\lambda'^i_{jt}\right]$$

$$TWCCS(S_i|S_t) = \frac{1}{2}\left[\sum_{n\in E}CCS(S_i,n)\left(\lambda^i_{nt} + \sum_{j\in EAS_n}\lambda'^i_{jt}\right)\right] \quad (2.4)$$

en définissant

$$\Lambda^i_{nt} = \lambda^i_{nt} + \sum_{j\in EAS_n}\lambda'^i_{jt} \quad (2.5)$$

(2.4) se réécrit

$$TWCCS(S_i|S_t) = \frac{1}{2}\sum_{n\in E}CCS(S_i,n)\Lambda^i_{nt} \quad (2.6)$$

Le trafic dans les deux sens entre S_i et le réseau reste un calcul qui dépend seulement du trafic intercommutateur de celui-ci. Pour calculer le nombre de liaisons, on utilise *NewRatio* qui est le trafic moyen par liaison et par technologie. Le nombre de liaisons entre S_i et les commutateurs de technologie t devient donc

$$\#liaisons(S_i|S_t) = \frac{TWCCS(S_i|S_t)}{NewRatio(S_i,t)} \tag{2.7}$$

$$\#total\ liaisons(S_i) = \sum_{t=1}^{3} \#liaisons(S_i|S_t)$$

La fonction Lambda (2.6) se comporte comme si la charge sur les liaisons dans le reste du réseau ne changeait pas durant la recherche binaire. Les coefficients λ^i_{nt} (2.2) ne sont pas affectés par le trafic de S_i puisque selon la définition de $d(i,j)$ et $den(i,n)$

$$den(i,n) = \sum_{j\in S} CCS(S_j|n)\, d(i,j) = \sum_{j\in S\setminus\{i\}} CCS(S_j|n)\, d(i,j) + CCS(S_i|n)\, d(i,i)$$

$$= \sum_{j\in S\setminus\{i\}} CCS(S_j|n)\, d(i,j) \quad (\text{puisque } d(i,i)=0)$$

Par contre les λ'^i_{nt} (2.3) sont affectés puisque $d(i,j) > 0$ pour tout $i \neq j$ et que $den(j,n)$ contient $CCS(S_i|n)$. Mais puisque $CCS(S_i|n)$ se retrouve au dénominateur et que tous les autres commutateurs participent à la somme de $den(j,n)$, on considère cette déviation comme négligeable. Nous discuterons au Chapitre 5 des conséquences de cette approche.

2.1.3 Caractéristiques de la fonction Lambda

Il est toujours intéressant de trouver des liens entre ce que l'on développe et ce que l'on veut remplacer. C'est le cas de la fonction Lambda. En effet, la fonction Lambda généralise l'approximation du noeud TRAFFICTRUNK. Le trafic, calculé par le noeud TRAFFICTRUNK, est égal au trafic de la fonction Lambda si tous les Λ^i_{nt} respectaient la contrainte suivante:

$$\sum_{t=1}^{3} \Lambda_{nt}^{i} = 2 \qquad (2.8)$$

c'est-à-dire

$$TWCCS(S_i)_{\text{TRAFFICTRUNK}} = \sum_{t=1}^{3} TWCCS(S_i|S_t) = \sum_{t=1}^{3} \left[\frac{1}{2} \sum_{n \in E} CCS(S_i,n)\, \Lambda_{nt}^{i} \right]$$

$$= \frac{1}{2} \sum_{n \in E} CCS(S_i,n) \sum_{t=1}^{3} \Lambda_{nt}^{i} = \sum_{n \in E} CCS(S_i,n)$$

On constate, qu'avec la contrainte (2.8), on retrouve le trafic intercommutateur total donné par TRAFFICTRUNK (voir explication p. 28). L'approximation de TRAFFICTRUNK n'est donc qu'un cas particulier de la fonction Lambda.

La fonction Lambda assure la convergence de l'algorithme d'approvisionnement du fait que le calcul de la charge maximale admissible reste *local*, c'est-à-dire ne dépend que du CP congestionné. En plus de cette caractéristique essentielle, la fonction Lambda

- **réagit en fonction du reste du réseau,**
  par exemple, si la charge ne change pas dans le réseau d'accès d'un commutateur, la charge sur ses liaisons généralement augmente puisque le reste du reséau croît. La fonction Lambda en tient compte contrairement à la fonction actuelle;
- **supprime les deux valeurs de trafic,**
  c'est-à-dire celle du côté gauche et celle du côté droit, il n'y en a qu'une seule qui est le trafic donné le MG;
- **permet de définir des points de congestion à tous les noeuds**
  de la base RCMB sans exception, le calcul de la CMA est exécuté dans une seule et même phase (voir les sous-sections 2.1.5 et 2.1.6).

2.1.4 Ancien calcul du nombre de liaisons

L'équation (2.7) n'a pas été la première utilisée afin de trouver le nombre de liaisons à partir du trafic de l'équation (2.6). L'équation (2.7) remplace une première méthode plus complexe qui n'apportait pas plus de précision aux résultats obtenus. L'ancienne méthode se basait sur la convexité de la fonction qui, à partir du trafic entre deux commutateurs, détermine le nombre de liaisons nécessaire afin d'assurer une probabilité de blocage prédéterminée.

Appelons la fonction qui transforme le trafic en nombre de liaisons $f(x)$. À partir du nombre total de liaisons T, calculé par le module Trunker pour un commutateur S_i, on calculait un "pseudo nombre" de commutateur $n \in R^1$ de technologie t qui satisfaisait l'équation

$$\left\lceil \left\{ n \cdot f\left[\frac{Trafic}{n} \right] \right\} \right\rceil = T$$

où *Trafic* est le total du trafic bidirectionnel entre le commutateur S_i et les commutateurs de technologie t du réseau. n était appelé pseudo-nombre de commutateur puisque c'était un nombre proche du nombre de commutateur, n étant égal au nombre de commutateurs du réseau si tous les commutateurs avaient le même trafic intercommutateur avec S_i. Ce nombre existe toujours à cause de la convexité de $f(x)$. On trouvait n à l'aide d'une recherche binaire. La borne supérieure pour la recherche était le nombre de commutateurs de technologie t dans le réseau et la borne inférieure, la moitié de cette valeur. Puisque le nombre de commutateur dans le réseau est $|S|$, il fallait donc éxécuter 3 fois $|S|$ recherches binaires à chaque fois que les coefficients de la fonction Lambda était remis à jour. La seule opération nécessaire pour obtenir *NewRatio* est de calculer la moyenne de trafic par liaisons. La méthode utilisant *NewRatio* réduit et simplifie le calcul et ce, pour les mêmes résultats.

2.1.5 Le noeud LAMBDA

On sait que la fonction utilisée pour traiter un noeud de la base RCMB n'est qu'un attribut de celui. Il est donc possible de le changer. Lorsque que le module Installer remet à jour la quantité de ressource à chacun des noeuds (à l'aide de I$OutSi, un des sous-programme qui compose le module Installer), le choix de la fonction est aiguillé par cet attribut. La base RCMB possède maintenant deux fonctions: la fonction STANDARD et la fonction LAMBDA[1]. Les noeuds traités par la fonction LAMBDA, c'est-à-dire la fonction précédemment décrite, sont appelés noeuds LAMBDA (voir Figure 12). Ce nouveau noeud diffère du noeud STANDARD de plusieurs façons.

Premièrement, l'information présente à l'entrée du noeud, ici le trafic provenant de TRAFFICTRUNK, n'est pas utilisée. Tous les autres noeuds, qui ne sont pas des noeuds principaux d'entrée, déterminent leur sortie à partir de l'information d'entrée. Cette information n'est pas utilisée parce que le trafic conduit par l'arc est le trafic intercommutateur total, alors que la fonction Lambda nécessite le trafic intercommutateur par échange ($CCS(S_i,n)$).

Deuxièmement, deux pointeurs sont passés au sous-programme I$OutSi pour le calcul de la fonction LAMBDA, soit:

- un pointeur à la structure de données du commutateur pour avoir accès au trafic intercommutateur par échange $CCS(S_i,n)$,
- un pointeur à la structure de données LAMBDA qui contient les coefficients Λ_{nt}^{i} et *NewRatio*.

1. On utilise LAMBDA pour distinguer la fonction Lambda comme telle de son implantation dans SNAP.

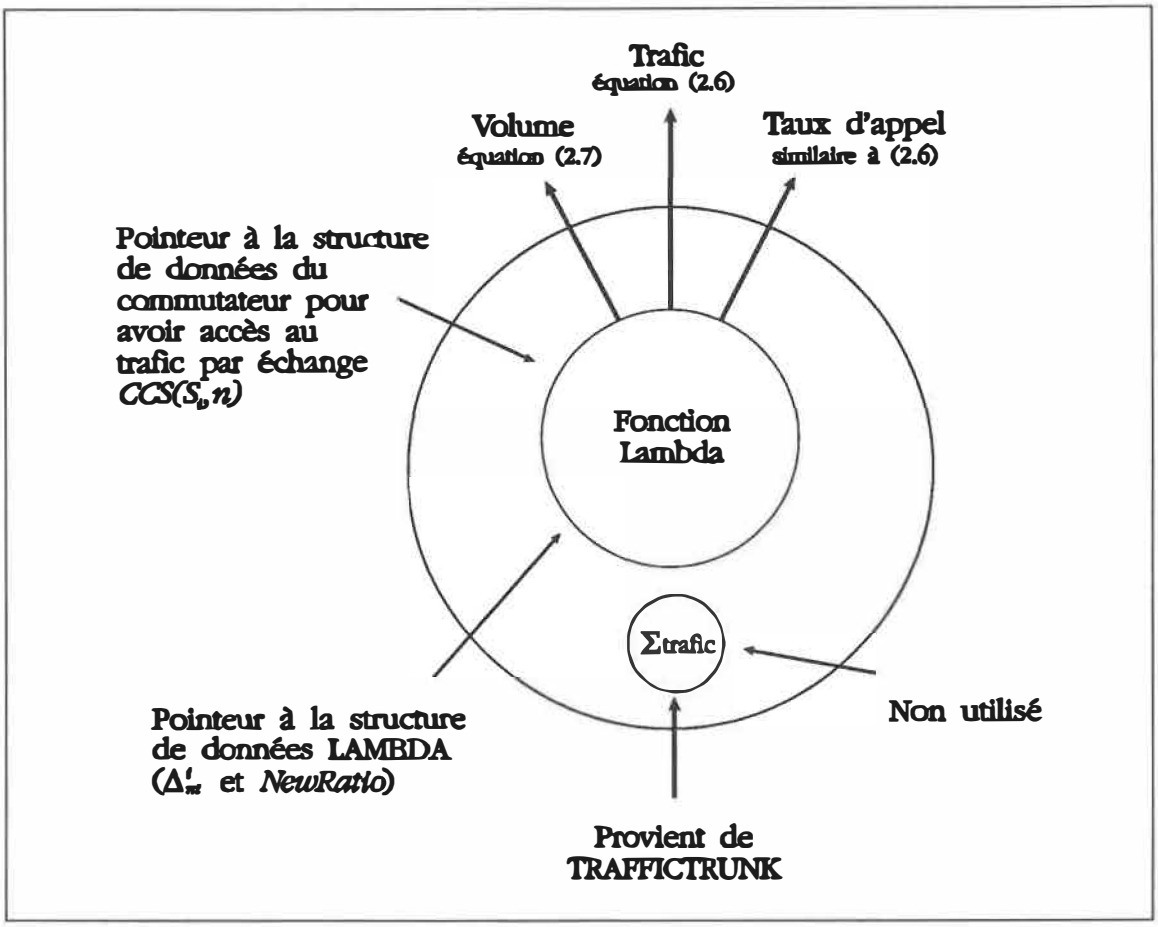


Figure 12. **Noeud LAMBDA**

Ce changement est important puisqu'avant, aucune information spécifique à un commutateur n'était passée comme paramètre pour le calcul des ressources. Avec le noeud LAMBDA, le même trafic intercommutateur peut générer un nombre différents de liaisons. Puisque la structure de données LAMBDA est remise à jour par SNAP (explication à la sous-section suivante), ces noeuds disposent d'une fonction qui varie dans le temps, selon le réseau. Tous les noeuds STANDARD utilisent des tables de capacité (qu'on peut considérer comme des fonctions) qui ne changent pas de toute la durée de l'étude.

De plus, lorsque le premier noeud LAMBDA d'un commutateur est traité par le sous-programme I$OutSi, il calcule sa sortie et la sortie de tous autres noeuds LAMBDA du même commutateur. Un drapeau de la structure de données LAMBDA indique si oui ou non la sortie est déjà calculée. Ce changement optimise le calcul puisque lorsque le trafic $TWCCS(S_i|S_t)$ est disponible, la quantité de tous les types de liaisons est donné par l'équation (2.7). On évite alors de recalculer ce trafic à chacun des noeuds d'un même CP.

Le changement le plus important apporté par le noeud LAMBDA est illustré à la Figure 13: la dichotomie gauche/droite n'existe plus. On a vu précédemment que le trafic présent à l'entrée du noeud LAMBDA n'est pas utilisé, d'où l'inutilité du noeud TRAFFICTRUNK. Le noeud TRAFFICTRUNK n'a pas été retiré pour des raisons de compatibilité avec l'ancienne version du programme. Les noeuds LAMBDA et le noeud STANDARD de rattachement sont rassemblés dans la configuration ZETA. L'emploi de la configuration ZETA permet de ne définir qu'une seule fois cet ensemble et de le rappeler pour chaque type de CP modélisés par la base RCMB. On observe que toutes les ressources qui était du côté liaison, à la Figure 4 de la page 15, se retrouvent du même côté que les ressources du côté ligne. Il faut noter que la dichotomie ligne/liaison n'existe qu'au niveau de SNAP: aucune information n'est contenue dans la base RCMB à savoir si un noeud est situé d'un côté plutôt que de l'autre. Les noeuds LAMBDA remplacent les noeuds d'entrée précédemment du côté droit.

Un autre changement apporté, plus technique celui-là, c'est que les noeuds LAMBDA disposent d'un mécanisme spécial de mise à jour. Pour optimiser le calcul des ressources, le module Installer ne rafraîchit pas la sortie d'un noeud et de ses fils si l'entrée du noeud n'a pas changée depuis la dernière itération. Ce critère est tout à fait en accord avec la modélisation utilisant des tables fixes de la fonction STANDARD. Par contre, les noeuds LAMBDA doivent être tous rafraîchit lorsque les

coefficients Λ_{nt}^{i} sont remis à jour, même si le trafic intercommutateur (l'entrée du noeud) n'a pas changé. La structure de données LAMBDA contient un drapeau, pour chaque noeud LAMBDA de chaque CP, qui indique si oui ou non le noeud (et ses descendants) doit être rafraîchi. Afin de déterminer si le noeud doit être remis à jour, on exécute un OU logique entre ce drapeau et l'ancienne condition.

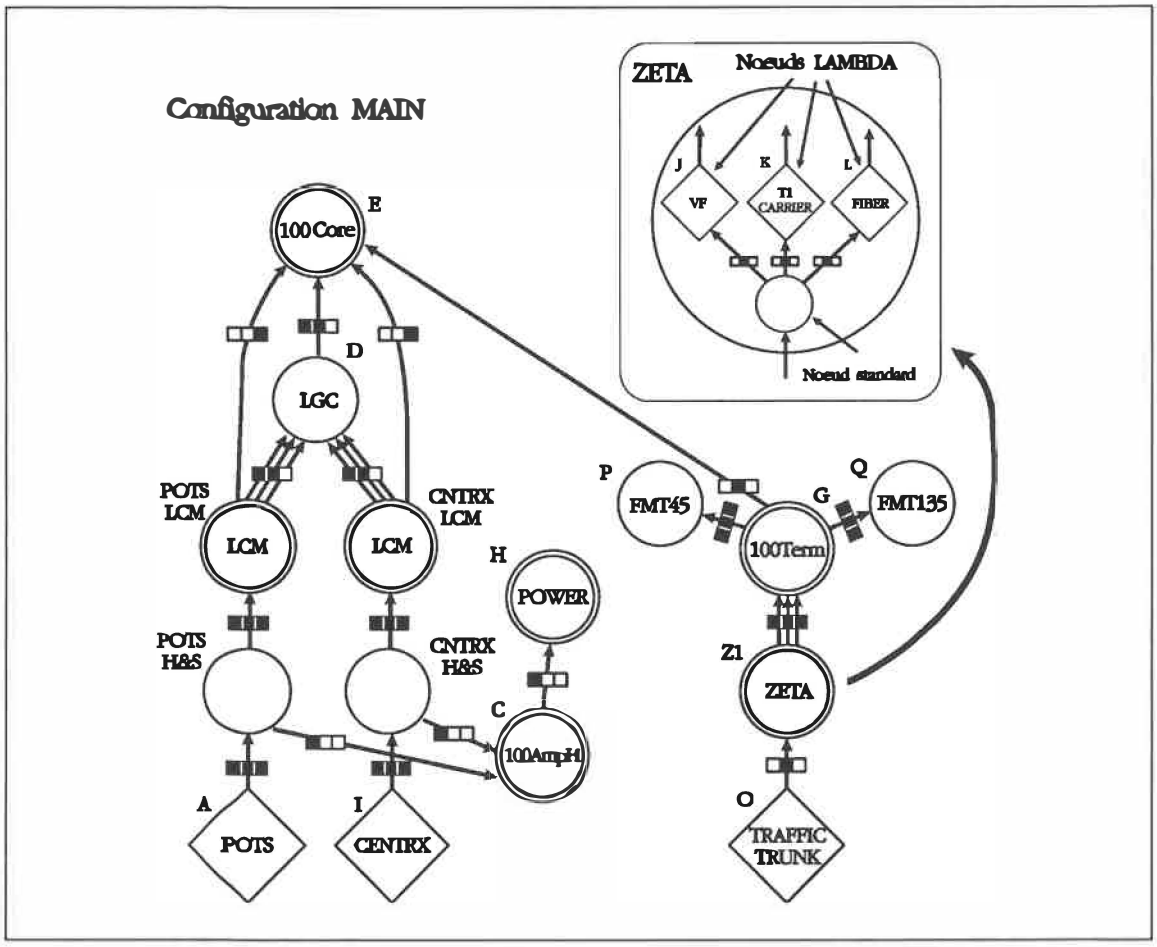


Figure 13. **La base RCMB avec le noeud LAMBDA**

2.1.6 Nouvel organigramme fonctionnel

Le changement apporté à l'algorithme d'approvisionnement se situe dans la boucle Compressor, illustrée à la Figure 14. Juste après que le trafic intercommutateur soit connu (calculé par Trafficker), on vérifie si c'est la première fois que Compressor est appelé pendant l'année ou si il y a eu fermeture ou ouverture de commutateurs depuis le dernier appel.

Le module Trunker, qui est modifié afin de calculer tous les paramètres nécessaires aux noeuds LAMBDA, est appelé la première fois dans l'année pour initialiser la structure de données LAMBDA. Si le module Assigner à fermé ou ouvert des commutateurs, on sait que le module a réassigné certaines lignes à travers le réseau. Tous les coefficients Λ_{nt}^{i} doivent donc être remis à jour.

Puisque le rafraîchissement des coefficients de la fonction Lambda survient juste avant l'appel du module Compressor, le trafic calculé par la fonction Lambda, lors du premier appel du module Installer par Compressor, est exactement le même que celui donné par le MG. À partir de ce moment, les coefficient sont "gelés": le trafic donné par la fonction Lambda *n'est plus le même* que celui du MG s'il y a congestion. Les commutateurs sont maintement isolés par le calcul de la fonction Lambda. La fonction Lambda joue alors le même rôle que la fonction du noeud TRAFFICTRUNK, c'est-à-dire isoler les CP. Comme mentionné à la section 1.2.2, cet isolement est nécessaire à la convergence de l'algorithme d'approvisionnement de SNAP. À la Figure 14 (p. 53), on constate que le nouvel organigramme ne possède plus de côté droit et que le module de transition Trunker se retrouve maintenant intégré à la boucle du module Compressor; il s'agit d'un réarrangement des modules originaux (voir Figure 10, p. 27).

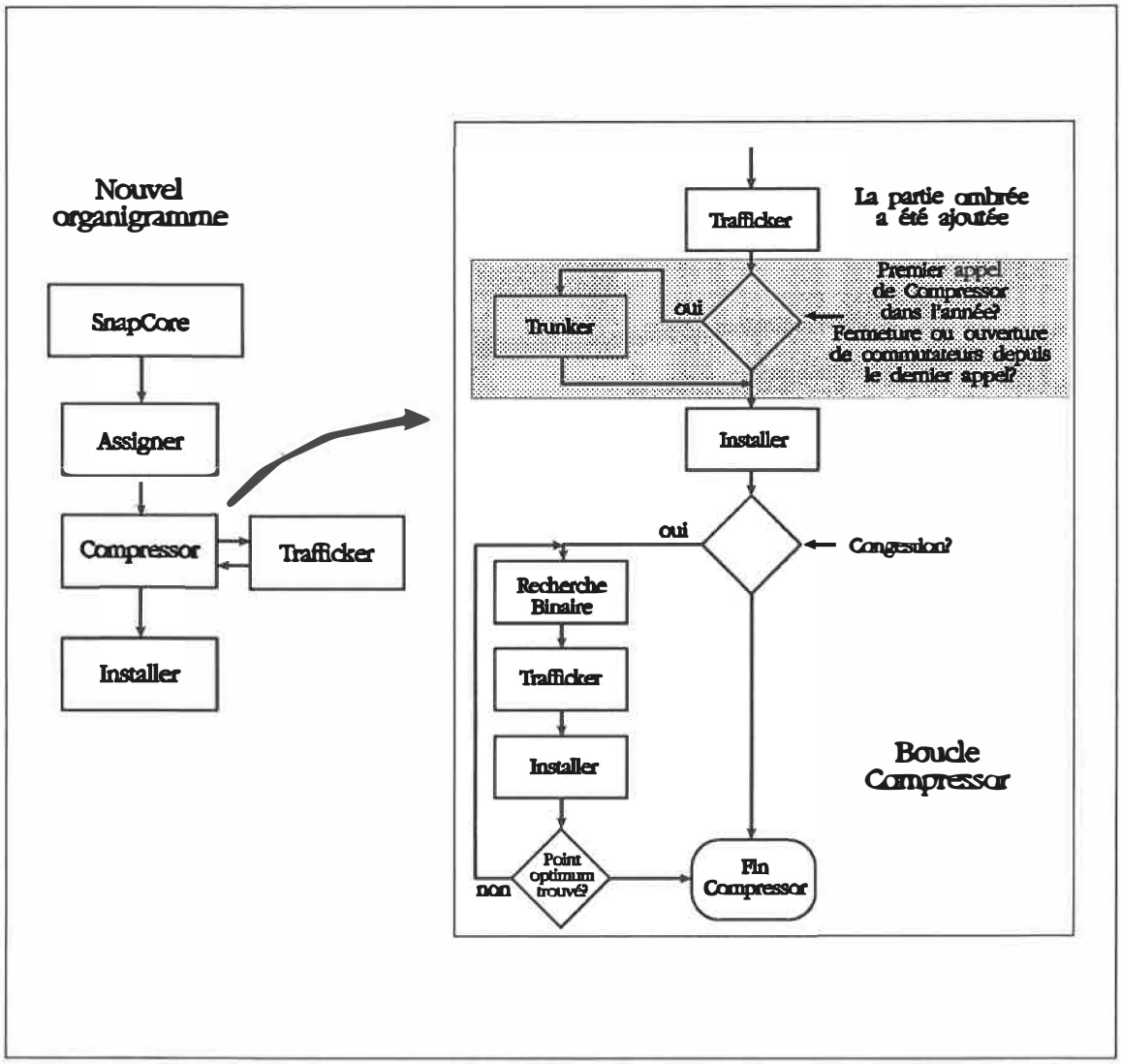


Figure 14. **Nouvel organigramme fonctionnel**

2.2. Dimensionneur de réseaux de liaisons

Le nouveau DRL, appelé DRL/Lambda, est l'aboutissement d'un processus comportant plusieurs étapes. Chaque sous-section de la présente section traite d'une étape. Voici la liste des étapes:

- élaborer les hypothèses de travail,
- trouver la procédure d'optimisation du réseau de liaisons,
- déterminer les données nécessaires,
- corriger la procédure précédente à la modularité des liens,
- bâtir l'algorithme DRL,
- trouver la modélisation du tandem dans la base RCMB,
- changer l'algorithme d'approvisionnement.

Les trois premières étapes ne forment pas une séquence. Elles se sont déroulées simultanément afin de produire un algorithme compact qui permet l'obtention de résultats significatifs, avec le minimum de données de la part du planificateur (relativement à un vrai DRL). La quatrième section explique la méthode qui a été développée pour corriger la procédure d'optimisation retenue. La construction du DRL intègre, sous forme d'algorithme, la procédure d'optimisation et la procédure corrective. Enfin les deux dernières sous-sections décrivent la modélisations des tandems dans la base RCMB et les changements apportés à l'algorithme d'approvisionnement de SNAP. La procédure corrective et la modification de la base RCMB représentent les aspects les plus novateurs.

2.2.1 Hypothèses de travail

Les hypothèses de travail retenues sont basées sur une étude des éléments suivants: les DRL existants, l'architecture des réseaux locaux ainsi que les données disponibles dans SNAP. Certaines de ces hypothèses ont servi à simplifier le nouveau DRL, le but étant de voir si cette approche est viable et de l'améliorer si c'est le cas. Elles peuvent être facilement laissées de côté si nécessaire. Ces hypothèses seront identifiées par un (*). Voici la liste de ces hypothèses:

- ***les liaisons sont de trois types:***
  direct final (DF), primaire à haut usage (PHU) et final; les DRL existants modélisent aussi les liens de type IHU. L'adoption de cette hypothèse nous évite l'utilisation des tables de routage qui sont généralement difficilement accessibles (du moins au stage de la planification). Puisque qu'il n'y a pas de lien IHU, chaque groupe PHU entre deux commutateur "déborde" vers le groupe final,
- ***les tandems sont entièrement interconnectés (*):***
  cette hypothèse permet de simplifier le nouveau DRL et évite le recours au table de routage. Cette hypothèse est aussi justifiée par l'architecture des réseaux locaux actuels (on se retrouve avec l'ancien réseau logique à un dégré plus haut dans la hiérarchie),
- ***tout le trafic provenant du réseau d'accès est de Poisson (*):***
  c'est la théorie utilisée par tous les DRL.

Les deux premières hypothèses nous évitent le recours aux tables de routage tout en gardant les types de liens importants comme les lien finaux et PHU. Il est bon de rappeler ici, comme mentionné à la section 1.1.2, que SNAP est un outil de planification et non d'approvisionnement. Le but recherché est de pouvoir effectuer des comparaisons valides entre deux scénarios, non pas remplacer des procédures ou des outils spécialisés, utilisés dans l'approvisionnement des ressources.

2.2.2 Procédure d'optimisation

La procedure d'optimisation choisie est appelée CCS économique, ou ECCS (Economic CCS). C'est la procédure standard en usage dans les compagnies de télécommunications. Le principe derrière ECCS est le suivant: ajouter des liaisons au lien HU tant que le coût par unité de trafic le justifie. On sait que chaque commutateur du réseau public de télécommunications est connecté par un lien final à un tandem; il est donc toujours possible d'acheminer un appel via le réseau formé des seuls liens finaux. La Figure 15 illustre le cas où un commutateur C1 et C2 ont un lien final au même tandem T. L'explication qui suit est tiré de *Enginering and Operations in the Bell System* (AT&T Bell Laboratories, 1984).

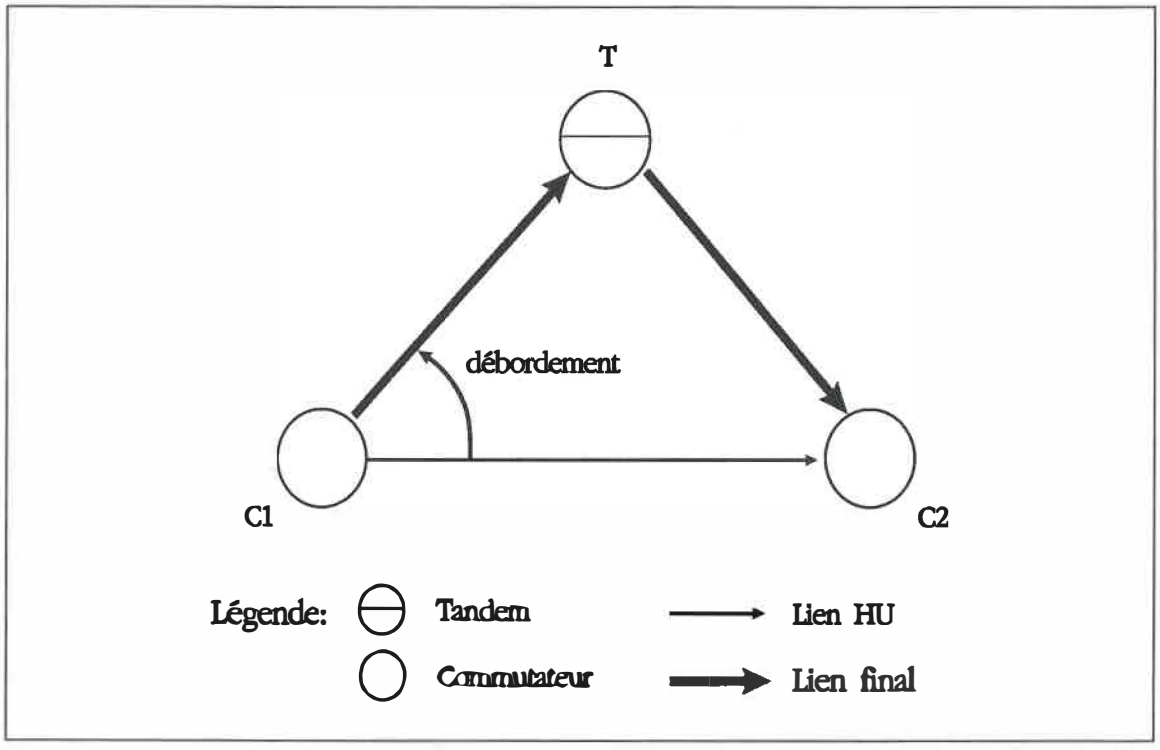


Figure 15. **Triangle route directe-finale**

Lorsque le trafic entre C1 et C2 est important, il y a avantage à installer un lien direct entre les deux. Puisque ce lien est souvent plus court que le lien via le tandem, il est permis de penser que la solution la plus économique consiste à n'utiliser que ce lien pour tout le trafic de C1 vers C2. Si c'était le cas, le lien entre C1 et C2 devrait être approvisionné pour rencontrer l'objectif de blocage, ce qui requiert un nombre élevé de liaisons. De plus, en cas de défaillance majeure du lien direct, il n'y aurait aucun moyen d'acheminer l'appel par une autre route. On utilisera le réseau de la Figure 15, avec les liens PHU et finaux, pour élaborer la théorie qui suit. Notons que ce type de réseau donne un meilleur service à moindre coût que les stratégies route directe seulement ou via tandem seulement.

Le modèle utilisé, pour determiner le nombre de liaisons du lien HU, est basé sur le triangle de la Figure 15. On suppose que l'on connaît le coût par liaison pour la route finale, C_A (qui inclut le coût de commutation via le tandem plus le coût des ressources de transmission) et le coût par liaison pour le lien direct HU C_D. La capacité incrémentale d'un lien, conçu pour un blocage donné, est défini comme étant la charge supplémentaire qui peut être offerte au lien, sans changer la probabilité de blocage, lorsqu'une liaison est rajoutée. Puisque les liens C1-T et T-C2 servent à d'autres trafics que le trafic C1-C2 (le trafic de C1 vers les autres commutateurs transite via C1-T et le trafic provenant d'autres commutateurs vers C2 transite par T-C2), ils contiennent beaucoup de liaisons, donc sont plus efficaces. Leur capacité incrémentale est conséquemment plus grande. On assume que les deux liens finaux ont la même capacité incrémentale γ_A.

Si n liaisons sont installées dans le lien C1-C2, le coût total de ces liaisons est $n \cdot C_D$. De plus, le trafic de débordement de ce groupe, $\alpha(n)$, va transiter par le lien final qui va nécessiter $\alpha(n)/\gamma_A$ liaisons de plus pour maintenir le blocage au même niveau. Le coût incrémental de l'installation de n liaisons dans le lien HU C1-C2, $C(n)$, est donné par

$$C(n) = n \cdot C_D + \frac{\alpha(n)}{\gamma_A} \cdot C_A$$

La Figure 16 montre $C(n)$ et ses deux composantes, lorsque l'on traite n comme une variable continue. $C(n)$ possède un minimum unique tel que

$$-\frac{\partial \alpha(n)}{\partial n} = \gamma_A \cdot \frac{C_D}{C_A} = \frac{\gamma_A}{C_R} \qquad (2.9)$$

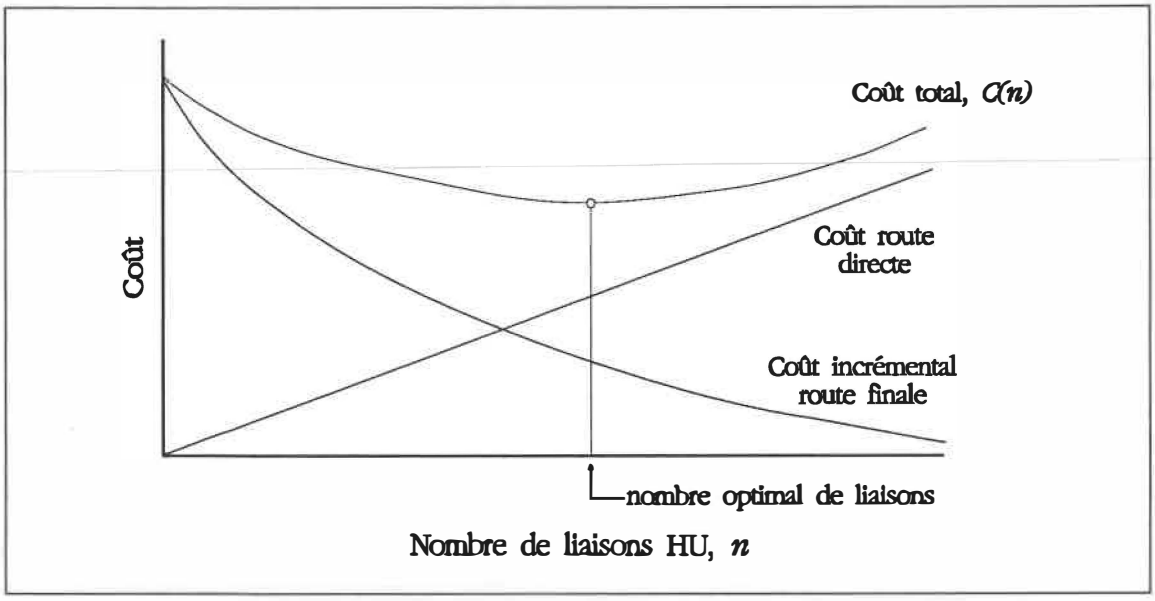


Figure 16. Coût total du réseau route directe-finale

où $C_R = C_A/C_D$. C_R est appelé ratio des coûts du lien HU (s'il n'y a aucune valeur pour laquelle l'équation est vérifiée, le minimum est situé à $n=0$, c'est-à-dire que la stratégie via tandem seulement est la plus économique). La quantité $-\partial\alpha(n)/\partial n$ possède une interprétation pratique: c'est le trafic qui transite par la dernière liaison du lien HU, en assumant que le modèle Erlang B[1] s'applique à ce lien. La quantité γ_A/C_R est donc la charge la plus économique pour la dernière liaison et le lien optimal est celui pour lequel le trafic transitant via la dernière liaison est γ_A/C_R CCS. Cette quantité est appelée CCS économique, d'où le nom de la procédure.

1. On peut consulter les ouvrages traitant des réseaux de télécommunications, listés dans la bibliographie, pour plus de renseignements sur les modèles de trafic.

2.2.3 Données nécessaires

Les hypothèses de travail et la procédure d'optimisation permettent de réduire substanciellement la quantité d'information nécessaire, comparativement à un vrai DRL. Voici la liste des nouvelles données requises de la part du planificateur:

- **Table de connexion,**
  cette matrice (symétrique) indique le type de lien existant ou prévu entre deux commutateurs. Les types possibles de connexion sont:
  - **Direct Final (DF),**
    le lien entre les deux commutateurs est dédié et le trafic ne déborde pas sur aucun autre lien,
  - **Jamais de connexion (PNC: Permanently Not Connected),**
    aucune liaison ne peut être installée entre cette paire, tout le trafic est offert au lien final via le tandem,
  - **Pas de connexion (TNC: Temporarily Not Connected),**
    aucune liaison mais le DRL peut installer un lien PHU entre les deux s'il est économiquement justifié,
  - **Primaire à haut usage (PHU),**
    à l'initialisation, tous les liens PHU sont défini comme TNC, l'algorithme DRL décide si oui ou non le lien est justifié et si oui, combien de liaisons il doit contenir,
- **Matrice des coûts,**
  le coût par type de liaison par unité de distance, le coût des ressources de transmission par type de liaison par type de commutateur. Ces données sont nécessaires à la procédure ECCS,
- **Matrice des technologies de croissance,**
  pour chaque paire de type de commutateur, cette matrice désigne quel type de liaison il faut installer si de nouvelles liaisons sont requises,
- **Modularité des liens,**
  par technologie de liaison.

2.2.4 Correction de procédure

Nous avons vu que la procédure ECCS suppose une petite modularité pour les liaisons rajoutées (à cause de la dérivée). La plupart des systèmes de liaisons présentement installés sont de type T1 (avec fils de cuivre ou fibre optique) qui possèdent une modularité de 24 liaisons. Pour corriger de façon simple ce problème, on peut imaginer la procédure suivante:

- installer le nombre minimal de modules entre les commutateurs de façon à couvrir le nombre de liaisons suggéré par la procédure ECCS;
- pour chaque dernier module installé entre une paire de commutateurs, distribuer les liaisons qui ne sont pas utilisées selon la moyenne du trafic de chaque direction.

Voici un exemple, à l'aide du réseau de la Figure 17, de l'application d'une telle procédure. On définit le format suivant pour les données relatives au réseau:

$$(C_i,C_j) \rightarrow (g_{ij},g_{ji},TDL)$$

où

$$g_{ij} = \text{nombre de liaisons à ajouter selon la méthode ECCS,}$$
$$TDL = \text{trafic de la dernière liaison,}$$

et

$$(C_i,T) \rightarrow (g_{iT},g_{Ti})$$

avec une capacité incrémentale de γ CCS constante pour tout les liens finaux (habituellement de 26 CCS pour les réseaux locaux urbains). g_{iT} et g_{Ti} sont déterminés à partir du trafic direct offert au lien final plus le minimum du trafic de débordement, c'est-à-dire le trafic de débordement qui provient des liens PHU si tous les "derniers" modules étaient installés.

Si l'on dispose des données suivantes

(C1,C2) → (6,7,15)
(C1,C3) → (6,6,20)
(C2,C3) → (12,8,10)

et

(C1,T) → (6,4)
(C2,T) → (6,10)
(C3,T) → (5,6)

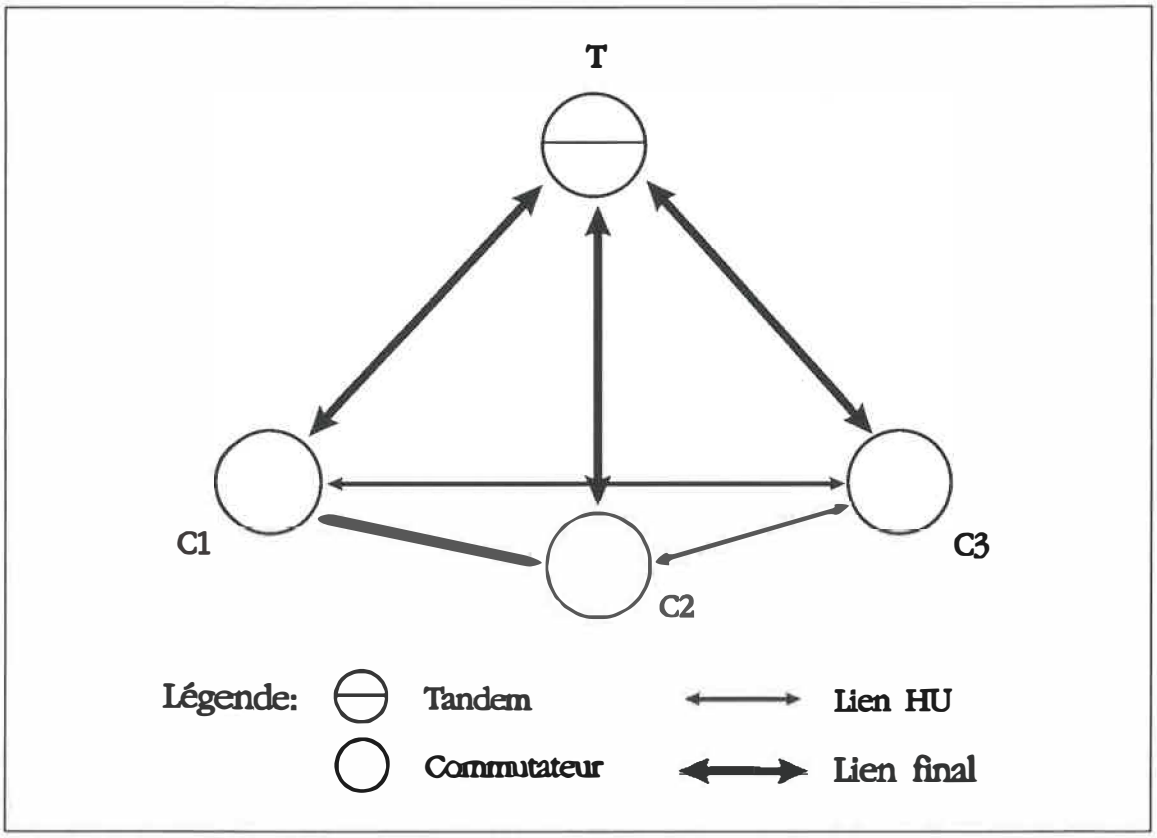


Figure 17. **Réseau du problème d'allocation des liaisons**

selon la procédure précédemment décrite, on devrait installer 6 modules (de type T1, modularité de 24 liaisons) et diviser les liaisons non utilisées selon la proportion de trafic de chaque direction. Par exemple si Trafic(C1,C2) = 500 CSS (le trafic de C1 vers C2) et Trafic(C2,C1) = 600 CCS, les 11 liaisons vacantes du module C1-C2 sont divisée comme suit:

Trafic Total = 1100 CCS
Modularité du lien T1 = 24
Liaisons à installer: 6 (C1 vers C2) + 7 (C2 vers C1)
Liaisons non utilisées: 24 - 6 - 7 = 11
Liaison à ajouter pour le lien C1 vers C2:

$$\frac{\text{Trafic(C1,C2)}}{\text{TraficTotal}} \cdot 11 \text{ liaisons} = \frac{500}{1100} \cdot 11 = 5$$

Liaisons de C2 vers C1: 11 - 5 = 6

Par contre, on peut utiliser le fait que les liens finaux sont plus efficaces, on peut alors réduire le nombre de modules à installer et avoir le même rendement. Avec l'hypothèse que la capacité incrémentale des liens finaux est la même pour tous les liens est qu'elle égale $\gamma = 26$ CCS, on peut convertir les liens PHU en liens finaux à l'aide de la formule

$$\left\lceil \left[\frac{TDL}{\gamma} \cdot l \right] \right\rceil \qquad (2.10)$$

où l est le nombre de liaisons que l'on veut transférer du lien PHU au lien final. La formule (2.10) nous donne le nombre de liaisons requises que l'on doit installer dans le lien final pour remplacer l liaisons du lien PHU avec un trafic moyen de *TDL* CCS.

Avec le problème décrit plus haut, l'application de cette formule nous donne

$$(C1,C2) \rightarrow \lceil 13.8 = 14$$
$$(C1,C3) \rightarrow \lceil 18.5 = 19$$
$$(C2,C3) \rightarrow \lceil 9.2 = 10$$

Les liaisons libres, dans les liens finaux, sont au nombre de

$$(C1,T) \rightarrow 14$$
$$(C2,T) \rightarrow 8$$
$$(C3,T) \rightarrow 13$$

On constate que l'on peut donc éliminer l'installation des liens PHU (C1,C2) et (C2,C3). Cette solution à l'avantage de comporter deux modules de moins que la procédure simple décrite plus haut et ce pour le même rendement.

Dans le but d'éviter les modules superflus, un programme en nombre entier optimise le réseau suggéré par la procédure ECCS, dans le sens décrit plus haut. Pour pouvoir formuler mathématiquement le problème, les hypothèses suivantes sont supposées valides:

- ***seul le dernier module d'un lien PHU fait l'objet de la décision,*** à savoir si oui ou non on l'installe dans le réseau,
- ***un nombre suffisant de modules pour le lien final est prévu,*** ce nombre doit couvrir le nombre de liaisons dans le cas où tous les derniers modules PHU du commutateur ne sont pas installés.

La formule (2.10) se base sur l'hypothèse que le lien final possède une capacité incrémentale de 26 CCS, ce qui n'est pas toujours le cas, par exemple, lorsqu'un commutateur ne possède qu'une petite charge. Nous verrons plus loin comment éviter le recours à cette hypothèse. Voici les définitions nécessaires à l'élaboration du modèle.

Définitions

x_{ij}	décision si oui ou non le dernier module, du lien de type PHU, doit être installé entre les commutateurs S_i et S_j ($x_{ij} \in B^1$). À partir de cette définition $x_{ij} = x_{ji}$,
m_{ij}	modularité du lien entre S_i et S_j,
y_{iq}	décision si oui ou non le $q^{ième}$ module du groupe final du commutateur S_i doit être installé ($y_{iq} \in B^1$),
m_{iq}	modularité du lien associé à y_{iq} ,
g_{ij}	nombre de liaisons requis de S_i vers S_j,
h_{ij}	nombre de liaisons supplémentaires requises dans le lien final de S_i si le module PHU entre S_i et S_j n'est pas installé,
d_{ij}	coût du module PHU entre S_i et S_j,
e_{iq}	coût du $q^{ième}$ module du lien final de S_i,
PHU_i	$PHU_i = \{ j \mid S_j$ a un lien PHU avec $S_i \}$,
$T(i)$	index du tandem auquel S_i est rattaché.

La Figure 18 (p. 65) illustre le réseau qui sert de modèle à l'élaboration des formules qui suivent. Premièrement, $g_{iT(i)}, g_{T(i)i}, g_{jT(j)}$ et $g_{T(j)j}$ sont déterminés par le trafic direct, offert à chacun des liens finaux, plus le minimum de trafic de débordement du lien PHU, c'est-à-dire quand le dernier module PHU est installé. Pour chacun des deux liens finaux, on installe tous les modules possibles de façon à ce que l'on obtienne

$$g_{kT(k)} + g_{T(k)k} < m_k \text{ pour } k = i, j \qquad (2.11)$$

où m_k est la modularité du lien final. L'inégalité (2.11) signifie que l'on installe toutes les liaisons possibles jusqu'à ce qu'il ne reste pas assez de liaisons pour combler un autre module. Si le module PHU est installé, on doit satisfaire les inégalités suivantes

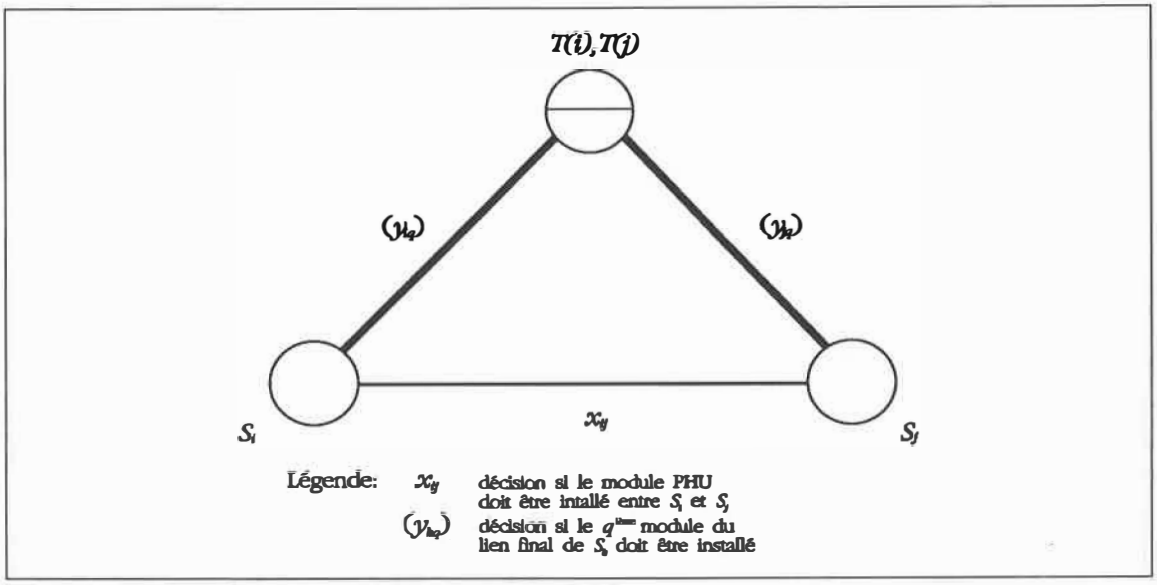


Figure 18. Réseau modèle

$$m_{k1}\, y_{k1} \geq g_{kT(k)} + g_{T(k)k} \quad \text{pour } k = i, j \tag{2.12}$$

L'inégalité (2.12) signifie que l'on doit disposer d'un nombre suffisant de liaisons dans le lien final ($y_{k1} = 0$ ssi $g_{kT(k)} + g_{T(k)k} = 0$). Si par contre l'on décide de ne pas installer le module PHU, on doit connaître le nombre supplémentaire de liaisons à installer dans chacun des liens finaux. Afin d'éviter le recours à l'hypothèse d'une capacité incrémentale de 26 CCS pour le lien final, on effectue la procédure suivante:

- on stocke le nombre total de liaisons requises dans chacun des lien finaux dans les variables r_i et r_j,
- on calcule le trafic de débordement si les g_{ij} et g_{ji} liaisons n'étaient pas installées,
- on calcule le nombre total de liaisons requises dans chacun des liens finaux avec le nouveau trafic de débordement, on stocke les résultats dans t_i et t_j,
- le nombre de liaisons supplémentaires des liens finaux de S_i et S_j sont respectivement $b_{ij} = t_i - r_i$ et $b_{ji} = t_j - r_j$.

Donc, si le module PHU n'est pas installé, on doit alors respecter les inégalités

$$\sum_{q=1}^{p_i} m_{iq} y_{iq} \geq h_{ij} + g_{iT(i)} + g_{T(i)i} = M_i \qquad (2.13)$$

$$\sum_{q=1}^{p_j} m_{jq} y_{jq} \geq h_{ji} + g_{jT(j)} + g_{T(j)j} = M_j \qquad (2.14)$$

où p_k est déterminé de façon à ce que

$$\sum_{q=1}^{p_k} m_{kq} \geq M_k > \sum_{q=1}^{p_k-1} m_{kq} \qquad (2.15)$$

Les inégalités (2.13) et (2.14) signifient que les liens finaux doivent comporter assez de liaisons pour absorber le trafic provenant du lien PHU non installé. L'inégalité (2.15) indique que p_k est choisi de façon à ce qu'un nombre suffisant de modules finaux soit prévu. Les inégalités (2.12) et (2.13) peuvent être réunies en une seule avec l'introduction de la variable binaire x_{ij}. Pour S_i on obtient

$$\sum_{q=1}^{p_i} m_{iq} y_{iq} + h_{ij} x_{ij} \geq h_{ij} + g_{iT(i)} + g_{T(i)i} = M_i$$

De façon similaire pour S_j

$$\sum_{q=1}^{p_j} m_{jq} y_{jq} + h_{ji} x_{ji} \geq h_{ji} + g_{jT(j)} + g_{T(j)j} = M_j$$

Mais les commutateur S_i et S_j possèdent des liens PHU avec d'autres commutateurs du réseau. Par exemple, S_i possède des liens PHU avec tous les S_j tels que $j \in PHU_i$. Les inégalités généralisées sont

$$\sum_{q=1}^{p_i} m_{iq} y_{iq} + \sum_{j \in PHU_i} h_{ij} x_{ij} \geq \sum_{j \in PHUi} h_{ij} + g_{iT(i)} + g_{T(i)i} = M_i \quad \text{pour tout } i \in S \qquad (2.16)$$

où S est l'ensemble des index de tous les commutateurs. Mais puisque $x_{ij} = x_{ji}$, on défini z_{ij} comme étant

$$z_{ij} = \begin{cases} x_{ij} & \text{si } i<j \\ x_{ji} & \text{si } i>j \end{cases}$$

Avec z_{ij}, les inégalités (2.16) deviennent

$$\sum_{q=1}^{p_i} m_{iq} y_{iq} + \sum_{j \in PHU_i} b_{ij} z_{ij} \geq \sum_{j \in PHUi} b_{ij} + g_{iT(i)} + g_{T(i)i} = M_i \quad \text{pour tout } i \in S \quad (2.17)$$

Les inégalités (2.17) sont appelées contraintes d'existence de trafic. Ces inégalités garantissent qu'un nombre suffisant de liaisons sont installées dans les liens finaux qu'importe les décisions prises pour les liens PHU. Le coût total des modules (incluant les liaisons), selon les décisions y_{iq} et z_{ij} est donné par

$$\sum_{i \in S} \left[\sum_{q=1}^{p_i} e_{iq} y_{iq} + \sum_{\substack{j \in PHU_i \\ j>i}} f_{ij} z_{ij} \right]$$

Si l'on fait la synthèse de ce qui a été développé plus haut, le problème à résoudre pour minimiser l'introduction des ressources dans le réseau de liaisons est

$$\min \sum_{i \in S} \left[\sum_{q=1}^{p_i} e_{iq} y_{iq} + \sum_{\substack{j \in PHU_i \\ j>i}} f_{ij} z_{ij} \right]$$

sous les contraintes

$$\sum_{q=1}^{p_i} m_{iq} y_{iq} + \sum_{j \in PHU_i} b_{ij} z_{ij} \geq M_i \quad \text{pour tout } i \in S$$

$$\text{où } M_i = \sum_{j \in PHU_i} b_{ij} + g_{iT(i)} + g_{T(i)i}$$

$$\text{et } p_i \rightarrow \sum_{q=1}^{p_i} m_{iq} \geq M_i > \sum_{q=1}^{p_i - 1} m_{iq}$$

Ce type de problème fait partie de la classe générale des programmes en nombres entiers. Plus spécifiquement, ce problème est un programme en variables binaires pur (pur dans le sens où toute les variables sont 0-1 sans exception) avec contraintes linéaires. Dans le reste du mémoire, on fera référence à ce type problème sous le nom de BIP (Binary Integer Program). Nous verrons au Chapitre 3 comment ce problème est résolu numériquement.

La procédure qui détermine les b_{ij} et la formulation du BIP ci-haut supposent que les trafics de débordement des liens PHU n'interagissent pas entre eux. Cette hypothèse ne tient pas compte du fait que le trafic moyen par liaison augmente avec le trafic. Par exemple, si les modules PHU entre S_i-S_j et S_i-S_k ne sont pas installés, le nombre réel de liaisons supplémentaires pour le lien final de S_i, w_i, respecte l'inégalité

$$w_i \leq b_{ij} + b_{ik}$$

Cet écart entre le nombre de liaisons prévu et le nombre nécessaire est un paramètre à évaluer dans les résultats, ceci afin de savoir si l'hypothèse est justifiée (une présentation partielle des résultats est donnée à l'Appendice C, p. 151).

2.2.5 Algorithme DRL/Lambda

Les sous-sections 2.2.1 à 2.2.4 contiennent tous les éléments nécessaires à l'élaboration d'un algorithme de type DRL. La procédure MapTransmission, utilisée dans le design du nouveau DRL, est une procédure qui, étant donné un nombre de liaisons requises dans chaque direction pour une paire de commutateur, installe toutes les liaisons possibles (selon une procédure similaire à celle de la page 61), sauf le dernier module. En plus, MapTransmission divise les liaisons libres du dernier module entre les deux commutateurs, selon la procédure décrite à la page 61, avant de renvoyer les résultats. La page 68 contient le design haut niveau du déroulement de DRL/Lambda.

Début

Initalisation des structure de données
Générer les trois **listes** suivantes:

- **les liens DF**
- **les liens PHU**
- **les liens Finaux**

Pour chaque paire de **commutateurs non connectés,**

- **additionner** le **trafic** sur les liens **finaux**

Pour chaque lien **DF**

- **calculer** les **liaisons** requises
- appeler **MapTransmission**
- **installer** le **dernier module** renvoyé par MapTransmission

Pour chaque lien **PHU**

- calculer C_A et C_D et déduire ECCS
- **calculer** les **liaisons** requises
- appeler **MapTransmission**
- **additioner** le **trafic de débordement minimal** sur les liens **finaux**, c'est-à-dire le trafic si le dernier module renvoyé par MapTransmission était installé

Pour chaque lien **final**

- **calculer** le nombre de **liaisons** requises
- appeler **MapTransmission**
- **générer** une **liste de modules** pour **couvrir** le **nombre maximal** de liaisons

Construire le BIP
Appeler et exécuter le module **BIP**
À partir des résultats renvoyés
Pour chaque module **PHU non installé**

- **calculer** le nouveau **trafic de débordement** et remettre à jour le trafic des liens finaux

Pour chaque lien **final**

- **calculer** le nombre de **liaisons** requises dans **chaque direction**
- **diviser** le **nombre maximal de liaisons permis** selon la **proportion** du nombre de liaisons calculé à l'étape précédente

Calculer le **trafic moyen par liaisons** et stocker l'information dans *NewRatio*

Fin

2.3. Modifications de la base RCMB et de SNAP

La présente section est divisée en deux sous-sections. La première sous-section décrit comment les tandems sont modélisés dans la base RCMB. En plus des tandems, la modélisation des commutateurs principaux, qui est légèrement modifiée, est aussi expliquée. La deuxième sous-section décrit les modification apportés à l'algorithme d'approvisionnement.

2.3.1 Les tandems et la base RCMB

Le tandem modélisé, illustré à la Figure 20, pour tester le nouvel algorithme est un commutateur numérique DMS-100 modifié. Puisque DRL/Lambda modélise un réseau de liens unidirectionnels, chaque type de liaisons est modélisé par deux noeuds: un noeud d'entrée (trafic provenant des autres commutateurs) et un noeud de sortie (trafic du commutateur vers le reste du réseau). On parlera de technologie de liaisons lorsqu'on fait référence à une paire de type de liaisons où l'on retrouve l'entrée et la sortie d'un module. La Figure 19 illustre le concept de technologie et de type de liaisons.

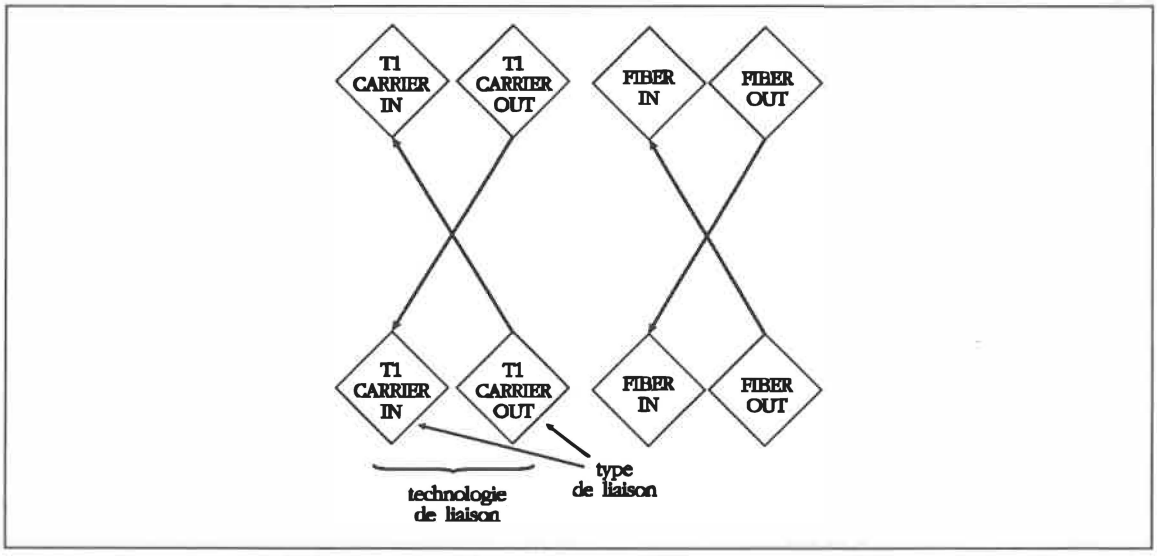


Figure 19. **Technologie et type de liaisons**

La modélisation des commutateurs principaux ne change pas beaucoup de celle décrite à la section 2.1.5. Une première différence vient du fait que le noeud LAMBDA utilise une version modifiée (décrite à la section 2.4.) de la fonction Lambda prototype. Le deuxième changement, plus important celui-là, c'est que chaque CP est maintenant défini comme un commutateur distant[1] du tandem auquel il est rattaché. Cette approche permet à un tandem qui est congestionné de réduire la demande sur les commutateurs qu'il dessert puisque l'algorithme d'approvisionnement peut, à partir du tandem, accéder aux noeuds principaux d'entrée (Figure 20).

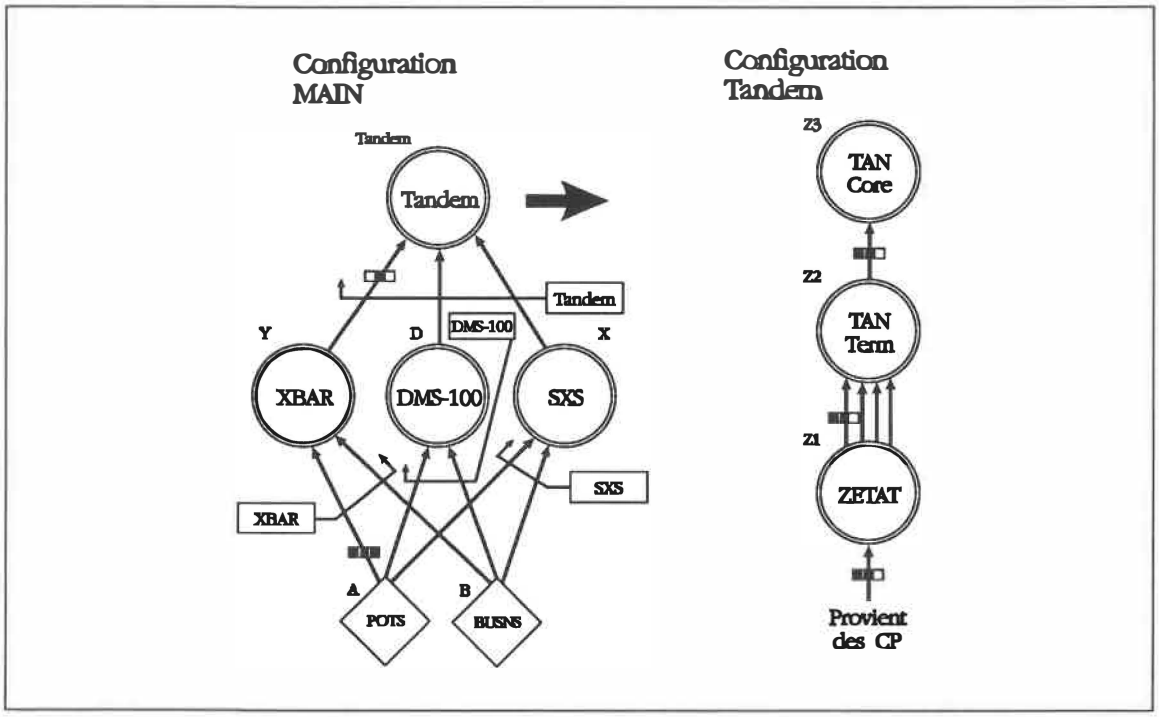


Figure 20. Modélisation des tandems dans la base RCMB

1. Une astuce est toutefois utilisée pour que SNAP ne considère pas ces liaisons comme un lien ombilical.

Contrairement aux CP, on ne peut utiliser la fonction Lambda avec le tandem puisqu'aucun trafic intercommutateur n'est généré par celui-ci. De plus, il est souhaitable que les totaux des liaisons estimés aux CP correspondent d'une façon proportionnelle à l'estimation pour le tandem. C'est-à-dire que si un CP voit sa charge réduite, donc conséquemment son nombre de liaisons avec le tandem, on veut que la réduction se répercute au tandem. La solution retenue consiste à garder une structure de données partagée où les CP inscrivent le nombre et le type de liaisons ainsi que le trafic qu'ils ont estimés avoir avec leur tandem. De cette façon, lorsque les noeuds des types de liaisons sont remis à jour pour le tandem, on dispose de l'information nécessaire, à savoir le nombre et le type de liaisons avec leur trafic. On n'a qu'à "piger" les données récoltées par la structure de données partagées. De cette façon, on est assuré de la cohérence interne de l'estimation: le total des liaisons pour un tandem correspond au total des estimations faites aux CP qui lui sont rattachés. Ce nouveau type de noeud est appelé LAMBDAT (pour LAMBDA Tandem) et se retrouve dans la configuration ZETAT (qui est semblable à la configuration ZETA de la page 51).

2.3.2 L'algorithme d'approvisionnement et DRL/Lambda

La modification décrite pour l'algorithme d'approvisionnement (à la page 52), avec la fonction Lambda prototype, reste valable pour le nouvel organigramme de SNAP avec DRL/Lambda. La seule différence c'est qu'une nouvelle séquence de modules est appelée, au lieu du seul module Trunker. Cette différence est illustrée à la Figure 21, page 73. On peut consulter la Figure 14, à la page 53, pour observer la différence et le reste de la boucle Compressor.

Lors du premier appel du module Trunker, un code indique au module qu'il doit produire la matrice des trafics intercommutateurs. Cette matrice est ensuite utilisée par DRL/Lambda pour dimensionner le réseau de liaisons. Il faut noter que la fonction Lambda n'est utilisée que par le module Installer (plus précisément, par le sous-programme I$OutSi). Lors du deuxième appel du module Trunker, on dispose alors de toute l'information nécessaire aux calcul des coefficients Lambda.

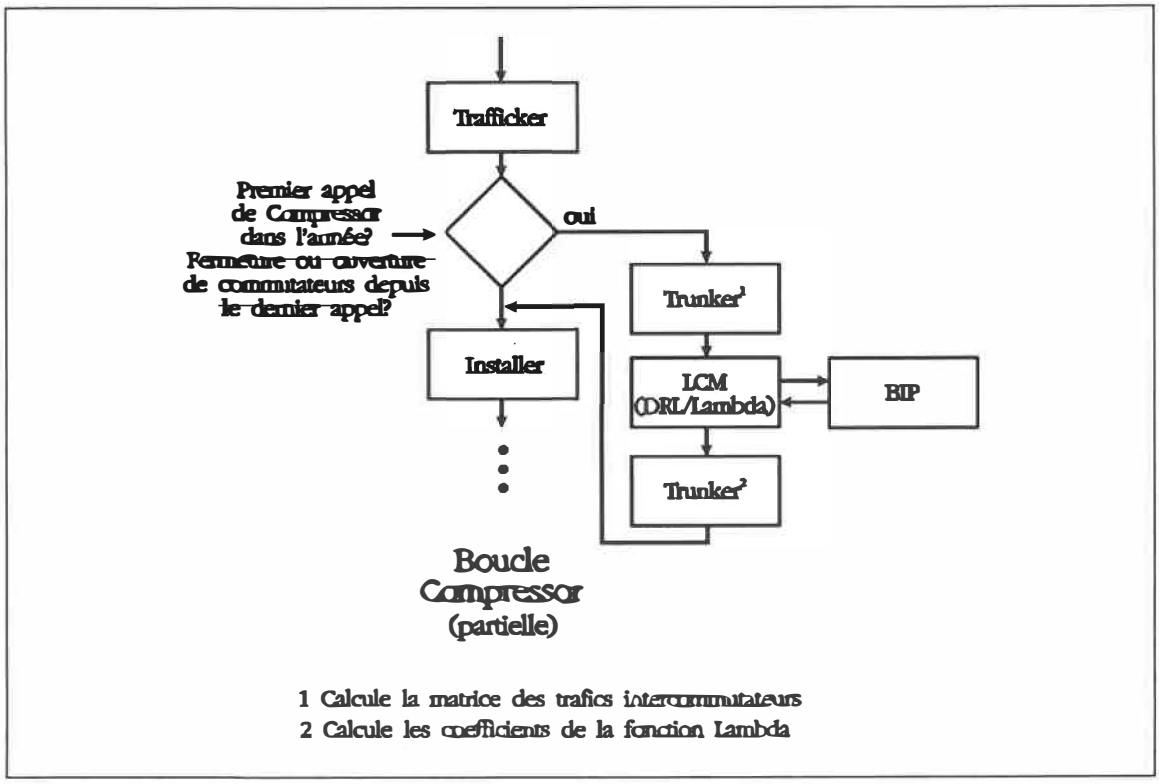


Figure 21. Organigramme fonctionnel avec DRL/Lambda

Le module BIP a comme tâche de solutionner le programme en nombres entiers qui est produit par DRL/Lambda. Les modules DRL/Lambda et BIP sont indépendants; le module BIP peut solutionner n'importe quel BIP qui lui est soumis. De cette façon, il est possible d'utiliser un autre algorithme plus performant en ne changeant que les procédures de l'interface dans le module DRL/Lambda (si nécessaire). Lorsque le module DRL/Lambda est terminé, le module est appelée une seconde fois. Cette fois c'est pour le calcul des coefficients de la fonction Lambda. Toutes les structures de données necessaires au calcul ont été initialisées par le module DRL/Lambda. À partir de ce moment, les coefficients sont figés pour toute la durée de l'itération de la boucle Compressor[1] .

1. Comme pour la fonction Lambda prototype.

2.4. La fonction Lambda

La fonction Lambda prototype, développée à la section 2.1, a été modifiée pour s'adapter au nouveau contexte de modélisation. La fonction Lambda utilisée avec DRL/Lambda contraint la modélisation du réseau de liaisons. En effet, on ne peut qu'avoir un type de technologie entre deux commutateurs donnés. Cette contrainte peut être facilement relâchée; elle n'a pour but que de simplifier l'écriture des procédures de calcul.

Puisque DRL/Lambda modélise un réseau de liens unidirectionnels, la fonction Lambda se scinde en deux parties, soit une partie pour le trafic vers le réseau et une partie pour le trafic en provenance du réseau. Voici les deux étapes du calcul des coefficients de la fonction Lambda. Premièrement, la matrice des trafics inter-commutateurs est calculée selon le MG. Deuxièmenent, le réseau de liaisons est dimensionné par l'algorithme DRL/Lambda décrit à la page 68. À ce moment, on connait les trafics intercommutateurs PHU et finaux ainsi que le nombre de liaisons que comprend chaque lien. Voici donc la dérivation de la fonction Lambda.

La fonction du modèle gravitationnel (MG) du trafic de S_i vers S_j,

$$CCS(S_i,S_j)=\sum_{n\in E}\frac{CCS(S_i,n)\,CCS(S_j|n)\,d(i,j)}{den(i,n)} \tag{2.18}$$

nous donne le trafic total qui provient de S_i vers S_j. Ce trafic est maintenant scindé en deux parties, soit une partie qui transite par le lien PHU et le trafic de débordement qui transite par le lien final. Si l'on défini α_{ij} comme étant la proportion du trafic qui transite par le lien PHU, alors $1-\alpha_{ij}$ est la proportion de trafic qui transite par le lien final[1]. À l'aide de ces définitions, on peut écrire

1. La somme des moyennes des trafics PHU et du trafic final d'un commutateur est égale au trafic intercommutateur total généré par celui-ci. Par contre, les variances des mêmes trafics ne s'additionne pas; la variance est beaucoup plus grande pour le trafic de débordement. De toute façon, la variance n'est pas utilisée dans le dimensionnement des liaisons par le module Installer (du moins pas directement).

$$PHU(S_i,S_j) = \alpha_{ij} \cdot \sum_{n \in E} \frac{CCS(S_i,n)\, CCS(S_j|n)\, d(i,j)}{den(i,n)} \tag{2.19}$$

et

$$Final(S_i,S_j) = (1-\alpha_{ij}) \cdot \sum_{n \in E} \frac{CCS(S_i,n)\, CCS(S_j|n)\, d(i,j)}{den(i,n)} \tag{2.20}$$

Au lieu de sommer par technologie de commutateur, comme à l'équation (2.1) (p. 44), on somme par technologie de liaisons, c'est-à-dire que l'on somme pour tous les S_j tel que $j \in SC_t^i = \{k \,|\, S_k$ possède un lien de technologie t avec $S_i\}$. Cette somme nous donne, à partir de (2.19) et (2.20),

$$PHU(S_i|SC_t^i) = \sum_{j \in SC_t^i} \left[\alpha_{ij} \cdot \sum_{n \in E} \frac{CCS(S_i,n)\, CCS(S_j|n)\, d(i,j)}{den(i,n)} \right] \tag{2.21}$$

et

$$Final(S_i|SC_t^i) = \sum_{j \in SC_t^i} \left[(1-\alpha_{ij}) \cdot \sum_{n \in E} \frac{CCS(S_i,n)\, CCS(S_j|n)\, d(i,j)}{den(i,n)} \right] \tag{2.22}$$

Si on définit

$$\Lambda_{nt}^{iPO} = \sum_{j \in SC_t^i} \alpha_{ij} \cdot \frac{CCS(S_j|n)\, d(i,j)}{den(i,n)} \tag{2.23}$$

$$\Lambda_{nt}^{iFO} = \sum_{j \in SC_t^i} (1-\alpha_{ij}) \cdot \frac{CCS(S_j|n)\, d(i,j)}{den(i,n)} \tag{2.24}$$

où $[\Lambda_{nt}^{ixy}]$ forme une matrice de dimension cinq dont les éléments sont des nombres réels. Les indices i et n sont respectivement l'indice de S_i et celui de l'échange n, l'indice t représente une des technologie de liaisons modélisée par la base RCMB (et non pas une technologie de commutateurs, comme à la page 43), l'indice $x \in \{P,F\}$, où P tient pour PHU et F pour final, enfin l'indice $y \in \{O,I\}$, où O est utilisé pour le trafic de sortie (out) et I est utilisé pour le trafic d'entrée (in).

Avec (2.23) et (2.24), les équations (2.21) et (2.22) deviennent

$$PHU(S_i|SC_t^i) = \sum_{n\in E} CCS(S_i,n)\,\Lambda_{nt}^{iPO} \tag{2.25}$$

$$Final(S_i|SC_t^i) = \sum_{n\in E} CCS(S_i,n)\,\Lambda_{nt}^{iFO} \tag{2.26}$$

De façon similaire pour le trafic d'entrée, le trafic provenant de S_j vers S_i, selon le modèle gravitationnel nous donne

$$CCS(S_j,S_i) = \sum_{n\in E} \frac{CCS(S_j,n)\,CCS(S_i|n)\,d(j,i)}{den(j,n)} \tag{2.27}$$

Le trafic provenant respectivement du lien PHU et du lien final sont

$$PHU(S_j,S_i) = \alpha_{ji}\cdot\sum_{n\in E} \frac{CCS(S_j,n)\,CCS(S_i|n)\,d(j,i)}{den(j,n)} \tag{2.28}$$

et

$$Final(S_j,S_i) = (1-\alpha_{ji})\cdot\sum_{n\in E} \frac{CCS(S_j,n)\,CCS(S_i|n)\,d(j,i)}{den(j,n)} \tag{2.29}$$

Les trafics PHU et final provenant de tous les commutateurs possédant des liens de technologie t avec S_i

$$PHU(SC_t^i|S_i) = \sum_{j\in SC_t^i}\left[\alpha_{ji}\cdot\sum_{n\in E} \frac{CCS(S_j,n)\,CCS(S_i|n)\,d(j,i)}{den(j,n)}\right] \tag{2.30}$$

$$Final(SC_t^i|S_i) = \sum_{j\in SC_t^i}\left[(1-\alpha_{ji})\cdot\sum_{n\in E} \frac{CCS(S_j,n)\,CCS(S_i|n)\,d(j,i)}{den(j,n)}\right] \tag{2.31}$$

En se rappelant que $CCS(S_i|n) = \sum_{k \in EAS_n} CCS(S_i,k)$, on défini

$$\Lambda_{nt}^{iPI} = \sum_{j \in SC_t^i} \left[\sum_{k \in EAS_n} \alpha_{ji} \cdot \frac{CCS(S_j,k)\, d(j,i)}{den(j,k)} \right] \tag{2.32}$$

$$\Lambda_{nt}^{iFI} = \sum_{j \in SC_t^i} \left[\sum_{k \in EAS_n} (1 - \alpha_{ij}) \cdot \frac{CCS(S_j|k)\, d(j,i)}{den(j,k)} \right] \tag{2.33}$$

À l'aide des équations (2.32) et (2.33), les équations (2.30) et (2.31) deviennent

$$PHU(SC_t^i|S_i) = \sum_{n \in E} CCS(S_i,n)\, \Lambda_{nt}^{iPI} \tag{2.34}$$

$$Final(SC_t^i|S_i) = \sum_{n \in E} CCS(S_i,n)\, \Lambda_{nt}^{iFI} \tag{2.35}$$

De façon similaire à l'équation (2.7), à la page 45, on trouve le nombre de liaisons en divisant le trafic par le trafic moyen par liaison. Si on défini $Ratio(S_i,x,y,t)$ comme étant le trafic moyen de S_i où $x \in \{PHU, Final\}$, $y \in \{In, Out\}$ et t qui est l'index de la technologie des liaisons, le nombre de liaisons $Liaisons(S_i,x,y,t)$ sont donnés par

$$Liaisons(S_i,PHU,Out,t) = \frac{PHU(S_i|SC_t^i)}{Ratio(S_i,PHU,Out,t)} \tag{2.36}$$

$$Liaisons(S_i,PHU,In,t) = \frac{PHU(SC_t^i|S_i)}{Ratio(S_i,PHU,In,t)} \tag{2.37}$$

$$Liaisons(S_i,Final,Out,t) = \frac{Final(S_i|SC_t^i)}{Ratio(S_i,Final,Out,t)} \tag{2.38}$$

$$Liaisons(S_i,Final,In,t) = \frac{PHU(SC_t^i|S_i)}{Ratio(S_i,Final,In,t)} \tag{2.39}$$

Les résultats des équations (2.38) et (2.39) sont stockés dans la structure de données décrite à la page 72. Les tandems dans la base RCMB ne possèdent pas de fonction comme les CP avec les équations produites dans cette section. Les liaisons sont calculées indirectement en utilisant les estimations faites aux CP du réseau. Cette méthode produit, comme mentionné, une cohérance interne des estimations.

Pour les noeuds LAMBDA modélisant des liaisons, il s'agit d'additionner les estimations *PHU* et *Final* correspondante pour obtenir le trafic total. De façon similaire, en additionnant *Liaisons* (S_i,*PHU*,y,t) et *Liaisons* (S_i,*Final*,y,t) pour $y = \{In,Out\}$ on obtient le nombre total de liaisons de technologie t. Ces estimations ont été scindées en partie PHU et partie finale essentiellement pour avoir l'estimation des liaisons pour les tandems.

3. Théorie et techniques

Le présent chapitre contient la théorie et la description des techniques utilisées dans la solution du réseau optimal de l'algorithme DRL/Lambda. Ce chapitre présente de façon plus spécifique la "mécanique" derrière la résolution numérique du BIP et la représentation spéciale développée pour les données.

3.1. Programmation en nombre entiers

La présente section expose la théorie utilisée pour la résolution numérique du programme en nombre entier de la section 2.2.4. Pour plus de détails, le lecteur est référé au livre de Nemhauser et Wolsey, *Integer and Combinatorial Optimization* (Wiley, 1988) d'où la théorie qui suit a été tirée.

3.1.1 Forme standard du BIP

Nous avons vu à la section 2.2.4 la description du BIP que l'on doit résoudre pour adapter la procédure ECCS à la modularité des liens. À partir du réseau optimal suggéré par la procédure ECCS, la solution du BIP nous donne le réseau qui s'en approche le plus tout en minimisant l'introduction de nouvelles liaisons à l'aide de la fonction économique. Le problème est le suivant:

$$\min \sum_{i \in S} \left[\sum_{q=1}^{p_i} d_{iq} y_{iq} + \sum_{j \in L_i} e_{ij} x_{ij} \right]$$

$$\text{où } L_i = \{ k \mid k \in PHU_i \text{ et } k > i \}$$

sous les contraintes

$$\sum_{q=1}^{p_i} m_{iq} y_{iq} + \sum_{j \in PHU_i} g_{ij} x_{ij} \geq M_i \quad \text{pour tout } i \in S$$

Le BIP précédent n'est pas sous forme standard. Il doit être transformé sous cette forme avant de pouvoir le résoudre numériquement, soit

$$\max\{cx: Ax \leq b, x \in B^n\} \quad \text{(forme standard)},$$

où A est une matrice $m \times n$ entière et $b \in Z_+^m$. Afin de passer du BIP initial à la forme standard, on introduit la variable complément. Si $x \in B$ alors son complément $\bar{x} \in B$ est défini comme suit

$$\bar{x} = \begin{cases} 0 & \text{si } x = 1 \\ 1 & \text{si } x = 0 \end{cases}$$

ou sous forme d'équation

$$\bar{x} = 1 - x$$

On réécrit la fonction économique du problème initial

$$\min \sum_{i \in S} \left[\sum_{q=1}^{p_i} d_{iq}(1 - \bar{y}_{iq}) + \sum_{j \in L_i} e_{ij}(1 - \bar{x}_{ij}) \right]$$

$$\min \sum_{i \in S} \left[\sum_{q=1}^{p_i} d_{iq} - \sum_{q=1}^{p_i} d_{iq}\bar{y}_{iq} + \sum_{j \in L_i} e_{ij} - \sum_{j \in L_i} e_{ij}\bar{x}_{ij} \right]$$

$$\min -\sum_{i \in S} \left[\sum_{q=1}^{p_i} \bar{y}_{iq} + \sum_{j \in L_i} \bar{x}_{ij} \right] + \textit{Constante}$$

qui est équivalent à

$$\max \sum_{i \in S} \left[\sum_{q=1}^{p_i} \bar{y}_{iq} + \sum_{j \in L_i} \bar{x}_{ij} \right]$$

Pour les contraintes on obtient

$$\sum_{q=1}^{p_i} m_{iq}(1 - \bar{y}_{iq}) + \sum_{j \in PHU_i} g_{ij}(1 - \bar{x}_{ij}) \geq M_i$$

$$\sum_{q=1}^{p_i} m_{iq}\bar{y}_{iq} + \sum_{j\in PHU_i} g_{ij}\bar{x}_{ij} \leq \sum_{q=1}^{p_i} m_{iq} + \sum_{j\in PHU_i} g_{ij} - M_i$$

$$\sum_{q=1}^{p_i} m_{iq}\bar{y}_{iq} + \sum_{j\in PHU_i} g_{ij}\bar{x}_{ij} \leq \tilde{M}_i$$

où

$$\tilde{M}_i = \sum_{q=1}^{p_i} m_{iq} + \sum_{j\in PHU_i} g_{ij} - M_i$$

Le problème sous forme standard devient

$$\max \sum_{i\in S} \left[\sum_{q=1}^{p_i} \bar{y}_{iq} + \sum_{j\in L_i} \bar{x}_{ij} \right]$$

$$\sum_{q=1}^{p_i} m_{iq}\bar{y}_{iq} + \sum_{j\in PHU_i} g_{ij}\bar{x}_{ij} \leq \tilde{M}_i \quad \text{pour tout } i\in S \tag{3.1}$$

où

$$\tilde{M}_i = \sum_{q=1}^{p_i} m_{iq} + \sum_{j\in PHU_i} g_{ij} - M_i$$

3.1.2 Définition des coupes

Ce type de problème est généralement résolu à l'aide d'un algorithme de type *branch-and-bound*[1] avec relaxation linéaire, mais les problèmes en nombres entiers de type BIP possèdent certaines propriétés qui permettent d'améliorer l'efficacité de l'algorithme.

1. Un algorithme de type *branch-and-bound* est essentiellement un algorithme d'énumération qui utilise l'information disponible à chacun des noeuds de l'arbre d'énumération (comparaison avec une solution connue, problème dual sans solution) pour réduire le nombre de branches à considérer.

L'algorithme qui a été utilisé pour le programme est appelé FCPA, pour *Fractional Cutting Plane Algorithm*. Cet algorithme est complémenté en dernier lieu par l'algorithme *branch-and-bound*. Le principe derrière FCPA est le suivant: on résout le problème

$$P_0 = \max\{cx: Ax \leq b, 0 \leq x \leq 1\}$$

c'est-à-dire la relaxation linéaire du problème en nombres entiers ($x_i \in [0,1]$ au lieu de $x_i \in \{0,1\}$). On définit l'ensemble $S_R[P_0]$ comme étant l'espace solution de P_0 et S comme étant

$$S = \{x |\ x \text{ satisfait les contraintes de } P_0 \text{ et } x \in B^n\}$$

c'est-à-dire l'ensemble des solutions entières possibles pour P_0. Si x^*est solution de P_0 et que $x^* \in B^n$, alors x^* est solution du BIP; sinon il existe un i tel que $x_i^* \notin B$. À l'aide de x^*on génère un ensemble de coupes, c'est-à-dire des contraintes de la forme $\sum_{j\in N} a_j x_j \leq b$ de tel façon que $\sum_{j\in N} a_j x_j^* > b$ tout en ayant que pour tout $\tilde{x} \in S \rightarrow \sum_{j\in N} a_j \tilde{x_j} \leq b$, c'est-à-dire que la nouvelle contrainte coupe x^* de l'espace-solution tout en nous assurant qu'aucun point entier de l'espace-solution $S_R[P_0]$ n'est enlevé de celui-ci. On appelle P_1 le nouveau problème qui comprend le problème P_0 et l'ensemble des coupes générées. On résout P_1 et l'on recommence la séquence jusqu'à ce que l'on trouve une solution entière ou que l'on ne peut plus générer de coupes (et l'on doit utiliser le *branch-and-bound*).

Les coupes sont des inégalités appelées couverts. On dira que $C \in N$ est un couvert si $\sum_{j\in C} a_j x_j > b$. Si C est un couvert, alors

$$\sum_{j\in C} x_j \leq |C| - 1$$

est une inégalité valide pour $S = \{x \in B^n: \sum_{j\in N} a_j x_j \leq b\}$, où $a_j \in Z_+^1$ pour $j \in N$ et $b \in Z_+^1$. Afin de trouver le couvert C, il nous faut résoudre le problème de séparation pour les inégalités de couverts,

$$\zeta = \min\left\{\sum_{j\in N}(1-x_j^*)z_j : \sum_{j\in N} a_j z_j > b, z \in B^n\right\} \qquad (3.2)$$

où z est le vecteur caractéristique de C. Si $\zeta \geq 1$, alors x^* satisfait tous les couverts de S. Par contre, si $\zeta < 1$, alors $\sum_{j\in C} x_j \leq |C| - 1$ est le couvert le plus *violé* pour S (on peut trouver la preuve dans Nemhauser et Wolsey, p. 460).

Il faut à chaque génération de coupes résoudre le problème (3.2) pour chacune des contraintes du problème initial. Ce type de problème est, après transformation à l'aide des variables compléments, de type Sac de Campeur 0-1, c'est-à-dire

$$\max\left\{cx: \sum_{j\in N} a_j z_j \leq b, x \in B^n\right\} \quad \text{(Sac de Campeur 0-1)}$$

L'algorithme utilisé pour résoudre le Sac de Campeur 0-1 (Knapsack 0-1) est décrit pp. 450-454 de Nemhauser et Wolsey.

Afin d'accélérer la convergence de l'algorithme, les coupes sont *renforcies* à l'aide de la Proposition 2.6 (pp. 268-270). Le renforcement des coupes consiste à augmenter les coefficients de la contrainte trouvée, toujours sans couper des points de S. De cette façon, une plus grande partie de l'espace-solution de la relaxation linéaire est enlevée, d'où l'accélération de l'algorithme puisque l'espace-solution entier est mieux circonscrit par les nouvelles contraintes.

3.2. Représentation matricielle

La matrice des contraintes générées par l'inégalité (3.1) possède une forme particulière. Une représentation spéciale a donc été développée pour en prendre avantage.

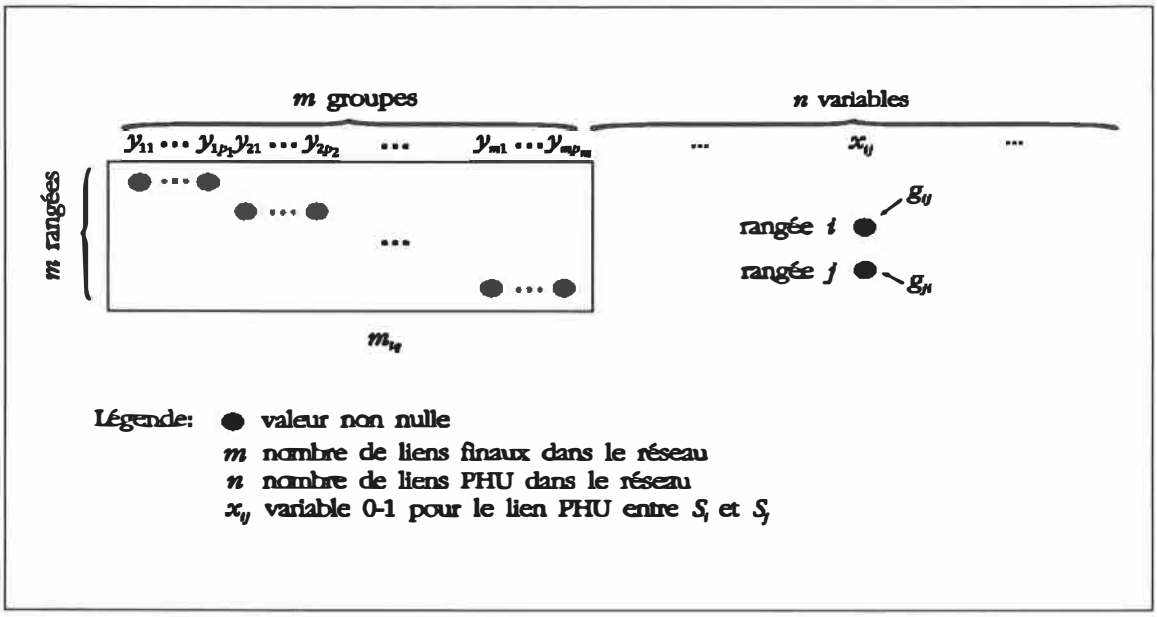


Figure 22. Forme générique de la matrice des contraintes

3.2.1 Structure de données

La Figure 22 illustre la forme générique de la matrice des contraintes du BIP. On constate que cette matrice contient beaucoup de 0 relativement au nombre de valeurs non nulles. Il y a aussi une redondance de l'information des a_{ij} pour les variables des liens finaux puisque selon (3.1), m_{iq} est la modularité du q ième module du lien final du commutateur. Mais puisque le nombre de technologie pour les liens est limité (de 3 à 6 en pratique), plusieurs m_{iq} possèdent la même valeur. Au lieu d'utiliser la réprésentation classique d'une matrice, c'est-à-dire un tableau doublement indicé, une représentation plus compacte et plus performante a été développée. La matrice des contraintes est composée des cinq structures de données suivantes: *ci*, *ri*, *cin*, *rin* et *a*. Dans les explications ci-bas, les variables *icol* et *irow* représentent les index originaux de la matrice des contraintes. Voici la pseudo déclaration de ces structures:

DCL *ci* (*icol*,{*n* ième valeur non nulle}) {
    ai -> index dans le vecteur a
    row -> rangée de *A*
}

DCL *ri* (*irow*,{*n* ième valeur non nulle}) {
    ai -> index dans le vecteur a
    col -> colonne de *A*
}

DCL $cin(icol) = \max_n ci(icol,n)$

(combien de valeur non nulle contient la colonne *icol*)

DCL $rin(irow) = \max_n ri(irow,n)$

(combien de valeur non nulle contient la rangée *irow*)

DCL *a* → valeurs de la matrice *A*

Le vecteur *a* est construit de la façon suivante, on suppose qu'il y a t technologie pour les liens, p qui est égal à $|\{g_{ij}\}|$ (la cardinalité de l'ensemble des g_{ij}), et que le nombre maximum de valeur non nulle dans une même rangée de *A* est z,

$a(i)$	=	modularité d'un module de technologie	pour $i = 1$ à t
$a(i+t)$	=	valeur de b_i	pour $i = 1$ à p
$a(i+t+p)$	=	i (utilisées pour les coupes)	pour $i = 1$ à $\lfloor [z/2] $

en supposant que l'on a assigné chaque g_{ij} à un b_k différent.

Le choix de cette représentation minimise autant que possible la redondance d'information de la matrice *A*..

3.2.2 Exemple de représentation et avantages

Voici un exemple d'une matrice représentée sous cette forme:

$$\begin{bmatrix} 24 & 24 & 0 & 0 & 0 & 0 & 13 & 16 & 0 \\ 0 & 0 & 12 & 12 & 24 & 0 & 15 & 0 & 23 \\ 0 & 0 & 0 & 0 & 0 & 24 & 0 & 8 & 9 \end{bmatrix}$$

qui devient

$$cin = [\,1,1,1,1,1,1,2,2,2\,]$$
$$rin = [\,4,5,3\,]$$

$$[\,ci(icol,n).row\,] = \begin{bmatrix} 1 & - \\ 1 & - \\ 2 & - \\ 2 & - \\ 2 & - \\ 3 & - \\ 1 & 2 \\ 1 & 3 \\ 2 & 3 \end{bmatrix} \quad [\,ri(irow,n).col\,] = \begin{bmatrix} 1 & 2 & 7 & 8 & - \\ 3 & 4 & 5 & 7 & 9 \\ 6 & 8 & 9 & - & - \end{bmatrix}$$

$$a = [\,12,24,13,16,15,23,9,8,1,2\,]\ ^{1}$$

$$[\,ci(icol,n).ai\,] = \begin{bmatrix} 2 & - \\ 2 & - \\ 1 & - \\ 1 & - \\ 2 & - \\ 2 & - \\ 3 & 5 \\ 4 & 7 \\ 6 & 8 \end{bmatrix} \quad [\,ri(irow,n).ai\,] = \begin{bmatrix} 2 & 2 & 3 & 4 & - \\ 1 & 1 & 2 & 5 & 6 \\ 2 & 7 & 8 & - & - \end{bmatrix}$$

1. Voir la note en bas de page de la page suivante pour l'explication concernant les deux dernières valeurs du vecteur *a*.

Cette représentation est plus compacte[1] lorsque le problème à résoudre est d'une certaine taille (soit 8 rangées et plus), mais possède d'autres avantages autre que la consommation réduite d'espace mémoire. Voici la liste de ces avantages:

- **les produits scalaires sont calculés très rapidement:**
  cette représentation élimine les multiplications par un élément nul, et cela sans être obligé de tester les valeurs à multiplier. Le produit scalaire par exemple entre la rangée *irow* et le vecteur v est donné par

$$\text{produit scalaire} = \sum_{k=1}^{rin(irow)} a(ri(irow,k).ai) \cdot v(ri(irow,k).col)$$

- **la génération des coupes est facilitée:**
  l'application de la Proposition 2.6 décrite précédemment nécessite que les a_{ij} soit ordonnés de façon décroissante; à l'initialisation du BIP, on appelle une procédure de type *HeapSort* pour chacune des rangées de A qui réordonne ri de façon à ce que

$$a(ri(irow,i).ai) \geq a(ri(irow,j).ai) \quad \text{pour tout } i < j$$

On note que cette procédure ne modifie que ri et aucune autre structure, de plus la procédure d'ordonnancement n'est exécutée qu'une seule fois à l'initialisation (ce qui évite l'utilisation de structures de données supplémentaires pour garder ces résultats). Seul l'ordre du calcul du produit scalaire est affecté par cette procédure, ce qui n'a aucune importance sur le résultat évidemment.

1. Ce qui n'est pas tout de suite évident avec l'exemple. Plus la matrice comporte de rangées, plus on gagne en compacité. De plus, cette structure permet l'ajout de contraintes de type couvert avec une addition minimale d'information: il s'agit de modifier les structures en conséquence et de rediriger tous les membres *.ai* vers les derniers éléments de *a*.

4. Résultats

Avant d'obtenir les résultats qui suivent, le noyau de SNAP et la base RCMB ont été modifiés pour tester la méthodologie décrite au Chapitre 2. Le présent chapitre est divisé en deux sections, soit une section par étape de solution. La première section présente les tests effectués avec la première version modifiée du noyau. Cette première version est une implantation de la fonction Lambda prototype décrite à la section 2.1. La deuxième section décrit les tests et présente les résultats obtenus à l'aide de la deuxième version du noyau. Cette deuxième version utilise le nouveau DRL/Lambda et la fonction Lambda. La deuxième version du noyau comprend tous les changements présentés aux sections 2.2. à 2.4.

4.1. Résultats de la première étape

Cette section comprend l'ensemble des résultats obtenus avec la méthodologie développée dans la première étape. Cette étape consistait à intégrer le calcul de la charge maximale en une seule et même phase. La première sous-section décrit sommairement comment le noyau a été modifié. La deuxième sous-section décrit les tests et les paramètres retenus pour l'évaluation des performances de la fonction Lambda prototype. La troisième et dernière sous-section comprend l'ensemble des résultats obtenus sous forme de tableaux.

4.1.1 Les modifications du noyau

La première étape a consisté à modifer le noyau afin de tester une version opérationnelle de la fonction Lambda prototype. L'élaboration de la méthodologie a nécessité un travail considérable puisqu'il fallait très bien connaître l'algorithme d'approvisionnement, la base de donnée RCMB et la relation entre les deux. La fonction Lambda prototype a été construite en ayant pour objectif l'intégration la plus transparente possible aux algorithmes en place. Cet objectif a été recontré d'une

façon très satisfaisante. La réalisation de la première étape n'a nécessité la modification ou l'ajout que d'au plus 250 lignes de PL/1[1]. En plus de ces changements, le module de validation de la base RCMB a été légèrement modifié. Cette modification permet aux noeuds LAMBDA, qui modélisent maintenant les liaisons, de ne pas être des noeuds principaux d'entrée. Ce nombre réduit de modifications s'explique par trois raisons.

La première raison, c'est la conception modulaire (et sa documentation) du noyau de SNAP. Chacun des quatres blocs de l'algorithme d'approvisionnement est divisé en sous-unités, chaque sous-unité exécutant une fonction spécifique. De plus, chaque sous-unité ne reçoit que les paramètres pertinents à sa fonction de la part de la sous-unité d'appel. De cette façon, il est facile de trouver l'endroit où doit s'insérer une partie de code.

Le design adopté pour la base RCMB et le noyau de SNAP constitue la deuxième raison. Le nom de la fonction, qui doit traiter un noeud de la base RCMB, est un attribut du noeud. Cet attribut permet la définition de fonction différente de la fonction STANDARD sans nécessiter de "bricolage". En fait, la fonction Lambda est la première fonction non-STANDARD à utiliser ce mécanisme.

Enfin, la troisième raison consiste au fait que la fonction Lambda "imite" le trafic du MG. Les coefficients Λ_{nt}^{i} et *NewRatio* sont contruits en insérant seulement quelques lignes de code dans le module Trunker pour collecter l'information nécessaire. Un drapeau indique au module Trunker s'il doit remettre à jour les coefficients de la fonction Lambda ou simplement calculer l'injection et le retrait de liaisons dans le réseau[2].

1. Le noyau est écrit en PL/1 et fonctionne sous CMS avec un ordinateur de type IBM 3090.
2. Soit juste avant la transition entre le réseau de l'année précédente et le nouveau réseau.

Contrairement à ce que pourrait nous laisser croire la comparaison de la Figure 10, à la page 27, et le nouvel organigramme de la Figure 14, à la page 53, la taille du noyau de SNAP n'a pas été réduite de beaucoup. Les modules Compressor et Installer des deux côtés de la base RCMB, à la Figure 10, sont les mêmes. Un paramètre indique au module s'il doit traiter le côté gauche ou le côté droit de la base RCMB.

4.1.2 Description des tests avec l'étude X

Deux études réelles de planification ont servi de banc d'essai pour la fonction Lambda prototype. Afin de respecter la confidentialité des documents, nous nous référerons à ces deux études sous les noms d'étude X et d'étude Y. Nous décrirons premièrement les tests faits à l'aide de l'étude X.

Dans cette première série de tests avec l'étude X, la fonction Lambda a été évaluée à partir de trois paramètres. Le premier paramètre d'évaluation est la précision des résultats. Ce paramètre est important puisque le planificateur s'attend à ce que tous les résultats produits par le noyau respecte la tolérance demandée[1]. Le deuxième paramètre d'évaluation est l'impact de la tolérance sur le temps de calcul. À l'aide de ce deuxième paramètre et du premier, nous sommes en mesure de trouver le meilleur compromis entre la précision des résultats et la précision de calcul. Le troisième paramètre d'évaluation est le nombre de recherches binaires requis. Le nombre de recherches binaires est un bon indicateur de la précision de la fonction Lambda dans la prédiction du trafic. De plus, le nombre de recherches binaires indique si les commutateurs congestionnés ont interagi entre eux. L'interaction entre les commutateurs est introduite par la fonction Lambda[2].

1. La tolérance est contenue dans la base DSN.
2. Voir l'Appendice A pour plus de détails.

L'étude X originale (DSN X) utilise une base de données RCMB très détaillée et complexe. Dans les tests qui suivent, une autre base RCMB a été utilisée et ce pour plusieurs raisons. Premièrement, la base RCMB originale est classée document confidentiel par la compagnie qui l'utilise. Deuxièmement, les modifications à apporter auraient nécessité beaucoup trop de temps[1] et une connaissance poussée de la structure et la sémantique des modèles de coûts (cost model). Enfin, une base RCMB complexe implique des temps de calcul long et intélorable dans la mise au point de programmes, où il faut exécuter à répétition le noyau lors du déverminage. Pour toutes ces raisons, il a été décidé de modifier DSN X afin de fonctionner avec une base RCMB facilement manipulable (donc petite). La base utilisée pour les tests est une version modifiée de la base RCMB COU[2]. La nouvelle base RCMB modifiée est appelée COM (pour COu Modifiée). En plus des modifications données à la section 2.1., deux points de congestion ont été rajoutés.

Le premier point de congestion (PC) est situé dans la configuration 100 Core. La Figure 23 illustre la configuration 100 Core. La partie ombragée contient les noeuds qui ont été rajoutés. Le noeud Z3, qui est défini comme étant un PC, divise le trafic reçu par 86 000 CCS pour calculer le volume. Défini de cette façon, le noeud génère deux unités de ressource THING lorsque le trafic dépasse 86 000 CCS. Le trafic à l'entrée de ce noeud est égal à la somme du trafic provenant du réseau d'accès et des liaisons. On peut s'en convaincre en suivant le trajet du trafic avec les Figures 24 et 25 (aux pp. 92 et 93) qui illustrent repectivement la partie de la configuration MAIN modélisant un DMS-100 et la configuration 100 Term. Les parties ombragées de ces figures indiquent les parties enlevées ou ajoutées à la base (soit pour transformer la base COU à COM ou l'addition d'un point de congestion dans l'une ou l'autre des bases.)

1. Notamment à cause du haut niveau de détail et du nombre considérable de types de commutateur modélisés par cette base RCMB.
2. Cette base est principalement utilisée pour les cours portant sur la base RCMB chez RBN.

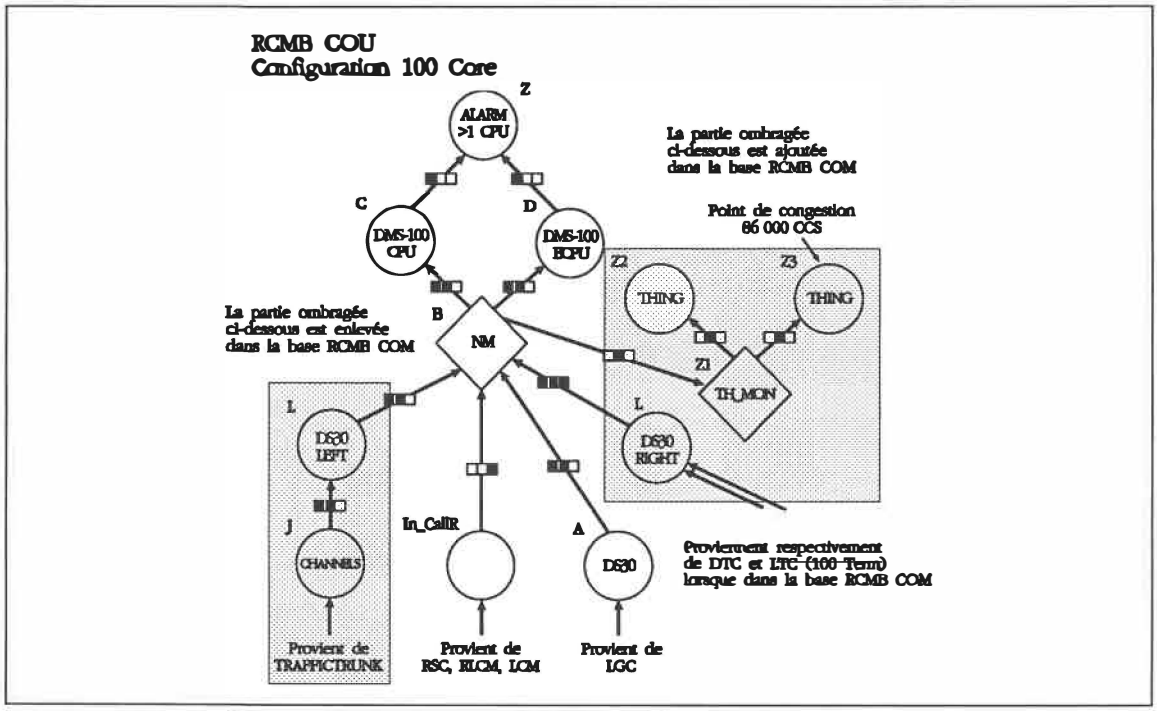


Figure 23. **Point de congestion: 86 000 CCS (DMS-100)**

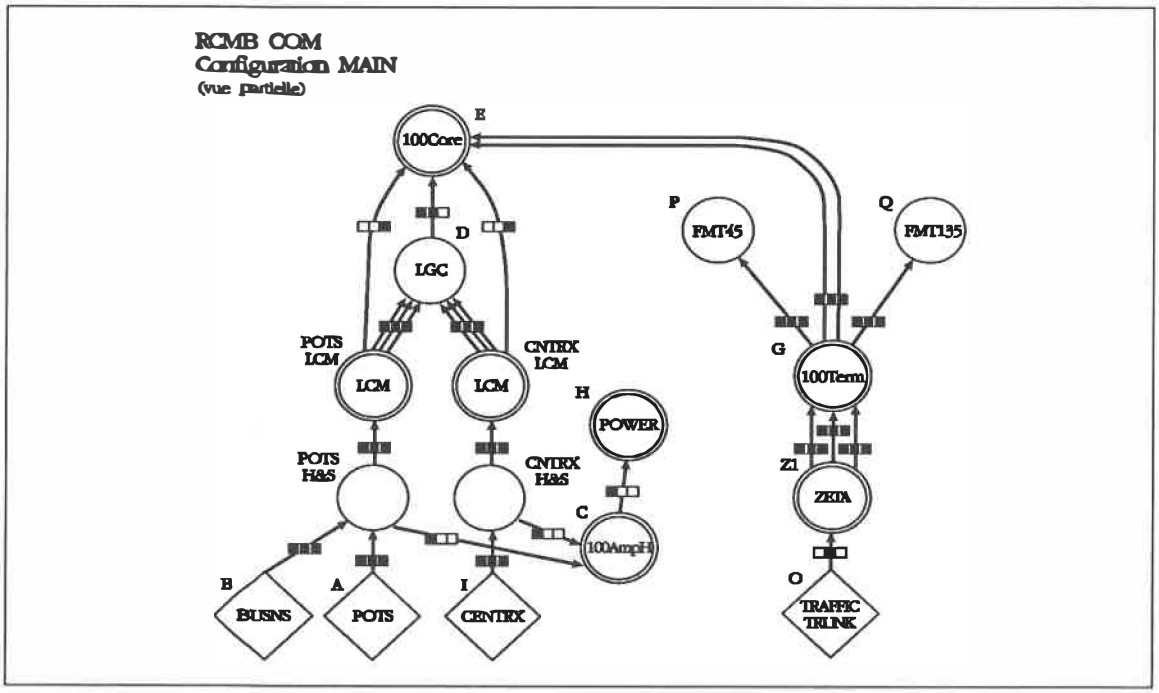


Figure 24. **Configuration MAIN (commutateur DMS-100)**

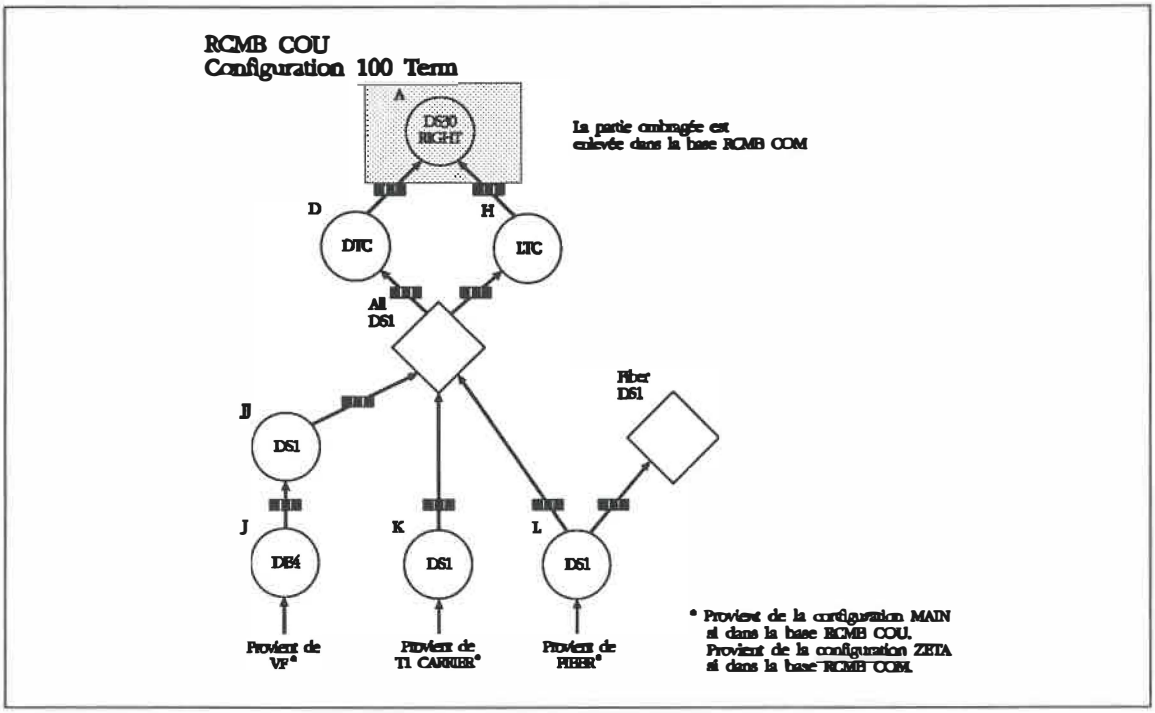


Figure 25. Configuration 100 Term (DMS-100)

Le deuxième PC est situé dans la configuration SXS (Figure 26). Ce point de congestion est activé lorsque le trafic des liaisons d'un commutateur de type SXS dépasse 7200 CCS. La Figure 27 montre le noeud d'appel de la configuration SXS[1].

On observe aux Figures 23 et 26 que chaque ensemble de noeuds rajoutés avec le PC contient un noeud stratégique. Il est possible, avec ce mécanisme, d'utiliser ou non le PC en choisissant une des deux branches comme le type de serveur à déployer. Un seul commutateur de type DMS-100 a été choisi dans l'étude X pour les tests. Ce commutateur est appelé DMS A. Le seul commutateur de type SXS, pour lequel le deuxième PC a été défini, est appelé SXS B.

1. La configuration MAIN est la configuration la plus haute dans la hiérarchie de la base RCMB COU et COM. C'est elle qui appelle toutes les configurations de deuxième niveau comme 100 Core, 100 Term, LCM, SXS, et ZETA.

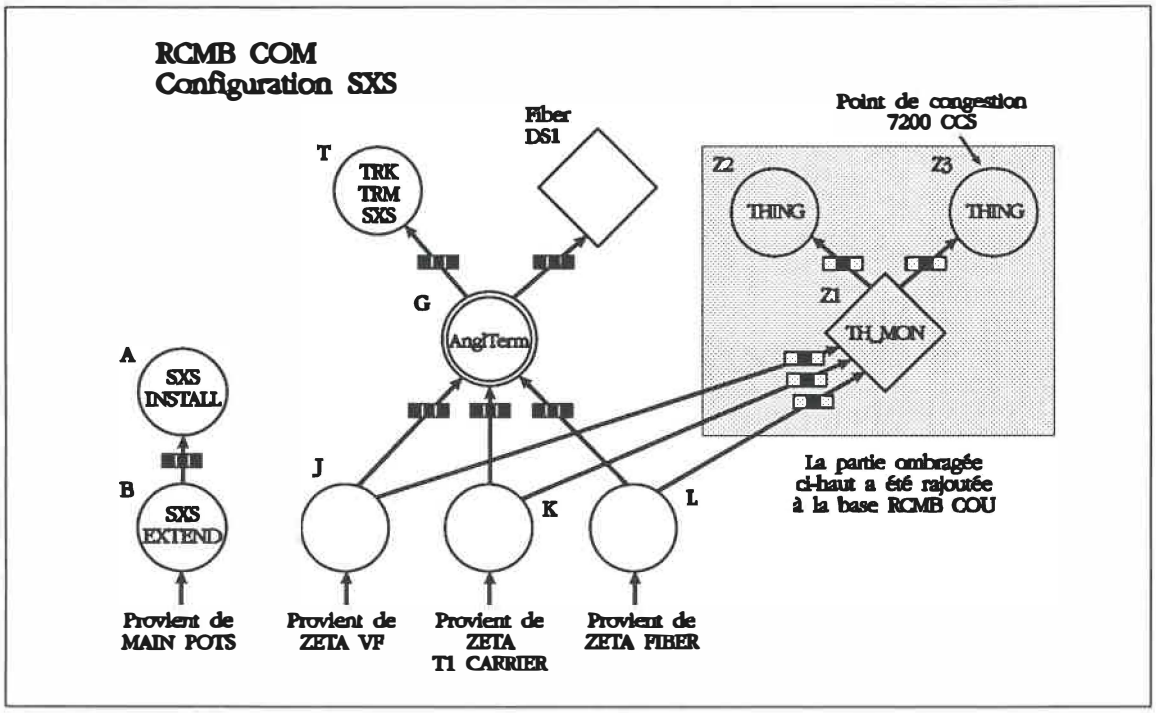


Figure 26. **Point de congestion: 7200 CCS (SXS)**

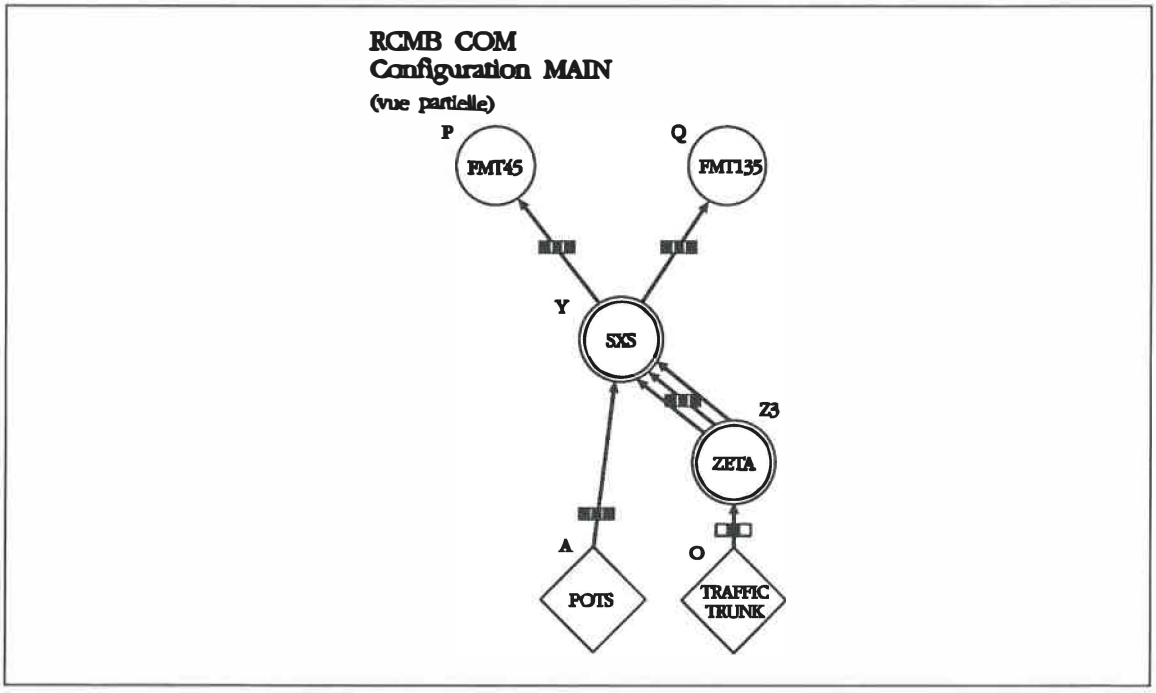


Figure 27. **Configuration MAIN (commutateur SXS)**

Six tests, soit trois tests avec deux scénarios différents ont été utilisés pour l'obtention des résultats. Les trois tests sont les suivants: avec le PC du DMS A seulement, avec le PC du SXS B seulement et avec les deux PC simultanément. Comme premier scénario, l'étude X sans modification a été utilisée. Le deuxième scénario est une version modifiée du premier où des stratégies ont été rajoutées afin d'absorber l'excédant de demande qui a découlé des PC précédemment décrits.

4.1.3 Résultats avec l'étude X

Le noyau a été exécuté pour chacun des tests avec trois tolérances, soit 2, 1 et 0%. L'étude X couvre une période de cinq années. Les résultats sont présentés sous forme de tableaux. Le Tableau 1 (p. 96) montre la forme générique utilisée pour les tableaux. Avant de présenter les résultats, voici les calculs et détails concernant les tableaux.

Le symbole Δ% représente l'écart en pourcentage entre la valeur retournée par le noyau et la valeur théorique maximale. Par exemple, voici comment Δ% est calculé pour le DMS A. On remarque à la Figure 23 (p. 92) que le noeud stratégique Z1 modélise la ressource TH_MON (THING MONitor). Ce noeud est défini de façon à ce que le volume généré soit dans un rapport 1:1 avec le trafic à l'entrée. Avec ce noeud, le rapport des ressources déployées dans le réseau comprend le trafic à l'entrée du noeud Z3[1] en terme d'unité de TH_MON. Puisque la valeur maximale admissible est de 86 000 CCS par unité d'équipement THING, la charge calculée par SNAP respecte la relation TH_MON ≤ 86 000. Donc Δ%, pour le DMS A, est obtenu par

$$\Delta\% = \left(\frac{86\,000 - \text{TH\_MON}}{86\,000}\right)\%$$

1. Nous avons vu que le trafic passe inchangé à travers les noeuds de la base RCMB.

Les cases ombragées des tableaux indiquent qu'il y a eu congestion au commutateur en abscisse et à l'année en ordonnée. Par exemple, dans le tableau générique, le commutateur X aurait vu sa charge maximale admissible excédée aux années 1, 2 et 5. C'est à partir de ces cases ombragées que l'on calcule Max Δ%. Les années où il n'y a pas de congestion ne peuvent servir au calcul de Max Δ% puisqu'aucune recherche binaire n'est exécutée pour le commutateur.

Tableau 1. **Forme générique**

Nom de l'étude **Points de congestion utilisés**							
Nom du commutateur	**Δ%**					**Max Δ%**	**Test**
Comm→	**Δ% année 1**	**Δ% année 2**	**Δ% année 3**	**Δ% année 4**	**Δ% année 5**	**(Voir texte)**	No. du test Tolérance en % Temps CPU requis
Nombre de Recherches Binaires (NRB)	**NRB année 1**	**NRB année 2**	**NRB année 3**	**NRB année 4**	**NRB année 5**	**NRB Total**	

Les pages qui suivent contiennent les tableaux des résultats de la première étape. Tous les tests ont été exécutés avec le même ordinateur, soit un IBM 3090-200E, avec le même noyau. Le temps requis, en secondes, est le temps réel qui inclut la lecture des bases DSN et RCMB ainsi que le temps de calcul et la production des rapports. La tolérance, en pourcentage, est la plage de tolérance utilisée par l'algorithme de recherche binaire du module Compressor. Il est normal (et prévu) que la valeur Max Δ% dépasse la tolérance démandée, surtout lorsque la tolérance est petite. Les tests ont pour but, entre autre, de trouver la plage de tolérance où l'on est assuré que les résultats produits respectent la tolérance demandée.

Tableau 2. **Résultats du test no. E1-1**

Étude X originale Point de congestion: DMS A							
Commutateur	Δ%					Max Δ%	Test
DMS A	0.8	0.7	0.7	0.1	0.7	0.8	No. E1-1-2 Tolérance: 2.0% Temps CPU: 697s
NRB	1	1	1	0	1	4	
DMS A	0.2	0.1	0.0	0.5	1.1	1.1	No. E1-1-1 Tolérance: 1.0% Temps CPU: 756s
NRB	1	1	1	1	1	5	
DMS A	0.1	0.1	0.1	0.1	0.1	0.1	No. E1-1-0 Tolérance: 0.0% Temps CPU: 1104s
NRB	1	2	2	1	1	7	

Tableau 3. **Résultats du test no. E1-2**

Étude X originale Point de congestion: SXS B							
Commutateur	Δ%					Max Δ%	Test
SXS B	1.1	1.7	2.7	1.5	0.7	1.7	No. E1-2-2 Tolérance: 2.0% Temps CPU: 629s
NRB	1	1	0	0	0	2	
SXS B	1.0	1.2	2.2	1.0	0.2	1.2	No. E1-2-1 Tolérance: 1.0% Temps CPU: 701s
NRB	1	2	0	0	0	3	
SXS B	1.0	1.0	2.1	0.8	0.0	1.0	No. E1-2-0 Tolérance: 0.0% Temps CPU: 838s
NRB	1	2	0	0	1	4	

Tableau 4. **Résultats du test no. E1-3**

Étude X originale Points de congestion: DMS A et SXS B							
Commutateur	Δ%					Max Δ%	Test
DMS A	0.5	0.4	0.3	0.8	0.2	0.8	No. E1-3-2 Tolérance: 2.0% Temps CPU: 767s
SXS B	1.2	1.8	6.1	6.3	5.4	1.2	
NRB	2	1	1	1	0	5	
DMS A	0.2	0.1	1.1	0.4	1.1	1.1	No E1-3-1 Tolérance: 1.0% Temps CPU: 861s
SXS B	1.0	1.7	6.2	6.1	5.3	1.0	
NRB	2	1	2	0	1	6	
DMS A	0.1	0.1	0.1	0.1	0.1	0.1	No. E1-3-0 Tolérance: 0.0% Temps CPU: 1241s
SXS B	1.0	1.7	5.9	6.0	5.1	1.0	
NRB	2	2	2	1	1	8	

Tableau 5. Résultats du test no. E1-4

Étude X modifiée Point de congestion: DMS A							
Commutateur	Δ%					Max Δ%	Test
DMS A	0.9	1.0	0.3	0.9	0.2	0.9	No. E1-4-2 Tolérance: 2.0% Temps CPU: 805s
NRB	1	1	1	1	0	4	
DMS	0.2	0.4	0.6	0.1	0.6	0.6	No. E1-4-1 Tolérance: 1.0% Temps CPU: 809s
NRB	1	1	1	0	1	4	
DMS	0.1	0.1	0.1	0.1	0.1	0.1	No. E1-4-0 Tolérance: 0.0% Temps CPU: 1064s
NRB	1	1	1	1	1	5	

Tableau 6. Résultats du test no. E1-5

Étude X modifiée Point de congestion: SXS B							
Commutateur	Δ%					Max Δ%	Test
SXS B	1.1	1.7	2.7	1.5	0.7	1.7	No. E1-5-2 Tolérance: 2.0% Temps CPU: 634s
NRB	1	1	0	0	0	2	
SXS B	1.0	1.2	2.2	1.0	0.2	1.2	No. E1-5-1 Tolérance: 1.0% Temps CPU: 704s
NRB	1	2	0	0	0	3	
SXS B	1.0	1.0	2.1	0.8	0.0	1.0	No. E1-5-0 Tolérance: 0.0% Temps CPU: 854s
NRB	1	2	0	0	1	4	

Tableau 7. **Résultats du test no. E1-6**

Étude X modifiée **Points de congestion: DMS A et SXS B**							
Commutateur	**Δ%**					**Max Δ%**	**Test**
DMS A	0.5	0.6	0.8	0.6	0.8	**0.8**	No. E1-6-2 Tolérance: 2.0% Temps CPU: 989s
SXS B	1.1	1.3	2.9	2.0	1.5	**1.5**	
NRB	2	2	1	0	1	**6**	
DMS A	0.2	0.3	0.5	1.1	0.4	**1.1**	No. E1-6-1 Tolérance: 1.0% Temps CPU: 994s
SXS B	0.9	0.4	2.0	1.1	1.4	**1.4**	
NRB	2	1	1	1	1	**6**	
DMS A	0.1	0.1	0.1	0.1	0.1	**0.1**	No. E1-6-0 Tolérance: 0.0% Temps CPU: 1322s
SXS B	0.9	1.0	2.6	1.7	1.0	**1.0**	
NRB	2	2	1	1	2	**8**	

4.1.4 Description des tests avec l'étude Y

Cette deuxième série de tests avait pour but de confirmer ou d'infirmer les résultats obtenus avec l'étude X. L'étude Y couvre un sous-réseau plus grand que l'étude X. Puisque l'étude Y comporte plus de commutateurs, on veut aussi évaluer l'impact de l'interaction entre les commutateurs sur le nombre de recherches binaires. L'étude Y a été utilisée pour comparer les résultats entre l'ancienne version du noyau (c'est-à-dire avec la dichotomie ligne/liaison) et le nouveau noyau (avec la fonction Lambda). Voici la description des trois points de congestion (PC) ajoutés pour les tests.

Premièrement, un PC de 1900 CHANNELS dans la configuration 100 Term (Figure 28) a été rajouté. La configuration 100 Term modélise les ressources des liaisons d'un commutateur de type DMS-100. L'étude X possédait un PC similaire sauf que le PC dépendait du trafic des liaisons (Figure 26, p. 94) alors qu'ici le PC dépend du nombre de liaisons.

Un deuxième PC a été rajouté dans la configuration XBAR. Ce PC n'est pas illustré par une figure puisque la configuration XBAR est identique à la configuration SXS de la Figure 26 (p. 94). Le PC de la configuration XBAR, qui modélise un commutateur de type XBAR, est activé si le trafic des liaisons dépasse 9600 CCS. Ce PC est intéressant puisque le réseau d'accès, du commutateur qui possède ce PC, n'est composé que d'un échange. La fonction Lambda ne possède alors que très peu d'information.

Le troisième et dernier PC a servi à comparer l'ancienne et la nouvelle version du noyau. À la Figure 23 (p. 92), on peut voir que la configuration 100 Core de la base RCMB COU possède le noeud L ayant comme ressource DS30 LEFT. La configuration 100 Term (Figure 25, p. 93) de la même base contient le noeud A ayant comme ressource DS30 LEFT. Dans la base RCMB COM, le noeud DS30 RIGHT est

enlevé de la configuration 100 Term et transféré dans la configuration 100 Core. Afin de comparer les deux versions du noyau, on a défini un PC de 70 DS30 LEFT[1] (Figure 29, p. 103) dans la base COU pour l'ancienne version. De la même façon, on a défini un PC de 70 DS30 RIGHT (Figure 30, p. 103) dans la base RCMB COM.

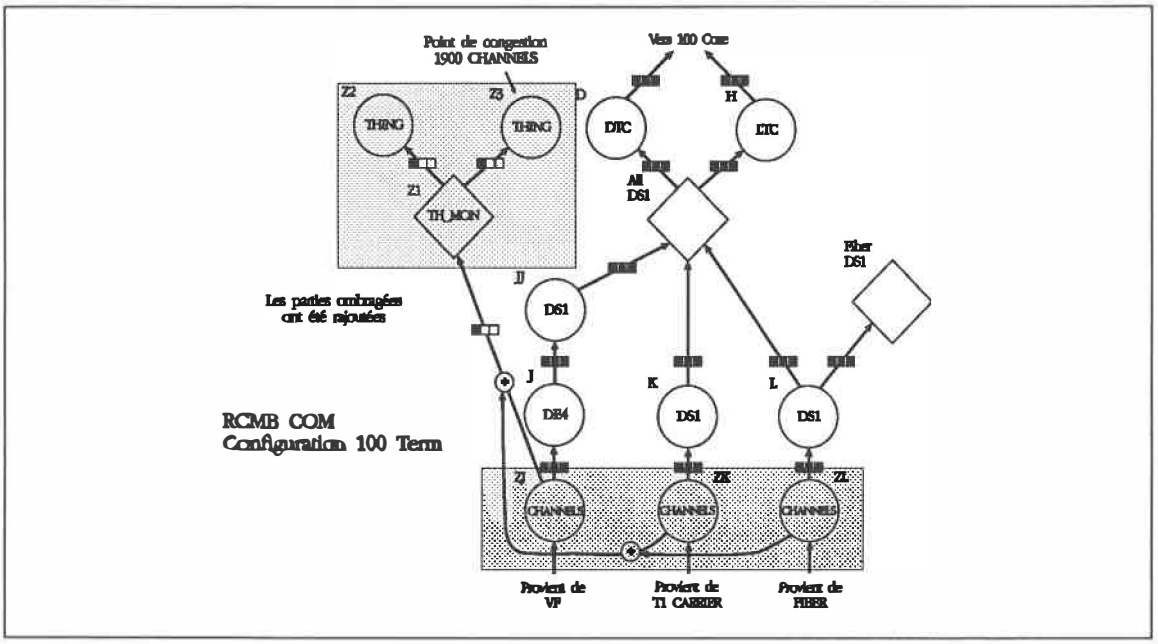


Figure 28. Point de congestion: 1900 CHANNELS (DMS-100)

4.1.5 Résultats de l'étude Y

Puisque l'ensemble des résultats obtenus avec l'étude Y occupent plusieurs pages et que l'information pertinente obtenue avec cette étude ne représente qu'une petite partie, il a été décidé de les présenter en annexe. On peut les retrouver à l'Annexe B, page 133.

1. Ce PC n'a comme seul but la comparaison des résultats des deux versions du noyau. Ce PC est artificiel dans le sens où rien n'indique qu'une telle contrainte existe (ou peut exister) pour un commutateur de type DMS-100. De plus, la valeur de DS30 est contrainte d'être un multiple de 16. Cette contrainte implique que lorsque le PC de 70 DS30 est utilisé, le multiple de 16 plus petit ou égal à 70 est 64, d'où un écart résiduel de 8.6% dans les résultats.

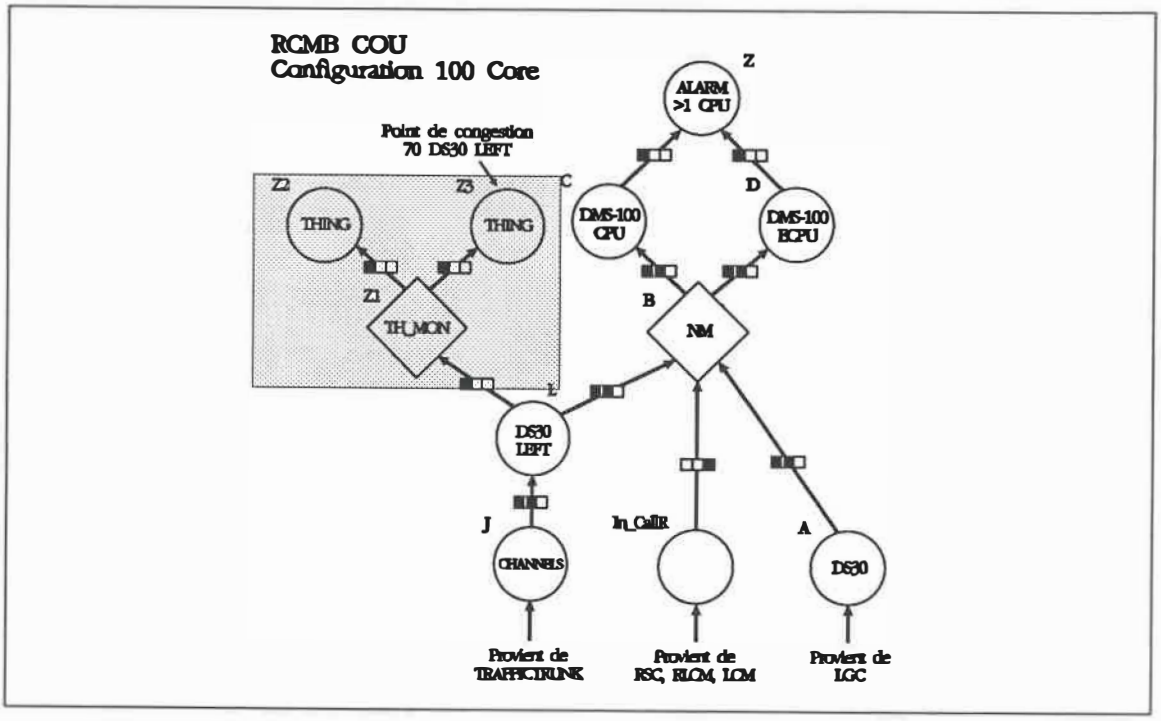


Figure 29. **Point de congestion: 70 DS30 LEFT (DMS-100)**

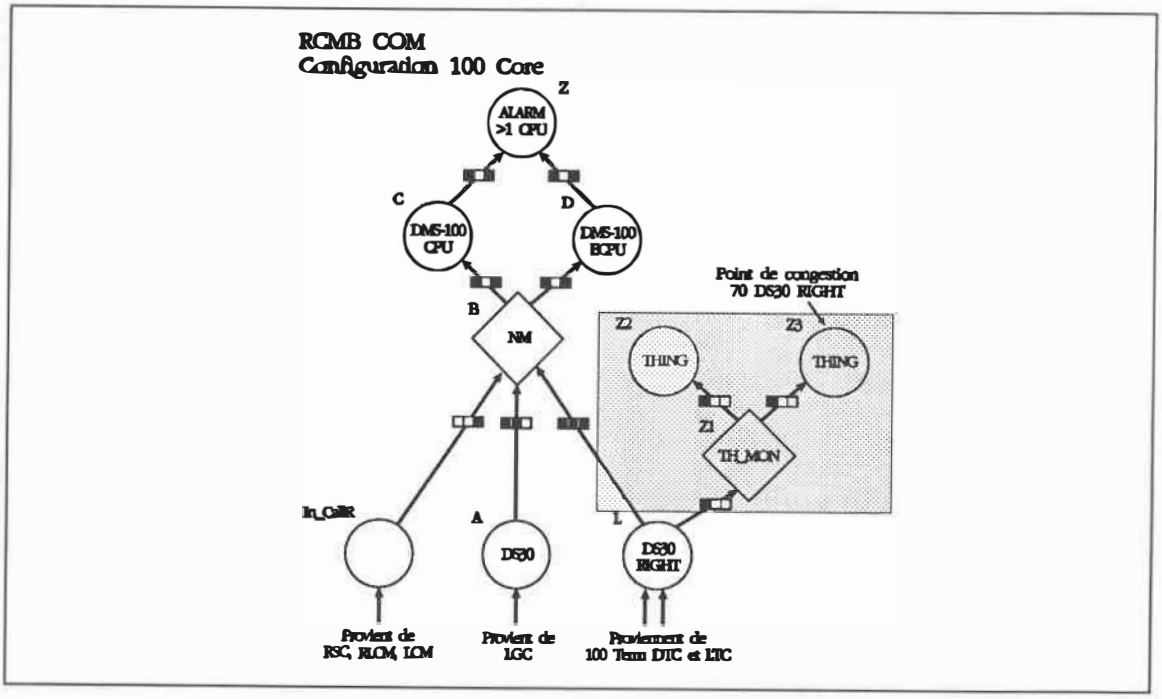


Figure 30. **Point de congestion: 70 DS30 RIGHT (DMS-100)**

4.2. Résultats de la deuxième étape

Pour tester la version du noyau incorporant le nouveau DRL/Lambda, une version modifiée de l'étude X a été utilisée. Pour ce faire, un tandem a été rajouté au centre des douze immeubles que comporte l'étude X[1]. De plus, chaque commutateur du réseau est défini comme "commutateur distant" de cet unique tandem. La méthodologie concernant cette dernière modification est donnée à la sous-section 2.3.1, page 70. En tout, trois tests se sont déroulés. Voici la description de ces tests.

Le premier test vise à comparer l'ancien DRL et DRL/Lambda. Nous avons vu que l'ancien DRL modélise un réseau logique où tous les commutateurs sont reliés entre eux par un lien de type direct final. On peut imiter l'ancien DRL avec DRL/Lambda en initialisant tous les éléments de la matrice des types de liens avec le code DF (Direct Final). De cette façon, DRL/Lambda installe un lien de type direct final entre toutes les paires de commutateurs ayant un trafic intercommutateur non nul. Contrairement à l'ancien DRL, où la modularité des liens est supposée égale à un, DRL/Lambda approvisionne les liens en installant le nombre minimal de modules couvrant le nombre de liaisons voulu. Les résultats de ce premier scénario sont comparés au scénario où tous les liens entre les commutateurs sont déclarés de type PHU. À chaque commutateur, le nombre de liaisons calculés pour le deuxième scénario doit être égal ou légèrement inférieur[2] au nombre calculé pour le réseau de liens direct finaux. Les résultats de ce premier test sont présentés au Tableau 8, à la page 105.

1. Plus exactement, les coordonnées de l'immeuble qui abrite le tandem sont les moyennes arithmétiques des coordonnées en x et y des douze immeubles. On utilise dans les réseau réels un calcul similaire dont les coordonnées sont pondérées par le trafic généré à l'immeuble. Ce calcul est similaire du centre de masse d'un objet.
2. On doit cependant tenir compte de certains facteurs qui peuvent générés plus de liaisons. Ces facteurs sont donnés au Chapitre 5.

Tableau 8. Résultats du test no. E2-1

Comparaison du nombre de modules installés entre DRL/SNAP et DRL/Lambda												
	Comm.	1	2	3	4	5	6	7	8	9	10	Total
Année 0	DRL SNAP	9	55	22	7	82	71	29	6	23	28	332
	DRL Lambda	10	56	20	7	79	69	28	6	23	27	324
	Δ	+1	+1	-2	0	-3	-2	-1	0	0	-1	-8
Année 1	DRL SNAP	10	60	23	7	92	74	32	8	27	29	362
	DRL Lambda	11	61	22	8	90	73	30	7	28	29	360
	Δ	+1	+1	-1	+1	-2	-1	-1	-1	+1	0	-2
Année 2	DRL SNAP	11	65	25	9	105	77	37	8	27	32	396
	DRL Lambda	12	65	24	9	103	76	34	7	28	31	389
	Δ	+1	0	-1	0	-2	-1	-3	-1	+1	-1	-7

Le deuxième test vise à évaluer le comportement du DRL/Lambda lorsqu'un tandem congestionne. Un PC de 1368 CHANNELS (ou liaisons) a été défini pour l'unique tandem. Ce PC n'est pas illustré puisque qu'il est semblable à celui de la Figure 28 (p. 102), à l'exception que la configuration s'appelle TAN Term au lieu de 100 Term et que les noeuds des liens sont doublés. Le Tableau 9 (p. 106) contient les résultats obtenus, avec le même scénario et le PC de 1368 CHANNELS, pour 6 tolérances différentes, soit de 0 à 5% par incrément de 1%.

Tableau 9 **Résultats du test no. E2-2**

Étude X avec tandem Point de congestion: 1368 CHANNELS (tandem)							
Tolérance (%)		0	1	2	3	4	5
Temps CPU (s)		525	311	342	341	341	349
Nombre RB		2	1	1	1	1	1
Demande non satisfaite	**BUSNS**	324	324	324	324	324	324
	POTS	1954	1561	1941	1941	1941	1941
Nombre de liaisons	**T1F IN**	667	685	667	667	667	667
	T1F OUT	653	659	653	653	653	653
Δ%		3.5	1.8	3.5	3.5	3.5	3.5
Δ module		2	1	2	2	2	2

Puisque DRL/Lambda utilise un programme en nombre entier, il est probable que les résultats obtenus présentent des "sauts", c'est-à-dire des résultats qui ne changent pas de façon continue mais par bond dépendemment des coûts et des contraintes. Le troisième test vise à évaluer ce comportement. Ce dernier test utilise un PC similaire au deuxième test sauf que le maximum est de 1344 CHANNELS, soit un module de moins (24 liaisons). Le Tableau 10 (p. 107) contient les résultats de ce test.

Tableau 10. **Résultats du test no. E2-3**

Étude X avec tandem Point de congestion: 1344 CHANNELS (tandem)							
Tolérance (%)		0	1	2	3	4	5
Temps CPU (s)		410	352	328	341	341	349
Nombre RB		1	1	1	1	1	1
Demande non satisfaite	**BUSNS**	324	370	370	370	576	576
	POTS	2712	3100	3544	3544	4226	4226
Nombre de liaisons	**T1F IN**	669	613	637	637	612	612
	T1F OUT	651	611	635	635	612	612
Δ%		1.8	8.9	5.4	5.4	8.9	8.9
Δ module		1	5	3	3	5	5

5. Discussion et conclusion

Le présent chapitre contient l'analyse et la discussion des résultats du Chapitre 4. L'analyse des résultats est scindée en deux parties, soit une partie pour chaque étape de la solution. L'analyse des résultats, qualitative et quantitative, de la première étape est présentée à la première section. La deuxième section contient l'analyse des résultats de la deuxième étape. La troisième section décrit les tests et améliorations, à apporter dans la modélisation des tandems, ainsi que les avenues possibles dans la modélisation des commutateurs du réseau interurbain[1]. Enfin, la quatrième et dernière section contient la conclusion de ce mémoire.

5.1. Analyse des résultats de la première étape

La présente section est divisée en deux sous-sections. La première sous-section décrit qualitativement comment l'algorithme d'approvisionnement procède à la recherche de la charge maximale admissible, à l'aide de la fonction TRAFFICTRUNK dans l'ancienne version de SNAP, et à l'aide de la fonction Lambda dans la nouvelle version. La deuxième sous-section contient l'analyse quantitative des résultats obtenus à l'aide des études X et Y.

5.1.1 Analyse qualitative

Pour bien comprendre la portée des tests qui ont servis à évaluer la fonction Lambda, il faut connaître le comportement de l'algorithme d'approvisionnement de façon qualitative. Cette sous-section est divisée en trois parties. La première partie décrit le comportement de l'algorithme d'approvisionnement de l'ancienne version du noyau, avec la fonction TRAFFICTRUNK. Ensuite, les deux autres parties décrivent respectivement les deux cas de déviation de la fonction Lambda prototype, soit la sous-estimation et la sur-estimation.

1. C'est-à-dire les commutateurs de Classe 1 à la Classe 4.

La fonction TRAFFICTRUNK

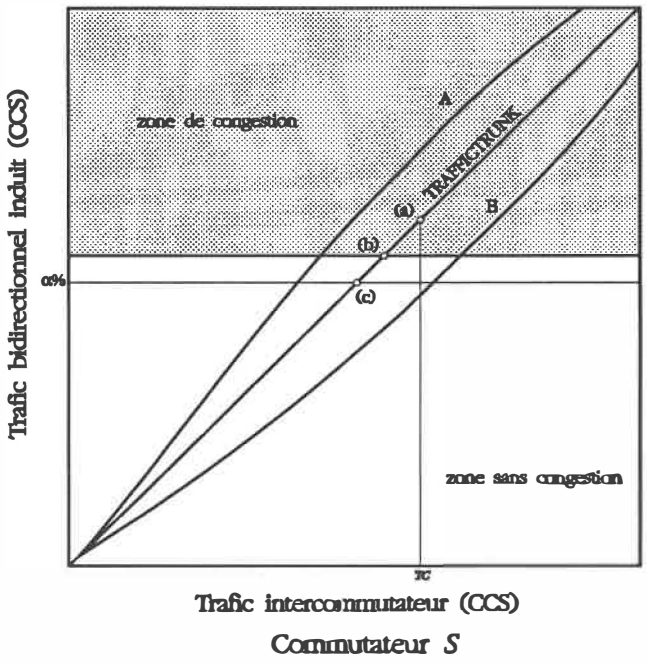
Nous avons vu que le noeud TRAFFICTRUNK estime le trafic des liaisons pour le côté gauche de la base RCMB. L'estimation suppose que le trafic intercommutateur total est égal au trafic dans les deux sens (p. 28) donné par le modèle gravitationnel (MG). Tous les commutateurs principaux (CP) sont ainsi isolés les uns des autres lors de l'éxécution de l'algorithme d'approvisionnement. Afin de bien comprendre le comportement des fonctions TRAFFICTRUNK et Lambda, nous illustrerons graphiquement la recherche de la charge maximale admissible (CMA), d'un commutateur du réseau simulé, avec l'algorithme d'approvisionnement. Le graphique de la Figure 31 illustre un cas de recherche binaire avec la fonction TRAFFICTRUNK.

Pour étudier la recherche de la CMA, on utilisera un réseau fictif[1]. On suppose que le réseau comporte un commutateur, nommé *S*, et que c'est le seul commutateur qui subit une congestion. De plus, le trafic du commutateur *S* ne provient que d'un seul échange (nous verrons les conséquences plus loin, avec la fonction Lambda).

À chaque charge du réseau d'accès correspond un trafic intercommutateur, calculé par le module Trafficker (p. 28). C'est ce trafic qui est représenté en abscisse. On suppose que la base RCMB contient un point de congestion de x CCS. Tout trafic intercommutateur plus grand que cette valeur cause une congestion, donc le déclenchement d'une recherche binaire. C'est cette valeur de trafic qui scinde le graphique en zone de congestion et zone sans congestion.

1. Ce réseau est composé de façon à ce que l'étude du comportement de l'algorithme d'approvisionnement soit simplifiée. Cette simplification ne diminue en rien la validité de l'explication pour les cas plus complexes.

Figure 31. **Recherche binaire avec la fonction TRAFFICTRUNK**



Dans l'exemple, le module Assigner produit des ordres d'installation de telle façon que le trafic intercommutateur généré atteint *TC* CCS. Puisque le point (a) est situé dans la zone de congestion, le module Compressor doit trouver la CMA. Graphiquement, l'algorithme d'approvisionnement se comporte comme un algorithme de bissection qui chercherait la racine unique de la fonction TRAFFICTRUNK (en prenant comme abscisse la transition des deux zones).

Avec une tolérance de α%, n'importe quelle charge, qui produit un trafic intercommutateur compris entre les points (c) et (b) inclusivement, peut être trouvée comme CMA par le module Compressor. Puisque l'ancienne version ne permet les points de congestion que du côté gauche de la base RCMB, seule la droite TRAFFICTRUNK sert à la recherche de la CMA. La fonction TRAFFICTRUNK étant fixe, il est possible d'obtenir des résultats avec 0% d'erreur. Une recherche binaire avec 0% de tolérance trouverait le point (b).

À la même Figure 31, les courbes A et B illustrent deux courbes possibles du trafic bidirectionnel, selon le MG, induit par le trafic intercommutateur en abscisse. On constate que la droite d'approximation de TRAFFICTRUNK sur-estime le trafic bidirectionnel de la courbe B. Cette situation se présente généralement lorsqu'un commutateur est relié à des commutateurs plus "petits" que lui, soit des commutateurs générant moins de trafic intercommutateur[1]. La courbe A illustre le cas inverse, soit un petit commutateur relié à des commutateur plus importants.

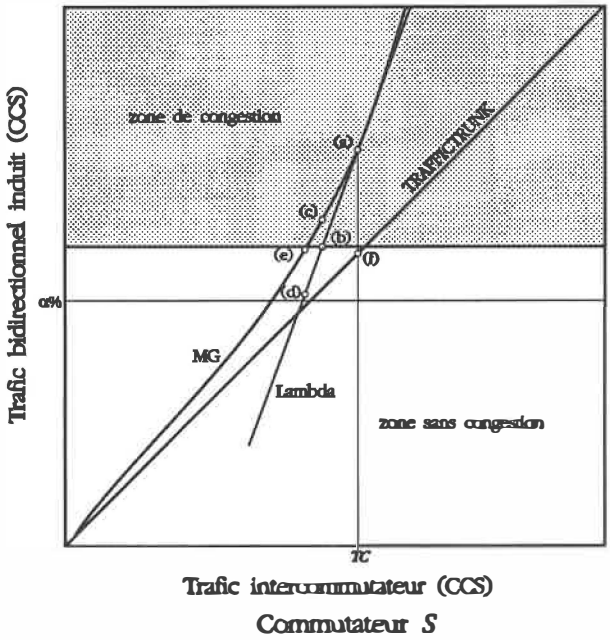
On constate cependant que l'utilisation de la fonction TRAFFICTRUNK introduit une sur-estimation ou une sous-estimation du trafic calculé par le MG pour les liaisons. Les résultats obtenus avec 0% de tolérance n'ont de signification que si l'on tient compte exclusivement de l'approximation donné par le noeud TRAFFICTRUNK, c'est-à-dire en faisant fi du trafic calculé pour le côté droit par le module Trunker. Nous verrons que l'approximation plus fine faite par la fonction Lambda introduit deux types de complications qui n'existent pas avec la fonction TRAFFICTRUNK.

La fonction Lambda: sous-estimation

La fonction Lambda est une approximation linéaire par morceaux du trafic calculé par le MG. Cette approximation possède la propriété de passer par le point de trafic bidirectionnel donné par le MG au point d'estimation[2]. Pour illustrer la sous-estimation (Figure 32), nous supposerons le même réseau avec le commutateur *S* décrit précédemment. Les écarts entre les courbes aux Figures 32 et 33 sont délibérément amplifiés pour mieux illustrer le propos. Puisque le trafic du réseau d'accès n'est composé que d'un seul échange pour le commutateur *S*, la fonction Lambda se réduit à une simple droite (voir les équations p. 44).

1. Ce phénomène découle de la moyenne arithmétique des trafics des deux sens évaluée par DRL/SNAP.
2. C'est le cas analogue au développement de Taylor d'une fonction où l'approximation est égale à la fonction au point d'évaluation.

Figure 32. Recherche binaire avec la fonction Lambda: sous-estimation



La courbe MG est obtenue de la façon suivante: le reste du réseau étant fixe, on calcule le trafic bidirectionnel induit (selon le MG) par le commutateur *S* lorsque l'on varie sa charge. Cette courbe n'existe pas dans le sens où la boucle Compressor n'utilise jamais les fonctions du MG pour le calcul du trafic intercommutateur.

Toujours à l'aide du réseau fictif, le module Assigner commence l'algorithme en produisant des ordres d'installation tels que le trafic intercommutateur généré atteint *TC* CCS. À l'initialisation de la boucle Compressor, le module Trunker modifié est appelé (voir p. 53) pour calculer les coefficients nécessaires à la fonction Lambda. L'approximation de la fonction Lambda (courbe Lambda) passe par le point (a) donné par la courbe MG avec la tentative de charge *TC*, comme mentionné. Seule la fonction Lambda est utilisée par l'algorithme de recherche binaire dans le calcul de la charge maximum. Si la tolérance est de 0%, le point (b) est trouvé comme étant le trafic intercommutateur maximum que peut admettre le commutateur.

Lors du deuxième appel de la boucle Compressor, les coefficients de la fonction Lambda sont remis à jour (puisqu'il y a eu congestion) et la droite passe maintenant par le point (c)[1] qui est dans la zone de congestion. Il y aura donc une deuxième recherche binaire avec la fonction Lambda et les nouveaux coefficients. Cette deuxième recherche découle directement de la sous-estimation faite par la fonction Lambda du trafic donné par la courbe MG.

Cette situation est particulièrement préoccupante lorsque la tolérance est petite. Dans le cas où la tolérance est plus élevée, par exemple α%, la boucle Compressor pourrait trouver le point (d) dans un premier temps. Après le rafraîchissement de la fonction Lambda, la nouvelle droite d'approximation passerait par le point (e) qui n'est pas dans la zone de congestion.

La droite TRAFFICTRUNK nous donne l'approximation du trafic bidirectionnel fait par la fonction du même nom dans l'ancienne version du noyau. On constate, dans le cas illustré, que l'utilisation de cette approximation n'aurait engendré aucune congestion puisque le point (f), c'est-à-dire le trafic bidirectionnnel induit par la tentative de charge *TC*, n'est pas situé dans la zone de congestion.

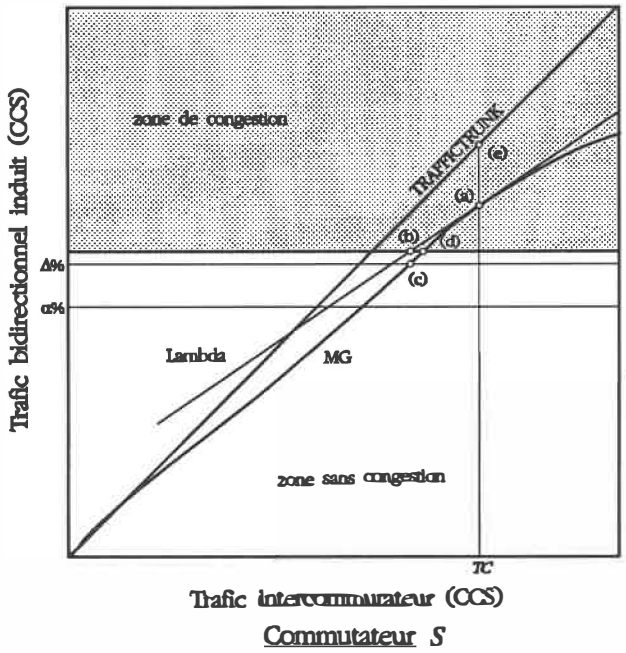
Nous venons de voir que de la sous-estimation de la fonction Lambda découle une augmentation du nombre de recherches binaires. L'autre cas de déviation, la sur-estimation, introduit un autre type de complication.

La fonction Lambda: sur-estimation

Si la fonction Lambda, dans certaines circonstances, sous-estime le trafic donné par le MG, elle le sur-estime dans d'autres. Toujours avec le même exemple, la Figure 33 illustre un cas de sur-estimation de la fonction Lambda. Voyons comment se déroule l'algorithme d'approvisionnement.

1. Cette deuxième droite n'est pas illustrée.

Figure 33. Recherche binaire avec la fonction Lambda: sur-estimation



Le module Assigner génère des ordres d'installation qui sont transférés au module Compressor. Le module Compressor à son tour appelle le module Trafficker qui calcule le trafic intercommutateur *TC*. Le module Trunker est ensuite appelé pour calculer les coefficients de la fonction Lambda. Puisque le point (a) est situé dans la zone de congestion, le module Compressor déclenche une recherche binaire pour trouver la CMA. Si la tolérance démandée est de 0%, le point (b) est la solution retournée par la recherche binaire.

Lorsque les coefficients de la fonction Lambda sont rafraîchis, la nouvelle droite d'estimation (non illustrée) passe par le point (c). Aucune recherche binaire n'est déclenchée puisque le point (c) est situé dans la zone sans congestion. Cependant, on constate que la charge pourrait être augmentée jusqu'à ce que le trafic inter-commutateur atteigne le point (d), pour obtenir la CMA théorique avec 0% de tolérance.

La différence entre la CMA théorique et la CMA trouvée dans l'exemple est de Δ%. Nous appellerons cette différence écart résiduel. Ce problème est plus grave que celui des recherches binaires supplémentaires induites par la sous-estimation. L'algorithme, dans sa forme actuelle, ne peut garantir que l'écart résiduel respecte un pourcentage donné. Les tests avaient donc pour but, entre autres, de montrer si oui ou non il fallait rajouter un mécanisme de contrôle de l'écart résiduel.

Les tests faits avec les études X et Y avaient aussi pour but d'évaluer l'impact de la sous-estimation et la sur-estimation de la fonction Lambda. De plus, un test avec l'étude Y devait nous permettre d'évaluer le cas, beaucoup plus complexe, où plusieurs commutateurs interagissent entre eux, lors de congestions simultanées.

5.1.2 Analyse quantitative

Cette sous-section contient l'analyse quantitative, en deux volets, des résultats du Chapitre 4 pour la première étape (fonction Lambda prototype). La première partie contient l'analyse de la précision des résultats en fonction de la tolérance. La deuxième partie contient l'analyse de l'impact de la tolérance sur le temps de calcul et le nombre de recherches binaires.

Précision des résultats

Le Tableau 11 (p. 116) contient les écarts résiduels Δ% maximum, pour le DMS A , obtenus avec l'étude X originale et modifiée.

Avec 2% de tolérance, le plus grand Δ% pour le DMS A, est de 0.9%, soit bien en deça du maximum permis. Avec 1% et 0% de tolérance, on obtient respectivement 1.1% et 0.1%. Il faut noter que le DMS A possède beaucoup d'information pour les coefficients Δ_{nt}^{i} de la fonction Lambda puisque son trafic provient de plusieurs échanges et que des commutateurs distants lui sont rattachés.

Tableau 11. Écart résiduel Δ% maximum pour le DMS A

Étude X	Tolérance		
	2%	1%	0%
originale	0.8	1.1	0.1
modifiée	0.9	1.1	0.1
Maximum	0.9	1.1	0.1

Pour le SXS B (Tableau 12), les Δ% maximum sont plus élevés, sûrement à cause de la "pauvreté" des Λ_{ne}^{i}. Comme dans l'exemple du commutateur *S* (p. 109), le trafic du commutateur SXS B ne provient que d'un seul échange. Avec 2%, 1% et 0% les Δ% maximum sont respectivement 1.7%, 1.4% et 1.0%.

Tableau 12. Écart résiduel Δ% maximum pour le SXS B

Étude X	Tolérance		
	2%	1%	0%
originale	1.7	1.2	1.0
modifiée	1.7	1.4	1.0
Maximum	1.7	1.4	1.0

On constate qu'avec 2% de tolérance, les Δ% restent en deça de 2%. Ce résultat est important puisque Recherches Bell-Northern compte préconiser l'usage de la plage 2%-4% à ses clients. Les raisons de ce choix sont simples. Premièrement, le temps de calcul devient rapidement prohibitif en deça de 1% de tolérance. Deuxièment, SNAP est un outil de planification et non d'approvisionnement. Il n'y

a donc aucun avantage à augmenter le temps de calcul (jusqu'à 62% entre les tests no. E1-1-2 et no. E1-1-0) pour, dans certains cas, n'obtenir que des différences marginales du nombre d'usagers pour un commutateur. C'est pour confirmer ou infirmer ces résultats que des tests avec l'étude Y ont eu lieu.

Le Tableau 13 contient les écarts résiduels Δ% maximum, pour les commutateur de type DMS et le commutateur de type XBAR, obtenus avec l'étude Y. Les écarts résiduels des tests no. E1-5 et E1-6, pour les commutateurs de types DMS, ne sont pas significatifs à cause de la raison donnée à la page 102.

Tableau 13. Écart résiduel Δ% maximum pour les DMS et le XBAR

Étude Y No. du test	Tolérance		
	2%	1%	0%
2 (DMS)	0.9	0.5	0.3
3 (XBAR)	1.2	1.0	1.0
4 (XBAR) (DMS)	1.2 0.9	1.0 0.7	1.0 0.3
5 (DMS)	8.6	8.6	8.6
6 (XBAR) (DMS)	1.2 8.6	1.0 8.6	1.0 8.6
Maximum **XBAR** **DMS**	1.2 0.9	1.0 0.7	1.0 0.3

On constate, comme avec les tests avec l'étude X, que l'écart résiduel reste en deça de la tolérance pour 2% (et 1% aussi dans ce cas). Cette deuxième série de tests confirme les résultats de la première série et rend inutile l'ajout d'un mécanisme supplémentaire pour contrôler l'écart résiduel.

Temps de calcul et nombre de recherches binaires

On constate qu'il y a peu de différence, entre les temps de calcul pour 2% et 1% de tolérance, avec l'étude X (6.7% en moyenne). Il faut noter que dans tous les tests avec l'étude X, sauf le test no. E1-4, l'écart résiduel est plus petit pour 2% que pour 1% de tolérance pour le DMS A.

Le nombre de recherches binaires, avec l'étude X, ne dépasse pas le nombre de 2 pour une même année. Ce nombre réduit de recherches binaires s'explique par le fait que deux commutateurs au plus se trouvaient congestionnés la même année. On remarque aussi que, comme prévu, le nombre de recherches est inversement proportionnel à la tolérance demandée.

Encore une fois, avec l'étude Y, on note que les temps de calcul entre 2% et 1% de tolérance sont relativement près (4.4% en moyenne). Le temps de calcul et le nombre de recherches binaires augmentent rapidement avec des tolérances en deça de 1%. De plus, le nombre de recherches binaires augmente lorsque plusieurs commutateurs sont simultanément congestionnés comme dans les tests no. E1-8, E1-10, E1-11 et E1-12 (on peut consulter les résultats de l'étude Y aux pp. 133-150). Le nombre maximal de recherches binaires enregistré pour une même année est de 4. Il faut mentionner que ce nombre a été obtenu lorsque 3 ou 4 commutateur de type DMS-100 ont simultanément congestionnés. Cette situation n'est pas commune dans les études de planification réelles, le maximum étant de 1 ou 2 commutateurs.

La plage de tolérance 0%-1% est ***inutilisable*** à cause du temps de calcul qui devient trop grand et du manque de borne de l'écart résiduel. Mais pour les raisons données à la page 116, cette plage n'est pas importante et sera progressivement délaissée pour la plage de 2-4%. Tous les résultats montrent que la fonction Lambda prototype réagit très bien avec une tolérance de 2% et que l'écart résiduel reste toujours en deça de 2%.

On constate que tous les temps de calculs avec l'étude Y, à l'exception des tests no. E1-7 et E1-12, sont moins élevés pour la nouvelle version. Ce fait s'explique évidemment par la tolérance plus élevée utilisée avec la nouvelle version du noyau. Ceci étant dit, si l'on compare les temps de calcul du test no. E1-7, où les deux versions ont été testées avec 0% de tolérance avec l'étude Y originale, la différence est de 2.2% plus de temps pour la nouvelle version. On constate donc que, pour des études avec très peu de congestion, le temps supplémentaire de calcul pour la nouvelle version est à toute fin négligeable.

5.2. Analyse des résultats de la deuxième étape

La présente section contient l'ensemble des résultats obtenus avec le noyau de SNAP qui intègre DRL/Lambda et la modélisation des tandems. La première sous-section contient la comparaison entre l'ancien réseau de liaison (réseau logique) et le nouveau réseau (réseau physique) modélisé par DRL/Lambda. La deuxième sous-section contient l'analyse des résultats des deux autres tests. La troisième et dernière sous-section analyse le temps de calcul supplémentaire et la compacité de DRL/Lambda.

5.2.1 Comparaison des deux réseaux de liaisons

Nous avons vu à la section 1.2.4 (p. 33) que DRL/SNAP modélise un réseau logique de liaisons, c'est-à-dire un réseau où tous les commutateurs sont interconnectés par des liens de type direct final (définition p. 33). On sait aussi que l'utilisation de ce type de réseau logique était possible du fait que SNAP ne modélisait que des commutateurs de Classe 5. On s'attend donc à ce qu'il y ait peu de différence pour les commutateurs de Classe 5 dans le nombre de liaisons installées.

Effectivement, l'analyse du Tableau 8 à la page 105, nous montre que le nombre de modules calculés par DRL/SNAP et DRL/Lambda diffèrent très peu. Lorsque DRL/Lambda a calculé des modules en excès, la différence n'est que d'un module. Cette différence provient de deux sources. Premièrement, les commutateur qui ne possèdent que des liens direct finaux dans avec DRL/SNAP ont, avec DRL/Lambda, un lien final avec le tandem. Les liens direct finaux avec DRL/Lambda deviennent des liens PHU. Le commutateur 1 entre autre ne possède qu'un lien avec DRL/SNAP. La deuxième source provient de l'hypothèse utilisée pour le BIP (p. 68) qui néglige l'interaction entre les liens PHU. L'erreur d'un seul module est donc acceptable.

On constate que, d'une façon globale, DRL/Lambda installe moins de modules que DRL/SNAP. Ceci s'explique évidemment par la concentration du trafic dans les liens finaux. L'analyse de ce premier test nous montre que les résultats produits par DRL/Lambda sont acceptables (du moins pour le nombre de liaisons calculés aux commutateurs de Classe 5).

5.2.2 Précision des résultats avec congestion du tandem

L'analyse du Tableau 9 (p. 106) nous montre la principale difficulté dans la recherche de la charge maximale admissible avec un DRL différent de DRL/SNAP: l'écart résiduel. On constate que l'écart résiduel est toujours plus grand que la tolérance demandée. Plusieurs raisons expliquent ce résultat. Premièrement l'écart

entre la fonction Lambda et le trafic calculé par le modèle gravitationnel (voir pp. 111-115). Deuxièment, il y a l'hypothèse de la fonction Lambda que la baisse de charge d'un commutateur se répercute proportionnellement dans ses liens PHU et son lien final (équations (2.36) à (2.39)).

Il y a aussi un phénomène d'instabilité pour la plage de tolérance 0%-2%. Par exemple, dans le test no. E2-1, l'écart résiduel avec 0% est de 3.5% (2 modules), baisse à 1.8% (1 module) pour 1% pour ensuite remonter et se stabiliser à 3.5% pour 2% de tolérance et plus. Il est impossible ici de décrire exactement l'interaction entre tous éléments en jeux. Pour avoir une idée précise et exacte de ce qui se passe, il faudrait premièrement suivre la recherche binaire pas à pas. Deuxièmement, il faudrait ensuite comparer les trafics calculés à la fin de boucle de Compressor avec ceux à l'initialisation de la deuxième boucle. Finalement, il faudrait analyser l'impact du BIP sur le partage des trafics PHU et final. Bref, l'analyse de ce phénomène est une entreprise très difficile. De plus, il est probable cet effet dépende trop des données d'entrée pour obtenir des résultats s'appliquant à tous les cas. Enfin, puisque la fonction Lambda doit être utilisée avec des tolérance de 2% et plus, l'étude appronfondie de ce phénomène n'a pas été entreprise.

Avec les Tableaux 9 et 10, on constate que les écarts résiduels pour les tests no. E2-1 et E2-2 avec 2%-3% de tolérance sont respectivement de 3.5% et 5.4%. Si la modularité des liens était de 1 et que le nombre de liaisons à installer était dicté par la seule procédure ECCS, le réseau de liaisons varierait de façon continue. Par contre, avec l'adjonction du BIP, on assiste à des variations discontinues du nombre de module pour le tandem. Ces variations ne sont pas uniques au BIP; les algorithmes DRL sont des algorithmes heuristiques qui font un travail analogue, d'une façon plus sophistiquée et précise, au programme en nombre entier de DRL/Lambda. Il y donc tout lieu de penser que cette situation serait présente avec n'importe quel algorithme de type DRL.

5.2.3 Temps de calcul et compacité

L'implantation de DRL/Lambda est faite en deux modules. Le premier module, appelé LCM (Logical Channel Mapping), contient l'ensemble des procédures décrite dans le déroulement en pseudo code de la page 68, sauf la résolution du BIP. La résolution du BIP, qui est préparé par le module LCM et passé en paramètre, est effectué par le module appelé BIP. Ce dernier module transforme le problème sous forme standard et ensuite le solutionne numériquement en utilisant l'algorithme décrit à la section 3.1. Le module LCM compte 2700 lignes et le module BIP environ 1700 lignes (sans lignes de commentaire). Ce nombre de lignes comprend l'ensemble des procèdures qui ont servis à déverminer les modules et les procédures de vérification. En optimisant les procédures et en enlevant les procédures de déverminage et de vérification, il est probablement possible de réduire la taille des deux modules d'au moins 10-15%. DRL/Lambda est donc plus ou moins compact.

L'ensemble des données nécessaires à DRL/Lambda, pour les test no. E2-1, E2-2 et E2-3, tient dans un fichier de 73 lignes de texte. Ces données ont été rajoutées à la fin du fichier, appelé GENERAL, qui contient l'année de base de l'étude, le nombre d'année que compte l'étude et la tolérance demandée. On constate que DRL/Lambda nécessite peu d'information supplémentaire.

Pour calculer le temps moyen de calcul de l'ensemble LCM-BIP on se sert des résultats du test no. E2-1. Lorsque DRL/Lambda "imite" DRL/SNAP, seul le module LCM est exécuté. De plus, on estime que le temps de calcul du module LCM dans ce mode est comparable à celui de DRL/SNAP. La différence, entre le temps de calcul de DRL/Lambda avec un réseau de liens PHU et de DRL/Lambda avec des liens directs finaux, nous donne approximativement le temps total requis par les modules LCM et BIP. Ces deux modules sont exécutés 6 fois[1]. La différence de temps observée a été de 154s; on obtient un temps moyen de 26s par itération de LCM-BIP.

1. Une fois par année que compte l'étude plus une fois à l'année de base.

La plus grande partie des 26s est passée à résoudre le problème en nombre entier. Il faut se rappeler que le module BIP initialise le problème en solutionnant la relaxation du problème à l'aide d'un algorithme simplex primal. L'algorithme simplex (primal ou dual) n'est pas un algorithme polynomial, cela même si des études statisques ont montré que l'espérance du temps de calcul est borné par un polynôme en m et n (où m et n sont respectivement le nombre de contraintes et le nombre de variables[1]). De plus, l'algorithme simplex dual est exécuté à chaque génération de coupes et à chaque noeud de l'algorithme *branch-and-bound*. Finalement, à chaque génération de coupes, il faut exécuter un problème de type sac de campeur 0-1 pour chacune des m contraintes.

Étant donné que nous ne disposons pas des temps de calculs pour les logiciels DRL existants, il est difficile de déterminer si DRL/Lambda est un algorithme rapide. Il est possible de diminuer le temps de calcul de 10-20% en enlevant les procédures de vérification. Certaines de ces procédures sont exécutées pour chacune des étapes des algorithmes simplex et ce pour toutes les variables.

5.3. Tests et améliorations à apporter

Afin de vraiment montrer vraiment si DRL/Lambda est un algorithme viable, d'autres tests doivent être faits. Premièrement, on devra soumettre DRL/Lambda à une série de tests tirés d'études réelles de planification avec des tandems. De cette façon, on pourra constater si le logiciel SNAP, avec l'addition de DRL/Lambda, produit des résultats s'approchant de la solution optimale trouvée par les outils standard. On devra tester si DRL/Lambda produit des matrices de liaisons qui s'approche suffisamment de celles produites par les DRL existants.

1. Voir Nemhauser et Wolsey, p. 122.

Certaines questions restent par contre en suspend. Par exemple, est-ce qu'il est souhaitable qu'un tandem qui est congestionné réduise la charge sur les commutateurs qui lui sont rattachés? Cette question, qui sort du cadre de ce travail, devra être répondue (probablement par les planificateurs). Si oui, on devra déterminer comment le logiciel doit procéder pour réduire la charge. Dans le cas où il n'est pas souhaitable qu'une telle procédure se produise, une solution possible consisterait à ajouter des contraintes au BIP de DRL/Lambda afin que le nombre de liaisons se terminant au tandem ne dépasse pas une certaine valeur.

Dans les logiciels DRL existants on utilise, pour les matrices de trafics, les matrices de trafics entre les immeubles et non entre les commutateur, comme on fait avec DRL/Lambda[1]. Si DRL/Lambda doit effectivement être utilisé dans ce cadre, on peut le modifier et intégrer la notion de commutateur virtuel[2]. Le commutateur virtuel est un groupe de commutateur rassemblés pour n'en former qu'un seul pour les autres commutateurs. Une fois que l'on connait le nombre de liaisons entre les commutateurs, il ne reste qu'à distribuer les liaisons (inter-immeuble) proportionnellement entre les différents commutateurs et à calculer le nombre de liaisons (intra-immeuble) entre les commutateurs qui forment le commutateur virtuel.

Si DRL/Lambda est une solution viable, il est probablement possible de le généraliser pour les commutateurs du réseau interurbains (par exemple pour les liens bidirectionnels). Il faudra toutefois vérifier si la taille du problème en nombre entier ne rend pas DRL/Lambda très désavantagé par rapport à des algorithmes DRL standards, puisque les programmes en nombres entiers nécessitent beaucoup de temps de calcul, et ce même pour de petite instance de ceux-ci.

1. Dans les réseau de télécommunications, les liaisons sont rajoutés entre les immeubles. Ensuite, à l'aide d'un Digital Cross Connect, on les partage entre les commutateurs de l'immeuble de façon à maximiser l'utilisation de celles-ci.
2. Les commutateurs virtuels sont partiellement définis dans la version courante de DRL/Lambda.

Enfin il faudrait optimiser les calculs effectués dans le module LCM[1] et, comme mentionné précédemment, enlever les procédures de vérification dans le module BIP. Une amélioration du temps de calcul de 10-20% est envisageable avec les optimisations simples décrites.

5.4. Conclusion

Les résultats obtenus avec les deux noyaux modifiés de SNAP montrent qu'il est possible de repousser les limites de modélisation de la version actuelle de ce logiciel. Par contre, on constate que l'on atteint une frontière où l'implantation de nouveaux algorithmes, qui rendent les commutateurs dépendants du reste du réseau (comme les DRL), ne peut se faire qu'avec un compromis de la précision des résultats.

La méthodologie de la première étape est applicable immédiatement puisqu'elle rencontre l'objectif de précision voulu, soit la fourchette de tolérance de 2% et plus. De plus, l'implantation de la méthodologie ne nécessite que des modifications mineures tandis que le temps de calcul supplémentaire est acceptable, sinon négligeable pour des études comportant peu de congestion. Enfin, l'utilisation de cette approche élimine la dichotomie ligne/liaison. Des tests, avec des études sans point de congestion supplémentaire, montrent que les résulats produits par la version actuelle du logiciel et ceux produits par le nouveau noyau sont identiques à quelques exceptions près, ce qui confirme que l'approximation faite par le noeud TRAFFICTRUNK est acceptable. Mais la nouvelle méthodologie possède l'avantage de simplifier la modélisation et de permettre la définition de points de congestion pour tous les noeuds de la base RCMB

1. Par exemple, les probabilités de blocages sont calculées par la formule originale récursive alors qu'il existe des méthodes plus rapides exploitant les propriétés de ce type de fonction.

Il est important de mentionner que la fonction Lambda prototype est indépendante du DRL choisi. On peut donc utiliser la méthodologie avec tout autre DRL qui est le mieux approprié pour la tâche. Avec la contrainte de *localité* imposée par l'algorithme d'approvisionnement, il est difficile de trouver d'autres approches qui pourraient fournir des écarts résiduels inférieurs avec un temps de calcul raisonnable. Il est permis de penser qu'une optimisation plus poussée de l'utilisation des ressources du réseau simulé ne peut se faire qu'en remplaçant l'algorithme d'approvisionnement actuel avec un algorithme de type parallèle (global).

Le deuxième noyau modifié, qui implante DRL/Lambda, doit être comparé à des DRL existants et éprouvés afin de vérifier si les résultats produits sont assez précis pour être utilisés dans le cadre de la planification. Le problème abordé par le BIP est fortement contraint (décision pour le dernier module seulement) et devra probablement être raffiné pour se rapprocher d'un vrai DRL. L'introduction d'un programme en nombre entier montre une voie possible dans l'utilisation de tels algorithmes dans le contexte du dimensionnement des réseaux de liaisons, recours qui sera de plus en plus fréquent à mesure la modularité des liens augmente[1] (comme avec les liens optiques à haute capacité). Si DRL/Lambda n'est pas retenu comme solution permanente, il aura tout de même servi à montrer le type de problème (l'écart résiduel) qui se présentera lors de l'implantation d'un éventuel DRL avec SNAP.

1. Problème mentionné dans le rapport de la référence 12.

Bibliographie

1. AT&T Bell Laboratories, *Enginering and Operations in the Bell System*, AT&T Bell Laboratories, 2e éd., Murray Hill, NJ (1984).
2. AT&T Bell Laboratories, *Telecommunication Transmission Engineering*, Volume 3: Networks and Services, Western Electric Company Inc., 2e éd., Winston-Salem, CA (1977).
3. BAZARAA, M.S., JARVIS, J.J., *Linear Programming and Networks Flows*, John Wiley and Sons Inc., Atlanta, GE (1977).
4. BEAR, D., *Principles of telecommunication-traffic engineering*, Peter Pereginus Ltd, Angleterre (1976).
5. BOUCHER, J.R., *Voice Teletraffic Systems Engineering*, Artech House, Norwood MA, (1988).
6. FLOOD, J.E., *Telecommunications networks*, Peter Pereginus Ltd, Stevenage, Angleterre (1975).
7. FREEMAN, R.C. *Telecommunications System Engineering, Analog and Digital Network Design*, John Wiley & Sons, États-Unis (1980).
8. KEISER, B.E., STANGE, E., *Digital Telephony and Networks Integration*, Von Nostrand Reinhold Company, New York NY (1985).
9. NEMHAUSER, G.L., WOLSEY, L.A., *Integer and Combinatorial Optimization*, John Wiley and Sons, États-Unis (1988).
10. WILKINSON, R.I., "Theories for Toll Traffic Engineering in the U.S.A.", The Bell Technical Journal, Vol. 35, pp. 421-514, États-Unis (1956).

Rapports internes confidentiels (RBN)

11. LE NIR, A.S., HUBERMAN, R.M., DRWIEGA, T., *Dimensioning for a FHR/HPR Toll Network*, Recherches Bell-Northern Ltée, Montréal (1984).

12. MICHAILOV, S., HIGHAM, P., *Consolidation of Existing and New Methodology for a Mixed FHR/HPR Dimensioner*, Recherches Bell-Nortern Ltée, Montréal (1987).

Autres documents internes (RBN)

13. BNR Telco Network Planning Tools, SNAP 1.2: Reference Manual, RBN, Montréal (1988).

14. BNR Telco Network Planning Tools, SNAP 1.2: User Guide, RBN, Montréal (1988).

15. BNR Telco Network Planning Tools, Documentation sur le noyau et la base RCMB, Montréal (198-).

16. FOISY, C., "Rapport étude X" et "Rapport étude Y", Montréal (octobre 1989).

Autre documentation

17. FOISY, C., "Recherche d'un algorithme pour le calcul du point de saturation d'un commutateur dans un réseau de télécommunications", résumé de la présentation dans le cadre du cours Séminaires 3.699, Montréal (novembre 1989).

Appendices

A. Preuve de la convergence de l'algorithme d'approvisionnement de SNAP

Définitions

S	ensemble des index de tous les commutateurs
ES_i	ensembles des commutateurs congestionnés à l'itération i
$Strat$	ensemble des stratégies pour les commutateurs (ensemble fini)
$StratAc_i$	ensemble des stratégies actives à l'itération i

Lorsqu'un commutateur congestionne à l'itération k de la boucle Compressor, SNAP le "ferme" et il ne peut plus être utilisé pour absorber la croissance de la charge dans le réseau. Les stratégies ultérieures à cette fermeture sont tout simplement ignorées et un message d'avertissement est laissé au planificateur dans le rapport d'activité de SNAP. Le schéma bloc haut niveau de l'algorithme est donné à la Figure 9, page 24.

Nous voyons que la convergence de la boucle Assigner est assurée par le nombre fini de stratégies. Par contre la convergence de Compressor est assurée du fait que les commutateurs sont isolés lors de l'exécution de C$BSE (le sous-programme qui effectue la recherche binaire). Cette condition est primordiale. En poussant l'abstraction du RCMB et sa relation avec la charge du réseau d'accès et les point de congestion, chaque commutateur S_i (c'est-à-dire le commutateur principal et tous les commutateurs distants qui lui sont rattachées) se comporte comme une fonction $f_i \geq 0$ (monotone croissante) telle que

$$f_i(C_i) = \begin{cases} \leq 1 & \text{si } S_i \text{ n'est pas congestionné sous la charge } C_i \\ > 1 & \text{si } S_i \text{ est congestionné sous la charge} C_i \end{cases}$$

C'est le module Installer qui "calcule" les f_i et le module Compressor commence la recherche binaire si $\max_{i \in S} f_i > 1$ et se termine lorsque $\max_{i \in S} f_i \leq 1$. Pour chaque commutateur S_i congestionné, le module Compressor génère une suite de charge $\{C_{i1}, ..., C_{im}\}$ tel que $C_{ij} < C_i$ pour tout j (selon l'algorithme de recherche binaire) avec comme critère d'arrêt

$$\frac{|C_{i(m-1)} - C_{im}|}{C_{im}} \leq \varepsilon \quad \text{et} \quad f_i(C_{i(m-1)}) > 1 \,, f_i(C_{im}) \leq 1$$

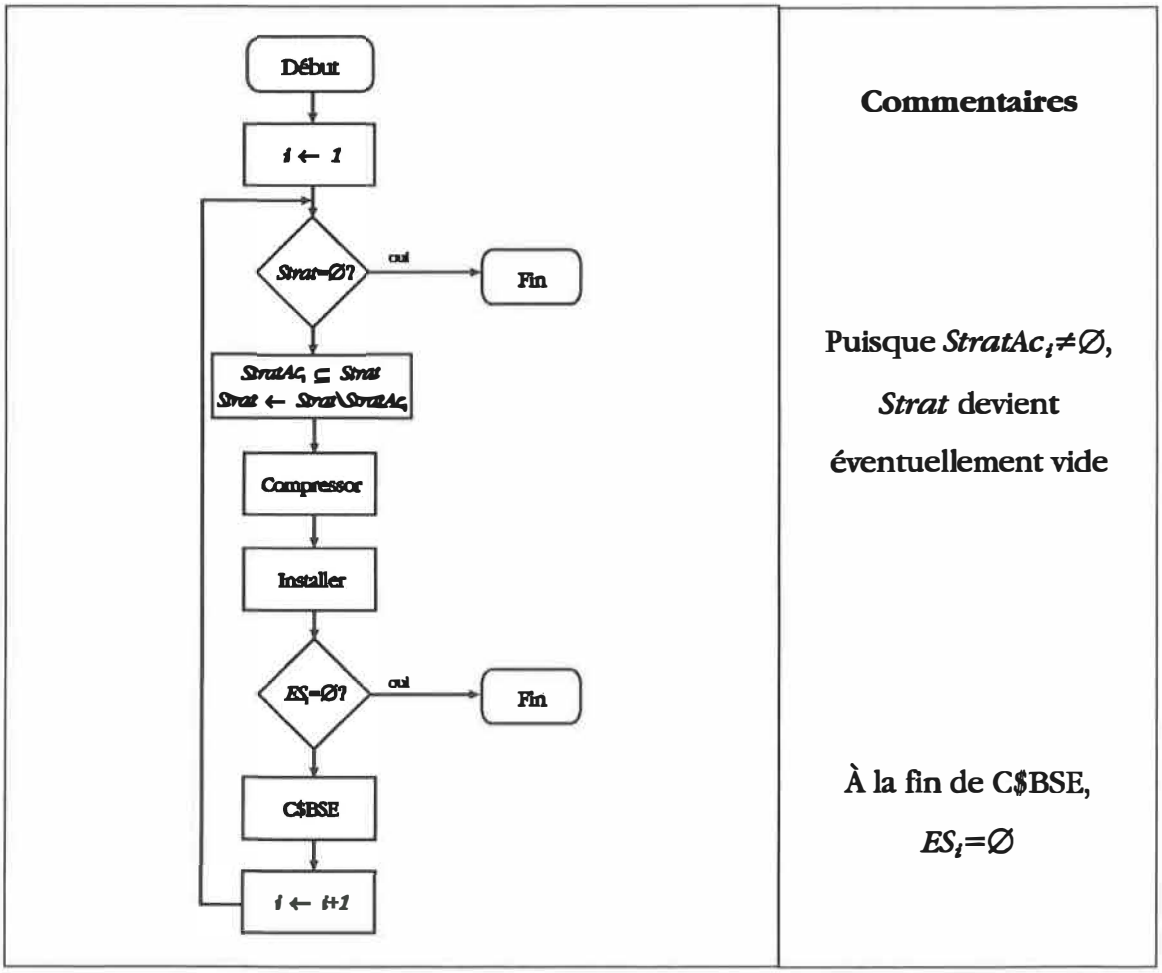


Figure 34. **Schéma bloc de l'algorithme d'approvisionnement**

où ε est la tolérance démandée par le planificateur. Puisque le module Compressor ne réassigne pas la charge et que les commutateurs sont "isolés" par le noeud TRAFFICTRUNK, le module Compressor se trouve à résoudre le problème

$$\text{maximiser} \sum_{i \in S} C_{im_i}$$

tel que

$$\max_{i \in S} f_i \leq 1$$

Chaque fonction étant indépendante, ce problème n'est qu'en fait l'optimisation séquentielle de $|S|$ fonctions monotones croissantes[1]. La convergence de la boucle Compressor dépend donc de la monoticité des f_i (monotone croissante) et de l'indépendance entre elles.

La seule différence entre l'ancienne fonction et la fonction Lambda est que ES_i (l'ensemble des commutateurs congestionnés à l'itération i) et ES_j respectent la relation

$$ES_i \cap ES_j = \varnothing \quad \text{pour tout } i < j$$

c'est-à-dire qu'un commutateur qui a saturé à l'itération i ne peut "resaturer" la même année, avec l'ancienne fonction (TRAFFICTRUNK), alors que l'intersection peut être non-vide pour la fonction Lambda (puisque la fonction est rafraîchie juste avant l'appel de Compressor (voir p. 53)).

1. L'analogie des fonctions f est justifiée puisque toutes les tables de capacité de la base RCMB sont obligatoirement monotones croissantes. Ceci implique que la fonction résultante à chacun des noeuds est une composition de fonction monotone croissante, qui résulte en une fonction monotone croissante.

B. Résultats de l'étude Y avec la fonction Lambda prototype

Les pages qui suivent contiennent l'ensemble des résultats recueillis avec les tests de l'étude Y et le noyau modifié avec la fonction Lambda prototype. Afin de respecter la confidentialité des documents, les noms des commutateurs ont été changés pour le type de commutateur et une lettre.

Voici les définitions des ressources que l'on retrouve dans ces pages. BUSNS et POTS sont tous deux des types de demande qui représentent, respectivement, les abonnés commerciaux et résidentiels (voir Figures 24 et 27). CHANNELS représente le nombre de liaisons installées pour le commutateur (voir Figure 28). DS30 LEFT et DS30 RIGHT représente le nombre de canaux de type DS30 des noeuds correspondants aux Figures 29 et 30. Notons que le nombre de DS30 LEFT est toujours 0 lorsque la base RCMB COM est utilisée puisque ce noeud n'existe plus dans cette base (voir Figure 28). Enfin, la ressource TH_MON nous donne le trafic (en CCS) qui entre dans le point de congestion THING (Figure 28, 29, et 30).

La base RCMB COM est la même que celle utilisée avec l'étude X. La base RCMB MOC est une base dérivée de RCMB COU. La base RCMB MOC ne contient que des points de congestion supplémentaires, elle possède comme RCMB COU deux côtés.

Les trois lettres (MOC ou COM) au dessus des colonnes indiquent la base RCMB qui a généré les résultats. La tolérance est aussi présente au dessus de la colonne correspondante. Les chiffre en gras indiquent qu'il y a eu congestion pour le commutateur correspondant à l'année indiquée. L'avant-dernière rangée CD Cong. donne le nombre de commutateurs distants qui ont congestionné pendant l'année. La rangée NRB indique le nombre de recherches binaires requis pendant l'année.

Point de congestion: aucun **Test No. E1-7**

Temps de calcul:
COM: 1008s
MOC: 986s

Année= 0	MOC	COM
DMS A	0%	0%
BUSNS	5465	5465
POTS	9791	9791
CHANNELS	1320	1242
DS30 LEFT	44	0
DS30 RIGHT	48	48
TH_MON	97529	97264
DMS B		
BUSNS	10438	10438
POTS	17253	17253
CHANNELS	1883	1623
DS30 LEFT	63	0
DS30 RIGHT	64	64
TH_MON	176661	174396
DMS C		
BUSNS	5157	5157
CHANNELS	406	501
DS30 LEFT	14	0
DS30 RIGHT	32	32
TH_MON	36099	36330
XBAR D		
BUSNS	2532	2532
POTS	2937	2937
CHANNELS	539	539
TH_MON	9528	9662
DMS E		
BUSNS	17399	17399
POTS	3409	3409
CHANNELS	1559	1386
DS30 LEFT	52	0
DS30 RIGHT	64	64
TH_MON	139929	137973
DMS F		
BUSNS	17398	17398
POTS	3408	3408
CHANNELS	1559	1386
DS30 LEFT	52	0
DS30 RIGHT	64	64
TH_MON	139916	137962
CD Cong.:	1	1
NRB:	1	1

Année= 1		
DMS A		
BUSNS	5795	5795
POTS	10809	10809
CHANNELS	1462	1355
DS30 LEFT	49	0
DS30 RIGHT	48	48
TH_MON	109076	108792
DMS B		
BUSNS	11188	11188
POTS	17770	17770
CHANNELS	1995	1704
DS30 LEFT	67	0
DS30 RIGHT	64	64
TH_MON	187223	184831
DMS C		
BUSNS	6007	6007
POTS	1010	1010
CHANNELS	531	611
DS30 LEFT	18	0
DS30 RIGHT	32	32
TH_MON	47422	47723
XBAR D		
BUSNS	2587	2587
POTS	3069	3069
CHANNELS	539	539
TH_MON	9839	9977
DMS E		
BUSNS	18399	18399
POTS	3409	3409
CHANNELS	1637	1446
DS30 LEFT	55	0
DS30 RIGHT	64	64
TH_MON	146929	144882
DMS F		
BUSNS	19188	19188
POTS	3934	3934
CHANNELS	1734	1509
DS30 LEFT	58	0
DS30 RIGHT	64	64
TH_MON	155245	153052
CD Cong.:	2	2
NRB:	1	1

Année= 2		
DMS A		
BUSNS	6125	6125
POTS	11826	11826
CHANNELS	1803	1591
DS30 LEFT	61	0
DS30 RIGHT	64	64
TH_MON	132345	131443
DMS B		
BUSNS	11838	11838
POTS	18286	18286
CHANNELS	2097	1774
DS30 LEFT	70	0
DS30 RIGHT	64	64
TH_MON	196582	194125
DMS C		
BUSNS	6832	6832
POTS	2018	2018
CHANNELS	654	713
DS30 LEFT	22	0
DS30 RIGHT	32	32
TH_MON	58559	58972
XBAR D		
BUSNS	2642	2642
POTS	3201	3201
CHANNELS	534	534
TH_MON	10149	10291
DMS E		
BUSNS	19399	19399
POTS	3409	3409
CHANNELS	1715	1492
DS30 LEFT	58	0
DS30 RIGHT	64	64
TH_MON	153929	151821
DMS F		
BUSNS	20938	20938
POTS	4511	4511
CHANNELS	1902	1626
DS30 LEFT	64	0
DS30 RIGHT	64	64
TH_MON	170563	168158
CD Cong.:	1	1
NRB:	1	1

Point de congestion: aucun **Test No. E1-7 (suite)**

Année= 3	MOC	COM
DMS A	0%	0%
BUSNS	6455	6455
POTS	12844	12844
CHANNELS	1953	1709
DS30 LEFT	66	0
DS30 RIGHT	64	64
TH_MON	142699	141701
DMS B		
BUSNS	12388	12388
POTS	18802	18802
CHANNELS	2195	1847
DS30 LEFT	74	0
DS30 RIGHT	64	64
TH_MON	205130	202573
DMS C		
BUSNS	7657	7657
POTS	3027	3027
CHANNELS	777	823
DS30 LEFT	26	0
DS30 RIGHT	32	32
TH_MON	69702	70181
XBAR D		
BUSNS	2697	2697
POTS	3332	3332
CHANNELS	534	534
TH_MON	10458	10606
DMS E		
BUSNS	19649	19649
POTS	3409	3409
CHANNELS	1738	1509
DS30 LEFT	58	0
DS30 RIGHT	64	64
TH_MON	155679	153585
DMS F		
BUSNS	21938	21938
POTS	5083	5083
CHANNELS	2015	1711
DS30 LEFT	68	0
DS30 RIGHT	64	64
TH_MON	180607	178072
CD Cong.:	0	0
NRB:	0	0

Année= 4		
DMS A		
BUSNS	6785	6785
POTS	13862	13862
CHANNELS	2102	1812
DS30 LEFT	71	0
DS30 RIGHT	64	64
TH_MON	153054	152021
DMS B		
BUSNS	12938	12938
POTS	19318	19318
CHANNELS	2288	1906
DS30 LEFT	77	0
DS30 RIGHT	80	80
TH_MON	213676	211059
DMS C		
BUSNS	8482	8482
POTS	30094	30094
CHANNELS	2447	2134
DS30 LEFT	82	0
DS30 RIGHT	80	80
TH_MON	219474	220392
XBAR D		
BUSNS	2752	2752
POTS	3464	3464
CHANNELS	529	529
TH_MON	10769	10934
DMS E		
BUSNS	19899	19899
POTS	3409	3409
CHANNELS	1754	1516
DS30 LEFT	59	0
DS30 RIGHT	64	64
TH_MON	157429	155387
DMS F		
BUSNS	22938	22938
POTS	5656	5656
CHANNELS	2128	1781
DS30 LEFT	71	0
DS30 RIGHT	64	64
TH_MON	190655	188041
CD Cong.:	0	0
NRB:	0	0

Année= 5		
DMS A		
BUSNS	6929	6929
POTS	15112	15112
CHANNELS	2250	1928
DS30 LEFT	75	0
DS30 RIGHT	80	80
TH_MON	163043	161931
DMS B		
BUSNS	13126	13126
POTS	20225	20225
CHANNELS	2365	1969
DS30 LEFT	79	0
DS30 RIGHT	80	80
TH_MON	220519	217839
DMS C		
BUSNS	8874	8874
POTS	31610	31610
CHANNELS	2567	2231
DS30 LEFT	86	0
DS30 RIGHT	80	80
TH_MON	230282	231237
XBAR D		
POTS	3464	3464
CHANNELS	392	392
TH_MON	5265	5352
DMS E		
BUSNS	20149	20149
POTS	3409	3409
CHANNELS	1777	1537
DS30 LEFT	60	0
DS30 RIGHT	64	64
TH_MON	159179	157149
DMS F		
BUSNS	23506	23506
POTS	6785	6785
CHANNELS	2240	1870
DS30 LEFT	75	0
DS30 RIGHT	64	64
TH_MON	200638	197899
CD Cong.:	1	1
NRB:	1	1

Point de congestion: 1900 CHANNELS (DMS) Test No. E1-8

Temps de calcul:
MOC: 1404s COM: 1917, 1226 et 1195s (0, 1 et 2%)

Année= 0	MOC	COM	COM	COM
DMS A	**0%**	**0%**	**1%**	**2%**
BUSNS	5465	5465	5465	5465
POTS	9791	9791	9791	9791
CHANNELS	1320	1242	1242	1242
DS30 LEFT	44	0	0	0
DS30 RIGHT	48	48	48	48
TH_MON	97529	97264	97261	97264
Δ%	30.5	34.6	34.6	34.6
DMS B				
BUSNS	10438	10438	10438	10438
POTS	17253	17253	17253	17253
CHANNELS	1883	1623	1623	1623
DS30 LEFT	63	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	176661	174396	174396	174396
Δ%	0.9	14.6	14.6	14.6
DMS C				
BUSNS	5157	5157	5157	5157
CHANNELS	406	501	501	501
DS30 LEFT	14	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	36099	36330	36330	36330
Δ%	78.6	73.6	73.6	73.6
XBAR D				
BUSNS	2532	2532	2532	2532
POTS	2937	2937	2937	2937
CHANNELS	539	539	539	539
TH_MON	9528	9662	9662	9662
DMS E				
BUSNS	17399	17399	17399	17399
POTS	3409	3409	3409	3409
CHANNELS	1559	1386	1386	1386
DS30 LEFT	52	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139929	137973	137973	137973
Δ%	17.9	27.1	27.1	27.1
DMS F				
BUSNS	17398	17398	17398	17398
POTS	3408	3408	3408	3408
CHANNELS	1559	1386	1386	1386
DS30 LEFT	52	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139916	137962	137962	137962
Δ%	17.9	27.1	27.1	27.1
CD Cong.:	1	1	1	1
NRB:	1	1	1	1

Année= 1				
DMS A				
BUSNS	5795	5795	5795	5795
POTS	10809	10809	10809	10809
CHANNELS	1462	1355	1355	1355
DS30 LEFT	49	0	0	0
DS30 RIGHT	48	48	48	48
TH_MON	109076	108792	108769	108754
Δ%	23.1	28.7	28.7	28.7
DMS B				
BUSNS	10438	11188	11188	11188
POTS	17542	17770	17770	17770
CHANNELS	1900	1704	1704	1704
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	178931	184831	184789	184724
Δ%	0.0	10.3	10.3	10.3
DMS C				
BUSNS	6007	6007	6007	6007
POTS	1010	1010	1010	1010
CHANNELS	531	611	611	611
DS30 LEFT	18	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	47422	47723	47723	47723
Δ%	72.1	67.8	67.8	67.8
XBAR D				
BUSNS	2587	2587	2587	2587
POTS	3069	3069	3069	3069
CHANNELS	539	539	539	539
TH_MON	9839	9977	9977	9977
DMS E				
BUSNS	18399	18399	18399	18399
POTS	3409	3409	3409	3409
CHANNELS	1637	1446	1446	1446
DS30 LEFT	55	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	146929	144882	144882	144882
Δ%	13.8	23.9	23.9	23.9
DMS F				
BUSNS	19188	19188	19188	19188
POTS	3934	3934	3934	3934
CHANNELS	1734	1509	1509	1509
DS30 LEFT	58	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	155245	153052	153052	153052
Δ%	8.7	20.6	20.6	20.6
CD Cong.:	3	2	2	2
NRB:	2	1	1	1

Point de congestion: 1900 CHANNELS (DMS) Test No. E1-8 (suite)

Année= 2	MOC	COM	COM	COM
DMS A	**0%**	**0%**	**1%**	**2%**
BUSNS	6125	6125	6125	6125
POTS	11826	11826	11826	11826
CHANNELS	1803	1591	1591	1591
DS30 LEFT	61	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	132345	131443	131406	131391
Δ%	5.1	16.3	16.3	16.3
DMS B				
BUSNS	10438	11838	11838	11838
POTS	17542	18286	18286	18286
CHANNELS	1900	1774	1774	1774
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	178931	194125	194085	194021
Δ%	0.0	6.6	6.6	6.6
DMS C				
BUSNS	6832	6832	6832	6832
POTS	2018	2018	2018	2018
CHANNELS	654	713	713	713
DS30 LEFT	22	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	58559	58972	58972	58972
Δ%	65.6	62.5	62.5	62.5
XBAR D				
BUSNS	2642	2642	2642	2642
POTS	3201	3201	3201	3201
CHANNELS	534	534	534	534
TH_MON	10149	10291	10291	10291
DMS E				
BUSNS	19425	19399	19399	19399
POTS	3409	3409	3409	3409
CHANNELS	1721	1492	1492	1492
DS30 LEFT	58	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	154111	151821	151821	151821
Δ%	9.4	21.5	21.5	21.5
DMS F				
BUSNS	20912	20938	20938	20938
POTS	4511	4511	4511	4511
CHANNELS	1900	1626	1626	1626
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	170381	168158	168158	168158
Δ%	**0.0**	14.4	14.4	14.4
CD Cong.:	1	1	1	1
NRB:	1	1	1	1

Année= 3				
DMS A				
BUSNS	6125	6455	6455	6455
POTS	12639	12844	12844	12844
CHANNELS	1900	1709	1709	1709
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139158	141701	141663	141651
Δ%	**0.0**	10.1	10.1	10.1
DMS B				
BUSNS	10438	12388	12388	12388
POTS	17542	18802	18802	18802
CHANNELS	1900	1847	1847	1847
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	178931	202573	202533	202468
Δ%	0.0	2.8	2.8	2.8
DMS C				
BUSNS	7657	7657	7657	7657
POTS	3027	3027	3027	3027
CHANNELS	777	823	823	823
DS30 LEFT	26	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	69702	70181	70181	70181
Δ%	59.1	56.7	56.7	56.7
XBAR D				
BUSNS	2697	2697	2697	2697
POTS	3332	3332	3332	3332
CHANNELS	534	534	534	534
TH_MON	10458	10606	10607	10606
DMS E				
BUSNS	20675	19649	19649	19649
POTS	3981	3409	3409	3409
CHANNELS	1852	1509	1509	1509
DS30 LEFT	62	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	165903	153585	153585	153585
Δ%	2.5	20.6	20.6	20.6
DMS F				
BUSNS	20912	21938	21938	21938
POTS	4511	5083	5083	5083
CHANNELS	1900	1711	1711	1711
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	170381	178072	178072	178072
Δ%	0.0	9.9	9.9	9.9
CD Cong.:	1	0	0	0
NRB:	1	0	0	0

Point de congestion: 1900 CHANNELS (DMS) Test No. E1-8 (suite)

Année= 4	MOC	COM	COM	COM
DMS A	**0%**	**0%**	**1%**	**2%**
BUSNS	5605	6785	6785	6785
POTS	12639	13862	13862	13862
CHANNELS	**1900**	1824	1824	1824
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139160	151995	151956	151941
Δ%	**0.0**	4.0	4.0	4.0
DMS B				
BUSNS	10438	12595	12483	12453
POTS	17542	19318	19318	19318
CHANNELS	1900	**1900**	**1890**	**1888**
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	80	80	80
TH_MON	178931	208681	207871	207598
Δ%	0.0	**0.0**	**0.5**	**0.6**
DMS C				
BUSNS	7657	7657	7657	7657
POTS	21952	25829	25898	25865
CHANNELS	**1900**	**1896**	**1900**	**1897**
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	80	80	80
TH_MON	170383	191905	192272	192092
Δ%	**0.0**	**0.2**	**0.0**	**0.2**
XBAR D				
BUSNS	2752	2752	2752	2752
POTS	3464	3464	3464	3464
CHANNELS	530	531	530	530
TH_MON	10769	10928	10928	10928
DMS E				
BUSNS	20879	19899	19899	19899
POTS	4554	3409	3409	3409
CHANNELS	**1900**	1527	1526	1526
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	170380	155370	155369	155369
Δ%	**0.0**	19.6	19.7	19.7
DMS F				
BUSNS	20912	22938	22938	22938
POTS	4511	5656	5656	5656
CHANNELS	1900	1793	1794	1793
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	170381	188017	188016	188016
Δ%	0.0	5.6	5.6	5.6
CD Cong.:	0	0	0	0
NRB:	1	3	2	2

Année= 5				
DMS A				
BUSNS	5165	6785	6785	6785
POTS	12639	14369	14322	14322
CHANNELS	**1900**	**1895**	**1892**	**1892**
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139155	157519	157254	157239
Δ%	**0.0**	**0.3**	**0.4**	**0.4**
DMS B				
BUSNS	10438	12204	12181	12178
POTS	17542	19318	19318	19318
CHANNELS	1900	**1896**	**1893**	**1893**
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	80	80	80
TH_MON	178931	206717	206459	206317
Δ%	0.0	**0.2**	**0.4**	**0.4**
DMS C				
BUSNS	7657	7412	7394	7133
POTS	21952	25829	25898	25865
CHANNELS	1900	**1899**	**1900**	**1882**
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	80	80	80
TH_MON	170383	190200	190439	188434
Δ%	0.0	**0.1**	**0.0**	**0.9**
XBAR D				
POTS	3464	3464	3464	3464
CHANNELS	393	393	392	392
TH_MON	5265	5348	5347	5347
DMS E				
BUSNS	20879	20149	20149	20149
POTS	4554	3409	3409	3409
CHANNELS	1900	1555	1555	1555
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	170380	157115	157112	157113
Δ%	0.0	18.2	18.2	18.2
DMS F				
BUSNS	20912	23506	23506	23506
POTS	4511	6785	6785	6785
CHANNELS	1900	1890	1890	1890
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	80	80	80
TH_MON	170381	197855	197853	197853
Δ%	0.0	0.5	0.5	0.5
CD Cong.:	0	1	1	1
NRB:	1	4	2	3

Point de congestion: 9600 TH_MON (XBAR) **Test No. E1-9**

Temps de calcul:
MOC: 1046s COM: 1111, 1014, 950s (0, 1 et 2%)

Année= 0	MOC	COM	COM	COM
DMS A	**0%**	**0%**	**1%**	**2%**
BUSNS	5465	5465	5465	5465
POTS	9791	9791	9791	9791
CHANNELS	1320	1242	1242	1242
DS30 LEFT	44	0	0	0
DS30 RIGHT	48	48	48	48
TH_MON	97529	97264	97261	97264
DMS B				
BUSNS	10438	10438	10438	10438
POTS	17253	17253	17253	17253
CHANNELS	1883	1623	1623	1623
DS30 LEFT	63	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	176661	174396	174396	174396
DMS C				
BUSNS	5157	5157	5157	5157
CHANNELS	406	501	501	501
DS30 LEFT	14	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	36099	36330	36330	36330
XBAR D				
BUSNS	2532	2532	2532	2532
POTS	2937	2937	2937	2937
CHANNELS	539	539	539	539
TH_MON	9528	9662	9662	9662
Δ%	0.8	-0.6	-0.6	-0.6
DMS E				
BUSNS	17399	17399	17399	17399
POTS	3409	3409	3409	3409
CHANNELS	1559	1386	1386	1386
DS30 LEFT	52	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139929	137973	137973	137973
DMS F				
BUSNS	17398	17398	17398	17398
POTS	3408	3408	3408	3408
CHANNELS	1559	1386	1386	1386
DS30 LEFT	52	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139916	137962	137962	137962
CD Cong.:	1	1	1	1
NRB:	1	1	1	1

Année= 1				
DMS A				
BUSNS	5795	5795	5795	5795
POTS	10809	10809	10809	10809
CHANNELS	1462	1354	1354	1354
DS30 LEFT	49	0	0	0
DS30 RIGHT	48	48	48	48
TH_MON	109076	108792	108769	108752
DMS B				
BUSNS	11188	11188	11188	11188
POTS	17770	17770	17770	17770
CHANNELS	1995	1704	1704	1704
DS30 LEFT	67	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	187223	184835	184793	184728
DMS C				
BUSNS	6007	6007	6007	6007
POTS	1010	1010	1010	1010
CHANNELS	531	611	611	611
DS30 LEFT	18	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	47422	47723	47723	47724
XBAR D				
BUSNS	2532	2451	2451	2446
POTS	2984	2937	2937	2937
CHANNELS	539	539	539	539
TH_MON	9600	9500	9500	9489
Δ%	0.0	1.0	1.0	1.2
DMS E				
BUSNS	18399	18399	18399	18399
POTS	3409	3409	3409	3409
CHANNELS	1637	1446	1446	1446
DS30 LEFT	55	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	146929	144887	144887	144887
DMS F				
BUSNS	19188	19188	19188	19188
POTS	3934	3934	3934	3934
CHANNELS	1734	1508	1508	1508
DS30 LEFT	58	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	155245	153055	153055	153055
CD Cong.:	2	2	2	2
NRB:	2	2	2	2

Point de congestion: 9600 TH_MON (XBAR) **Test No. E1-9 (suite)**

Année= 2	MOC	COM	COM	COM
DMS A	**0%**	**0%**	**1%**	**2%**
BUSNS	6125	6125	6125	6125
POTS	11826	11826	11826	11826
CHANNELS	1803	1592	1592	1592
DS30 LEFT	61	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	132345	131438	131400	131385
DMS B				
BUSNS	11838	11838	11838	11838
POTS	18286	18286	18286	18286
CHANNELS	2097	1775	1775	1775
DS30 LEFT	70	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	196582	194130	194090	194024
DMS C				
BUSNS	6832	6832	6832	6832
POTS	2018	2018	2018	2018
CHANNELS	654	715	715	715
DS30 LEFT	22	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	58559	58970	58970	58970
XBAR D				
BUSNS	2532	2451	2451	2446
POTS	2984	2937	2937	2937
CHANNELS	534	534	534	534
TH_MON	9600	9502	9502	9491
Δ%	0.0	1.0	1.0	1.1
DMS E				
BUSNS	19399	19399	19399	19399
POTS	3409	3409	3409	3409
CHANNELS	1715	1492	1492	1492
DS30 LEFT	58	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	153929	151828	151826	151828
DMS F				
BUSNS	20938	20938	20938	20938
POTS	4511	4511	4511	4511
CHANNELS	1902	1624	1624	1624
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	170563	168165	168165	168165
CD Cong.:	1	1	1	1
NRB:	1	1	1	1

Année= 3				
DMS A				
BUSNS	6455	6455	6455	6455
POTS	12844	12844	12844	12844
CHANNELS	1953	1705	1705	1705
DS30 LEFT	66	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	142699	141694	141656	141642
DMS B				
BUSNS	12388	12388	12388	12388
POTS	18802	18802	18802	18802
CHANNELS	2195	1846	1846	1846
DS30 LEFT	74	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	205130	202579	202538	202473
DMS C				
BUSNS	7657	7657	7657	7657
POTS	3027	3027	3027	3027
CHANNELS	777	825	825	825
DS30 LEFT	26	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	69702	70181	70181	70181
XBAR D				
BUSNS	2532	2451	2451	2446
POTS	2984	2937	2937	2937
CHANNELS	534	534	534	534
TH_MON	9600	9505	9505	9495
Δ%	0.0	1.0	1.0	1.1
DMS E				
BUSNS	19649	19649	19649	19649
POTS	3409	3409	3409	3409
CHANNELS	1738	1510	1510	1510
DS30 LEFT	58	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	155679	153594	153593	153594
DMS F				
BUSNS	21938	21938	21938	21938
POTS	5083	5083	5083	5083
CHANNELS	2015	1712	1712	1712
DS30 LEFT	68	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	180607	178082	178084	178082
CD Cong.:	0	0	0	0
NRB:	0	0	0	0

Point de congestion: 9600 TH_MON (XBAR) Test No. E1-9 (suite)

Année= 4	MOC	COM	COM	COM
DMS A	0%	0%	1%	2%
BUSNS	6785	6785	6785	6785
POTS	13862	13862	13862	13862
CHANNELS	2102	1811	1811	1811
DS30 LEFT	71	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	153054	152013	151975	151960
DMS B				
BUSNS	12938	12938	12938	12938
POTS	19318	19318	19318	19318
CHANNELS	2288	1904	1904	1904
DS30 LEFT	77	0	0	0
DS30 RIGHT	80	80	80	80
TH_MON	213676	211067	211028	210962
DMS C				
BUSNS	8482	8482	8482	8482
POTS	30094	30094	30094	30094
CHANNELS	2447	2135	2135	2135
DS30 LEFT	82	0	0	0
DS30 RIGHT	80	80	80	80
TH_MON	219474	220390	220392	220390
XBAR D				
BUSNS	2532	2451	2451	2446
POTS	2984	2937	2937	2937
CHANNELS	529	529	529	529
TH_MON	9600	9518	9518	9508
Δ%	0.0	0.9	0.9	1.0
DMS E				
BUSNS	19899	19899	19899	19899
POTS	3409	3409	3409	3409
CHANNELS	1754	1517	1517	1517
DS30 LEFT	59	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	157429	155399	155399	155401
DMS F				
BUSNS	22938	22938	22938	22938
POTS	5656	5656	5656	5656
CHANNELS	2128	1781	1781	1781
DS30 LEFT	71	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	190655	188055	188055	188055
CD Cong.:	0	0	0	0
NRB:	0	0	0	0

Année= 5				
DMS A				
BUSNS	6929	6929	6929	6929
POTS	15112	15112	15112	15112
CHANNELS	2250	1926	1926	1926
DS30 LEFT	75	0	0	0
DS30 RIGHT	80	80	80	80
TH_MON	163043	161922	161883	161868
DMS B				
BUSNS	13126	13126	13126	13126
POTS	20225	20225	20225	20225
CHANNELS	2365	1969	1967	1966
DS30 LEFT	79	0	0	0
DS30 RIGHT	80	80	80	80
TH_MON	220519	217848	217749	217627
DMS C				
BUSNS	8874	8874	8874	8874
POTS	31610	31610	31610	31610
CHANNELS	2567	2229	2230	2230
DS30 LEFT	86	0	0	0
DS30 RIGHT	80	80	80	80
TH_MON	230282	231237	231238	231236
XBAR D				
POTS	2984	2937	2937	2937
CHANNELS	342	341	341	341
TH_MON	4536	4541	4541	4541
Δ%	52.8	52.7	52.7	52.7
DMS E				
BUSNS	20149	20149	20149	20149
POTS	3409	3409	3409	3409
CHANNELS	1777	1536	1536	1536
DS30 LEFT	60	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	159179	157160	157161	157160
DMS F				
BUSNS	23506	23506	23506	23506
POTS	6785	6785	6785	6785
CHANNELS	2240	1868	1868	1868
DS30 LEFT	75	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	200638	197913	197913	197913
CD Cong.:	1	1	1	1
NRB:	1	1	1	1

Points de congestion: PC de E1-8 et E1-9 **Test No. E1-10**

Temps de calcul:
MOC: 1447s COM: 1989, 1305, 1244s (0, 1 et 2%)

Année= 0	MOC	COM	COM	COM
DMS A	**0%**	**0%**	**1%**	**2%**
BUSNS	5465	5465	5465	5465
POTS	9791	9791	9791	9791
CHANNELS	1320	1242	1242	1242
DS30 LEFT	44	0	0	0
DS30 RIGHT	48	48	48	48
TH_MON	97529	97264	97261	97264
Δ%	30.5	34.6	34.6	34.6
DMS B				
BUSNS	10438	10438	10438	10438
POTS	17253	17253	17253	17253
CHANNELS	1883	1623	1623	1623
DS30 LEFT	63	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	176661	174396	174396	174396
Δ%	0.9	14.6	14.6	14.6
DMS C				
BUSNS	5157	5157	5157	5157
CHANNELS	406	501	501	501
DS30 LEFT	14	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	36099	36330	36330	36330
Δ%	78.6	73.6	73.6	73.6
XBAR D				
BUSNS	2532	2532	2532	2532
POTS	2937	2937	2937	2937
CHANNELS	539	539	539	539
TH_MON	9528	9662	9662	9662
Δ%	0.8	-0.6	-0.6	-0.6
DMS E				
BUSNS	17399	17399	17399	17399
POTS	3409	3409	3409	3409
CHANNELS	1559	1386	1386	1386
DS30 LEFT	52	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139929	137973	137973	137973
Δ%	17.9	27.1	27.1	27.1
DMS F				
BUSNS	17398	17398	17398	17398
POTS	3408	3408	3408	3408
CHANNELS	1559	1386	1386	1386
DS30 LEFT	52	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139916	137962	137962	137962
Δ%	17.9	27.1	27.1	27.1
CD Cong.:	1	1	1	1
NRB:	1	1	1	1

Année= 1				
DMS A				
BUSNS	5795	5795	5795	5795
POTS	10809	10809	10809	10809
CHANNELS	1462	1354	1354	1354
DS30 LEFT	49	0	0	0
DS30 RIGHT	48	48	48	48
TH_MON	109076	108792	108769	108752
Δ%	23.1	28.7	28.7	28.7
DMS B				
BUSNS	10438	11188	11188	11188
POTS	17542	17770	17770	17770
CHANNELS	1900	1704	1704	1704
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	178931	184835	184793	184728
Δ%	0.0	10.3	10.3	10.3
DMS C				
BUSNS	6007	6007	6007	6007
POTS	1010	1010	1010	1010
CHANNELS	531	611	611	611
DS30 LEFT	18	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	47422	47723	47723	47724
Δ%	72.1	67.8	67.8	67.8
XBAR D				
BUSNS	2532	2451	2451	2446
POTS	2984	2937	2937	2937
CHANNELS	539	539	539	539
TH_MON	9600	9500	9500	9489
Δ%	0.0	1.0	1.0	1.2
DMS E				
BUSNS	18399	18399	18399	18399
POTS	3409	3409	3409	3409
CHANNELS	1637	1446	1446	1446
DS30 LEFT	55	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	146929	144887	144887	144887
Δ%	13.8	23.9	23.9	23.9
DMS F				
BUSNS	19188	19188	19188	19188
POTS	3934	3934	3934	3934
CHANNELS	1734	1508	1508	1508
DS30 LEFT	58	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	155245	153055	153055	153055
Δ%	8.7	20.6	20.6	20.6
CD Cong.:	3	2	2	2
NRB:	3	2	2	2

Points de congestion: PC de E1-8 et E1-9 **Test No. E1-10 (suite)**

Année= 2	MOC	COM	COM	COM
DMS A	**0%**	**0%**	**1%**	**2%**
BUSNS	6125	6125	6125	6125
POTS	11826	11826	11826	11826
CHANNELS	1803	1592	1592	1592
DS30 LEFT	61	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	132345	131438	131400	131385
Δ%	5.1	16.2	16.2	16.2
DMS B				
BUSNS	10438	11838	11838	11838
POTS	17542	18286	18286	18286
CHANNELS	1900	1775	1775	1775
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	178931	194130	194090	194024
Δ%	0.0	6.6	6.6	6.6
DMS C				
BUSNS	6832	6832	6832	6832
POTS	2018	2018	2018	2018
CHANNELS	654	715	715	715
DS30 LEFT	22	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	58559	58970	58970	58970
Δ%	65.6	62.4	62.4	62.4
XBAR D				
BUSNS	2532	2451	2451	2446
POTS	2984	2937	2937	2937
CHANNELS	534	534	534	534
TH_MON	9600	9502	9502	9491
Δ%	0.0	1.0	1.0	1.1
DMS E				
BUSNS	19425	19399	19399	19399
POTS	3409	3409	3409	3409
CHANNELS	1721	1492	1492	1492
DS30 LEFT	58	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	154111	151828	151826	151828
Δ%	9.4	21.5	21.5	21.5
DMS F				
BUSNS	20912	20938	20938	20938
POTS	4511	4511	4511	4511
CHANNELS	1900	1624	1624	1624
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	170381	168165	168165	168165
Δ%	0.0	14.5	14.5	14.5
CD Cong.:	1	1	1	1
NRB:	1	1	1	1

Année= 3				
DMS A				
BUSNS	6125	6455	6455	6455
POTS	12639	12844	12844	12844
CHANNELS	1900	1705	1705	1705
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139158	141694	141656	141642
Δ%	0.0	10.3	10.3	10.3
DMS B				
BUSNS	10438	12388	12388	12388
POTS	17542	18802	18802	18802
CHANNELS	1900	1846	1846	1846
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	178931	202579	202538	202473
Δ%	0.0	2.8	2.8	2.8
DMS C				
BUSNS	7657	7657	7657	7657
POTS	3027	3027	3027	3027
CHANNELS	777	825	825	825
DS30 LEFT	26	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	69702	70181	70181	70181
Δ%	59.1	56.6	56.6	56.6
XBAR D				
BUSNS	2532	2451	2451	2446
POTS	2984	2937	2937	2937
CHANNELS	534	534	534	534
TH_MON	9600	9505	9505	9495
Δ%	0.0	1.0	1.0	1.1
DMS E				
BUSNS	20675	19649	19649	19649
POTS	3981	3409	3409	3409
CHANNELS	1852	1510	1510	1510
DS30 LEFT	62	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	165903	153594	153593	153594
Δ%	2.5	20.5	20.5	20.5
DMS F				
BUSNS	20912	21938	21938	21938
POTS	4511	5083	5083	5083
CHANNELS	1900	1712	1712	1712
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	170381	178082	178084	178082
Δ%	0.0	9.9	9.9	9.9
CD Cong.:	1	0	0	0
NRB:	1	0	0	0

Points de congestion: PC de E1-8 et E1-9 **Test No. E1-10 (suite)**

Année= 4	MOC	COM	COM	COM
DMS A	**0%**	**0%**	**1%**	**2%**
BUSNS	5605	6785	6785	6785
POTS	12639	13862	13862	13862
CHANNELS	1900	1823	1825	1825
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139160	151986	151947	151933
Δ%	0.0	4.1	3.9	3.9
DMS B				
BUSNS	10438	12615	12483	12453
POTS	17542	19318	19318	19318
CHANNELS	1900	1899	1892	1892
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	80	80	80
TH_MON	178931	208828	207879	207607
Δ%	0.0	0.1	0.4	0.4
DMS C				
BUSNS	7657	7657	7657	7657
POTS	21952	25820	25668	25865
CHANNELS	1900	1895	1886	1899
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	80	80	80
TH_MON	170383	191853	191043	192091
Δ%	0.0	0.3	0.7	0.1
XBAR D				
BUSNS	2532	2451	2451	2446
POTS	2984	2937	2937	2937
CHANNELS	530	529	529	529
TH_MON	9600	9512	9512	9503
Δ%	0.0	0.9	0.9	1.0
DMS E				
BUSNS	20879	19899	19899	19899
POTS	4554	3409	3409	3409
CHANNELS	1900	1527	1529	1528
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	170380	155381	155379	155380
Δ%	0.0	19.6	19.5	19.6
DMS F				
BUSNS	20912	22938	22938	22938
POTS	4511	5656	5656	5656
CHANNELS	1900	1792	1793	1793
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	170381	188030	188030	188031
Δ%	0.0	5.7	5.6	5.6
CD Cong.:	0	0	0	0
NRB:	1	3	3	2

Année= 5				
DMS A				
BUSNS	5165	6785	6785	6785
POTS	12639	14423	14322	14322
CHANNELS	1900	1897	1891	1892
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139155	157767	157243	157229
Δ%	0.0	0.2	0.5	0.4
DMS B				
BUSNS	10438	12178	12181	12178
POTS	17542	19318	19318	19318
CHANNELS	1900	1896	1895	1895
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	80	80	80
TH_MON	178931	206549	206467	206326
Δ%	0.0	0.2	0.3	0.3
DMS C				
BUSNS	7657	7517	7396	7133
POTS	21952	25820	25668	25865
CHANNELS	1900	1900	1890	1882
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	80	80	64
TH_MON	170383	190886	189228	188432
Δ%	0.0	0.0	0.5	0.9
XBAR D				
POTS	2984	2937	2937	2937
CHANNELS	345	340	340	340
TH_MON	4536	4536	4536	4535
Δ%	52.8	52.8	52.8	52.8
DMS E				
BUSNS	20879	20149	20149	20149
POTS	4554	3409	3409	3409
CHANNELS	1900	1554	1554	1554
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	170380	157126	157125	157123
Δ%	0.0	18.2	18.2	18.2
DMS F				
BUSNS	20912	23506	23506	23506
POTS	4511	6785	6785	6785
CHANNELS	1900	1887	1887	1888
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	80	80
TH_MON	170381	197868	197867	197867
Δ%	0.0	0.7	0.7	0.6
CD Cong.:	0	1	0	0
NRB:	1	4	3	3

Point de congestion: 70 DS30 (LEFT COM - RIGHT MOC) Test No. E1-11

Temps de calcul:
MOC: 1337s COM: 2336, 1318 et 1290s (0, 1 et 2%)

Année= 0	MOC	COM	COM	COM
DMS A	**0%**	**0%**	**1%**	**2%**
BUSNS	5465	5465	5465	5465
POTS	9791	9791	9791	9791
CHANNELS	1320	1242	1242	1242
DS30 LEFT	44	0	0	0
DS30 RIGHT	48	48	48	48
TH_MON	97529	97264	97261	97264
Δ%	37.1	31.4	31.4	31.4
DMS B				
BUSNS	10438	10438	10438	10438
POTS	17253	17253	17253	17253
CHANNELS	1883	1623	1623	1623
DS30 LEFT	63	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	176661	174396	174396	174396
Δ%	10.0	8.6	8.6	8.6
DMS C				
BUSNS	5157	5157	5157	5157
CHANNELS	406	501	501	501
DS30 LEFT	14	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	36099	36330	36330	36330
Δ%	80.0	54.3	54.3	54.3
XBAR D				
BUSNS	2532	2532	2532	2532
POTS	2937	2937	2937	2937
CHANNELS	539	539	539	539
TH_MON	9528	9662	9662	9662
DMS E				
BUSNS	17399	17399	17399	17399
POTS	3409	3409	3409	3409
CHANNELS	1559	1386	1386	1386
DS30 LEFT	52	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139929	137973	137973	137973
Δ%	25.7	8.6	8.6	8.6
DMS F				
BUSNS	17398	17398	17398	17398
POTS	3408	3408	3408	3408
CHANNELS	1559	1386	1386	1386
DS30 LEFT	52	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139916	137962	137962	137962
Δ%	25.7	8.6	8.6	8.6
CD Cong.:	1	1	1	1
NRB:	1	1	1	1

Année= 1				
DMS A				
BUSNS	5795	5795	5795	5795
POTS	10809	10809	10809	10809
CHANNELS	1462	1355	1355	1355
DS30 LEFT	49	0	0	0
DS30 RIGHT	48	48	48	48
TH_MON	109076	108792	108769	108754
Δ%	30.0	31.4	31.4	31.4
DMS B				
BUSNS	11188	11188	11188	11188
POTS	17770	17770	17770	17770
CHANNELS	1995	1704	1704	1704
DS30 LEFT	67	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	187223	184831	184789	184724
Δ%	4.3	8.6	8.6	8.6
DMS C				
BUSNS	6007	6007	6007	6007
POTS	1010	1010	1010	1010
CHANNELS	531	611	611	611
DS30 LEFT	18	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	47422	47723	47723	47723
Δ%	74.3	54.3	54.3	54.3
XBAR D				
BUSNS	2587	2587	2587	2587
POTS	3069	3069	3069	3069
CHANNELS	539	539	539	539
TH_MON	9839	9977	9977	9977
DMS E				
BUSNS	18399	18399	18399	18399
POTS	3409	3409	3409	3409
CHANNELS	1637	1446	1446	1446
DS30 LEFT	55	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	146929	144882	144882	144882
Δ%	21.4	8.6	8.6	8.6
DMS F				
BUSNS	19188	19188	19188	19188
POTS	3934	3934	3934	3934
CHANNELS	1734	1509	1509	1509
DS30 LEFT	58	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	155245	153052	153052	153052
Δ%	17.1	8.6	8.6	8.6
CD Cong.:	2	2	2	2
NRB:	1	1	1	1

Point de congestion: 70 DS30 (LEFT COM - RIGHT MOC) Test No. E1-11 (suite)

Année= 2	MOC	COM	COM	COM
DMS A	0%	0%	1%	2%
BUSNS	6125	6125	6125	6125
POTS	11826	11826	11826	11826
CHANNELS	1803	1591	1591	1591
DS30 LEFT	61	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	132345	131443	131406	131391
Δ%	12.9	8.6	8.6	8.6
DMS B				
BUSNS	11838	11838	11838	11838
POTS	18286	18286	18286	18286
CHANNELS	2097	1774	1774	1774
DS30 LEFT	70	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	196582	194125	194085	194021
Δ%	0.0	8.6	8.6	8.6
DMS C				
BUSNS	6832	6832	6832	6832
POTS	2018	2018	2018	2018
CHANNELS	654	713	713	713
DS30 LEFT	22	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	58559	58972	58972	58972
Δ%	68.6	54.3	54.3	54.3
XBAR D				
BUSNS	2642	2642	2642	2642
POTS	3201	3201	3201	3201
CHANNELS	534	534	534	534
TH_MON	10149	10291	10291	10291
DMS E				
BUSNS	19399	19399	19399	19399
POTS	3409	3409	3409	3409
CHANNELS	1715	1492	1492	1492
DS30 LEFT	58	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	153929	151821	151821	151821
Δ%	17.1	8.6	8.6	8.6
DMS F				
BUSNS	20938	20938	20938	20938
POTS	4511	4511	4511	4511
CHANNELS	1902	1626	1626	1626
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	170563	168158	168158	168158
Δ%	8.6	8.6	8.6	8.6
CD Cong.:	1	1	1	1
NRB:	1	1	1	1

Année= 3				
DMS A				
BUSNS	6455	6455	6455	6455
POTS	12844	12844	12844	12844
CHANNELS	1953	1709	1709	1709
DS30 LEFT	66	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	142699	141701	141663	141651
Δ%	5.7	8.6	8.6	8.6
DMS B				
BUSNS	11615	12388	12388	12388
POTS	18286	18802	18802	18802
CHANNELS	2100	1847	1847	1847
DS30 LEFT	70	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	196974	202573	202533	202468
Δ%	0.0	8.6	8.6	8.6
DMS C				
BUSNS	7657	7657	7657	7657
POTS	3027	3027	3027	3027
CHANNELS	777	823	823	823
DS30 LEFT	26	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	69702	70181	70181	70181
Δ%	62.9	54.3	54.3	54.3
XBAR D				
BUSNS	2697	2697	2697	2697
POTS	3332	3332	3332	3332
CHANNELS	534	534	534	534
TH_MON	10458	10606	10607	10606
DMS E				
BUSNS	19649	19649	19649	19649
POTS	3409	3409	3409	3409
CHANNELS	1738	1509	1509	1509
DS30 LEFT	58	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	155679	153585	153585	153585
Δ%	17.1	8.6	8.6	8.6
DMS F				
BUSNS	21938	21938	21938	21938
POTS	5083	5083	5083	5083
CHANNELS	2015	1711	1711	1711
DS30 LEFT	68	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	180607	178072	178072	178072
Δ%	2.9	8.6	8.6	8.6
CD Cong.:	0	0	0	0
NRB:	1	0	0	0

Point de congestion: 70 DS30 (LEFT COM - RIGHT MOC) Test No. E1-11 (suite)

Année= 4	MOC	COM	COM	COM
DMS A	**0%**	**0%**	**1%**	**2%**
BUSNS	6758	6785	6785	6785
POTS	13862	13862	13862	13862
CHANNELS	2100	1825	1825	1826
DS30 LEFT	**70**	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	152904	151988	151952	151935
Δ%	**0.0**	8.6	8.6	8.6
DMS B				
BUSNS	11336	12388	12388	12122
POTS	18286	18929	18838	18802
CHANNELS	2100	1872	1869	1851
DS30 LEFT	**70**	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	196971	205222	204704	202621
Δ%	**0.0**	8.6	8.6	8.6
DMS C				
BUSNS	7657	7657	7657	7657
POTS	25343	25349	25452	25452
CHANNELS	2100	1875	1880	1881
DS30 LEFT	**70**	0	0	0
DS30 RIGHT	80	64	64	64
TH_MON	188423	189342	189893	189890
Δ%	**0.0**	8.6	8.6	8.6
XBAR D				
BUSNS	2752	2752	2752	2752
POTS	3464	3464	3464	3464
CHANNELS	530	530	530	530
TH_MON	10769	10928	10928	10927
DMS E				
BUSNS	20218	19899	19899	19899
POTS	3409	3409	3409	3409
CHANNELS	1781	1527	1528	1529
DS30 LEFT	60	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	159662	155366	155365	155365
Δ%	14.3	8.6	8.6	8.6
DMS F				
BUSNS	22619	22938	22938	22938
POTS	5656	5656	5656	5656
CHANNELS	2100	1794	1794	1795
DS30 LEFT	**70**	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	188422	188013	188013	188011
Δ%	**0.0**	8.6	8.6	8.6
CD Cong.:	0	0	0	0
NRB:	1	3	2	2

Année= 5				
DMS A				
BUSNS	6190	6785	6785	6785
POTS	13862	14411	14403	14322
CHANNELS	2100	1900	1898	1894
DS30 LEFT	**70**	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	152905	157712	157634	157232
Δ%	**0.0**	8.6	8.6	8.6
DMS B				
BUSNS	11236	11973	12086	12122
POTS	18286	18929	18838	18802
CHANNELS	2100	1870	1871	1872
DS30 LEFT	**70**	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	196974	203090	203293	203229
Δ%	**0.0**	8.6	8.6	8.6
DMS C				
BUSNS	7657	7286	7268	7139
POTS	25343	25349	25452	25452
CHANNELS	2100	1868	1871	1864
DS30 LEFT	70	0	0	0
DS30 RIGHT	80	64	64	64
TH_MON	188423	186755	187178	186273
Δ%	0.0	8.6	8.6	8.6
XBAR D				
POTS	3464	3464	3464	3464
CHANNELS	392	392	392	393
TH_MON	5265	5347	5347	5347
DMS E				
BUSNS	21036	20703	20717	20574
POTS	4538	3409	3478	3409
CHANNELS	1910	1590	1593	1581
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	171394	160923	161381	160035
Δ%	8.6	8.6	8.6	8.6
DMS F				
BUSNS	22619	22952	22938	23081
POTS	5656	6785	6716	6785
CHANNELS	2100	1857	1853	1867
DS30 LEFT	70	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	188422	194043	193585	194929
Δ%	0.0	8.6	8.6	8.6
CD Cong.:	1	1	1	1
NRB:	1	4	4	4

Points de congestion: PC de E1-9 et E1-11 **Test No. E1-12**

Temps de calcul:
MOC: 1369s COM: 2059, 1521, 1436s (0, 1 et 2%)

Année= 0	MOC	COM	COM	COM
DMS A	**0%**	**0%**	**1%**	**2%**
BUSNS	5465	5465	5465	5465
POTS	9791	9791	9791	9791
CHANNELS	1320	1242	1242	1242
DS30 LEFT	44	0	0	0
DS30 RIGHT	48	48	48	48
TH_MON	97529	97264	97261	97264
Δ%	37.1	31.4	31.4	31.4
DMS B				
BUSNS	10438	10438	10438	10438
POTS	17253	17253	17253	17253
CHANNELS	1883	1623	1623	1623
DS30 LEFT	63	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	176661	174396	174396	174396
Δ%	10.0	8.6	8.6	8.6
DMS C				
BUSNS	5157	5157	5157	5157
CHANNELS	406	501	501	501
DS30 LEFT	14	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	36099	36330	36330	36330
Δ%	80.0	54.3	54.3	54.3
XBAR D				
BUSNS	2532	2532	2532	2532
POTS	2937	2937	2937	2937
CHANNELS	539	539	539	539
TH_MON	9528	9662	9662	9662
Δ%	0.8	-0.6	-0.6	-0.6
DMS E				
BUSNS	17399	17399	17399	17399
POTS	3409	3409	3409	3409
CHANNELS	1559	1386	1386	1386
DS30 LEFT	52	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139929	137973	137973	137973
Δ%	25.7	8.6	8.6	8.6
DMS F				
BUSNS	17398	17398	17398	17398
POTS	3408	3408	3408	3408
CHANNELS	1559	1386	1386	1386
DS30 LEFT	52	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	139916	137962	137962	137962
Δ%	25.7	8.6	8.6	8.6
CD Cong.:	1	1	1	1
NRB:	1	1	1	1

Année= 1				
DMS A				
BUSNS	5795	5795	5795	5795
POTS	10809	10809	10809	10809
CHANNELS	1462	1354	1354	1354
DS30 LEFT	49	0	0	0
DS30 RIGHT	48	48	48	48
TH_MON	109076	108792	108769	108752
Δ%	30.0	31.4	31.4	31.4
DMS B				
BUSNS	11188	11188	11188	11188
POTS	17770	17770	17770	17770
CHANNELS	1995	1704	1704	1704
DS30 LEFT	67	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	187223	184835	184793	184728
Δ%	4.3	8.6	8.6	8.6
DMS C				
BUSNS	6007	6007	6007	6007
POTS	1010	1010	1010	1010
CHANNELS	531	611	611	611
DS30 LEFT	18	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	47422	47723	47723	47724
Δ%	74.3	54.3	54.3	54.3
XBAR D				
BUSNS	2532	2451	2451	2446
POTS	2984	2937	2937	2937
CHANNELS	539	539	539	539
TH_MON	**9600**	**9500**	**9500**	**9489**
Δ%	**0.0**	**1.0**	**1.0**	**1.2**
DMS E				
BUSNS	18399	18399	18399	18399
POTS	3409	3409	3409	3409
CHANNELS	1637	1446	1446	1446
DS30 LEFT	55	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	146929	144887	144887	144887
Δ%	21.4	8.6	8.6	8.6
DMS F				
BUSNS	19188	19188	19188	19188
POTS	3934	3934	3934	3934
CHANNELS	1734	1508	1508	1508
DS30 LEFT	58	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	155245	153055	153055	153055
Δ%	17.1	8.6	8.6	8.6
CD Cong.:	2	2	2	2
NRB:	2	2	2	2

Points de congestion: PC de E1-9 et E1-11 Test No. E1-12 (suite)

Année= 2	MOC	COM	COM	COM
DMS A	**0%**	**0%**	**1%**	**2%**
BUSNS	6125	6125	6125	6125
POTS	11826	11826	11826	11826
CHANNELS	1803	1592	1592	1592
DS30 LEFT	61	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	132345	131438	131400	131385
Δ%	12.9	8.6	8.6	8.6
DMS B				
BUSNS	11838	11838	11838	11838
POTS	18286	18286	18286	18286
CHANNELS	2097	1775	1775	1775
DS30 LEFT	70	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	196582	194130	194090	194024
Δ%	0.0	8.6	8.6	8.6
DMS C				
BUSNS	6832	6832	6832	6832
POTS	2018	2018	2018	2018
CHANNELS	654	715	715	715
DS30 LEFT	22	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	58559	58970	58970	58970
Δ%	68.6	54.3	54.3	54.3
XBAR D				
BUSNS	2532	2451	2451	2446
POTS	2984	2937	2937	2937
CHANNELS	534	534	534	534
TH_MON	9600	9502	9502	9491
Δ%	0.0	1.0	1.0	1.1
DMS E				
BUSNS	19399	19399	19399	19399
POTS	3409	3409	3409	3409
CHANNELS	1715	1492	1492	1492
DS30 LEFT	58	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	153929	151828	151826	151828
Δ%	17.1	8.6	8.6	8.6
DMS F				
BUSNS	20938	20938	20938	20938
POTS	4511	4511	4511	4511
CHANNELS	1902	1624	1624	1624
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	170563	168165	168165	168165
Δ%	8.6	8.6	8.6	8.6
CD Cong.:	1	1	1	1
NRB:	1	1	1	1

Année= 3				
DMS A				
BUSNS	6455	6455	6455	6455
POTS	12844	12844	12844	12844
CHANNELS	1953	1705	1705	1705
DS30 LEFT	66	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	142699	141694	141656	141642
Δ%	5.7	8.6	8.6	8.6
DMS B				
BUSNS	11615	12388	12388	12388
POTS	18286	18802	18802	18802
CHANNELS	2100	1846	1846	1846
DS30 LEFT	70	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	196974	202579	202538	202473
Δ%	0.0	8.6	8.6	8.6
DMS C				
BUSNS	7657	7657	7657	7657
POTS	3027	3027	3027	3027
CHANNELS	777	825	825	825
DS30 LEFT	26	0	0	0
DS30 RIGHT	32	32	32	32
TH_MON	69702	70181	70181	70181
Δ%	62.9	54.3	54.3	54.3
XBAR D				
BUSNS	2532	2451	2451	2446
POTS	2984	2937	2937	2937
CHANNELS	534	534	534	534
TH_MON	9600	9505	9505	9495
Δ%	0.0	1.0	1.0	1.1
DMS E				
BUSNS	19649	19649	19649	19649
POTS	3409	3409	3409	3409
CHANNELS	1738	1510	1510	1510
DS30 LEFT	58	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	155679	153594	153593	153594
Δ%	17.1	8.6	8.6	8.6
DMS F				
BUSNS	21938	21938	21938	21938
POTS	5083	5083	5083	5083
CHANNELS	2015	1712	1712	1712
DS30 LEFT	68	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	180607	178082	178084	178082
Δ%	2.9	8.6	8.6	8.6
CD Cong.:	0	0	0	0
NRB:	1	0	0	0

Points de congestion: PC de E1-9 et E1-11 **Test No. E1-12 (suite)**

Année= 4	MOC	COM	COM	COM
DMS A	**0%**	**0%**	**1%**	**2%**
BUSNS	6758	6785	6785	6785
POTS	13862	13862	13862	13862
CHANNELS	2100	1825	1825	1825
DS30 LEFT	70	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	152904	151982	151943	151929
Δ%	0.0	8.6	8.6	8.6
DMS B				
BUSNS	11336	12388	12388	12388
POTS	18286	19194	19141	19141
CHANNELS	2100	1884	1882	1882
DS30 LEFT	70	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	196971	206614	206296	206231
Δ%	0.0	8.6	8.6	8.6
DMS C				
BUSNS	7657	7657	7657	7657
POTS	25343	25399	25378	25378
CHANNELS	2100	1879	1876	1876
DS30 LEFT	70	0	0	0
DS30 RIGHT	80	64	64	64
TH_MON	188423	189610	189497	189498
Δ%	0.0	8.6	8.6	8.6
XBAR D				
BUSNS	2532	2451	2451	2446
POTS	2984	2937	2937	2937
CHANNELS	529	529	529	529
TH_MON	9600	9511	9511	9502
Δ%	0.0	0.9	0.9	1.0
DMS E				
BUSNS	20218	19899	19899	19899
POTS	3409	3409	3409	3409
CHANNELS	1781	1529	1529	1529
DS30 LEFT	60	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	159662	155378	155378	155378
Δ%	14.3	8.6	8.6	8.6
DMS F				
BUSNS	22619	22938	22938	22938
POTS	5656	5656	5656	5656
CHANNELS	2100	1793	1793	1793
DS30 LEFT	70	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	188422	188027	188027	188027
Δ%	0.0	8.6	8.6	8.6
CD Cong.:	0	0	0	0
NRB:	1	3	2	2

Année= 5				
DMS A				
BUSNS	6190	6785	6785	6785
POTS	13862	14495	14432	14166
CHANNELS	2100	1903	1899	1886
DS30 LEFT	70	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	152905	158107	157766	156474
Δ%	0.0	8.6	8.6	8.6
DMS B				
BUSNS	11236	11811	11801	11543
POTS	18286	19194	19141	19141
CHANNELS	2100	1871	1869	1855
DS30 LEFT	70	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	196974	203370	202924	201022
Δ%	0.0	8.6	8.6	8.6
DMS C				
BUSNS	7657	7366	7398	7140
POTS	25343	25399	25378	25378
CHANNELS	2100	1873	1873	1860
DS30 LEFT	70	0	0	0
DS30 RIGHT	80	64	64	64
TH_MON	188423	187582	187693	185882
Δ%	0.0	8.6	8.6	8.6
XBAR D				
POTS	2984	2937	2937	2937
CHANNELS	342	340	340	340
TH_MON	4536	4535	4536	4536
Δ%	52.8	52.8	52.8	52.8
DMS E				
BUSNS	21036	20629	20641	20574
POTS	4538	3409	3409	3409
CHANNELS	1910	1586	1587	1583
DS30 LEFT	64	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	171394	160426	160508	160046
Δ%	8.6	8.6	8.6	8.6
DMS F				
BUSNS	22619	23026	23014	23081
POTS	5656	6785	6785	6785
CHANNELS	2100	1863	1861	1866
DS30 LEFT	70	0	0	0
DS30 RIGHT	64	64	64	64
TH_MON	188422	194566	194482	194939
Δ%	0.0	8.6	8.6	8.6
CD Cong.:	1	1	1	1
NRB:	4	4	4	4

C. Résultats partiels du BIP

Nous avons vu, à la page 68, que le BIP suppose que les trafics de débordement n'interagissent pas entre eux. On sait que cette supposition est fausse puisque le trafic moyen par liaison augmente avec la charge. Dans le but de vérifier si la supposition est acceptable, les modules LCM et BIP peuvent générer des fichiers de renseignements détaillés de leur activité (nombre de liaisons de tel type installées entre telle paire de commutateurs, les valeurs de ECCS, le problème BIP et sa solution, les statistiques de la résolution du BIP en terme de noeuds déployés et coupés pour le *branch-and-bound*, etc). Les deux pages qui suivent donne une partie d'un fichier type produit par le module LCM. Dans la première page, on retrouve le BIP sous forme condensé, les statistiques du *branch-and-bound*, ainsi que la solution retournée par le module BIP.

La partie qui nous intéresse se trouve à la deuxième page. La première série de résultats intitulée *BIP gaps* nous donne l'écart calculé par le BIP. Par exemple, la première contrainte du BIP, telle que lue au haut de la première page, nous donne $24x_1 + 12x_{46} \leq 21$. La solution optimale retournée par le module BIP (au bas de la première page) nous donne que $x_1 = 1$ et $x_{46} = 0$. L'écart calculé est donc de $24(1) + 12(0) - 21 = 3$ (row = 1, deuxième page). La deuxième série de résultats intitulée *Map BIP Final gaps* nous donne l'écart entre le nombre de liaisons installées et le nombre de liaisons nécessaires calculé. La troisième série (qui a été rajoutée pour les besoins d'analyse) nous donne la différence absolue entre les écarts. Si on néglige tous les écarts absolus tel que $|\Delta| \leq 1$ (que l'on peut attribuer à des erreurs d'arrondi), il nous reste les rangées 2 et 7 avec des écarts de 2 (ce qui est très bon) et la rangée 6 avec -4 (?). Ce -4 doit être pondéré par le fait que de façon relative cet écart est négligeable par rapport au nombre de liaisons que possède le commutateur de la rangée 6, soit 192. Des tests supplémentaires sont nécessaires pour déterminer s'il s'agit d'une déviation provenant du design où d'un phénomène "borné" et explicable.

```
--- BIP To Solve ---
Year = 0
 First PHU Index = 28
* Constraint # 1
j:      1 46     b
a(j):  24 13    21
* Constraint # 2
j:      2  3  4 42 43 44 45     b
a(j):  24 24 24  6 16 16  4    53
* Constraint # 3
j:      5  6 40 41 45 46     b
a(j):  24 24 11 12  4 10    45
* Constraint # 4
j:      7  8 38 39     b
a(j):  24 24  9  8    40
* Constraint # 5
j:      9 10 11 12 13 33 34 35 36 37 41 44     b
a(j):  24 24 24 24 24 11 12 12 14  7  9 15   101
* Constraint # 6
j:     14 15 16 17 30 31 32 37 40 43     b
a(j):  24 24 24 24  8  9 12  6  9 15    82
* Constraint # 7
j:     18 19 20 28 29 32 36 39 42     b
a(j):  24 24 24  4  7 16 17  6  7    69
* Constraint # 8
j:     21 22 29 35     b
a(j):  24 24  9 15    30
* Constraint # 9
j:     23 24 31 34     b
a(j):  24 24 12 14    47
* Constraint #10
j:     25 26 27 28 30 33 38     b
a(j):  24 24 24  2 11 13  5    51
** Cost vector
un-     1     2     3     4     5     6     7     8     9    10
  0   541   353   352   351   435   434   550   549   440   439
  1   438   437   436   439   438   437   436   416   415   414
  2   577   576   567   566   435   434   433   449   549   501
  3   577   635   501   577   940   635   240   485   506   691
  4   691   514   562   562   378   465

BandB...
zip =   10570
zip =   11080  (level  8)(node   8)
--- BandB Stats ---
nodes                        =   8
pruned by opt (non integer) =   0
pruned by opt (integer)      =   0
pruned by inf                =   0
backtracks                   =   0
(max level   8)
BandB end
---------

*** Solution
un-   1  2  3  4  5  6  7  8  9 10
  0   1  1  1  1  1  1  1  1  0  1
  1   1  1  1  0  1  1  1  0  1  1
  2   0  1  1  1  0  1  1  0  1  0
  3   0  1  1  0  0  0  0  0  0  0
  4   0  0  0  0  0  0
```

```
--- BIP gaps ---
gap =  3 for row =  1
gap = 19 for row =  2
gap =  3 for row =  3
gap =  8 for row =  4
gap =  6 for row =  5
gap =  2 for row =  6
gap =  2 for row =  7
gap =  3 for row =  8
gap =  1 for row =  9
gap = 10 for row = 10
--- Map BIP Final gaps ---
gap =  3  TotalLC =   48  err =  6.3%  row = 1
gap = 21  TotalLC =  192  err = 10.9%  row = 2
gap =  2  TotalLC =   96  err =  2.1%  row = 3
gap =  9  TotalLC =   48  err = 18.8%  row = 4
gap =  5  TotalLC =  240  err =  2.1%  row = 5
gap = -2  TotalLC =  192  err = -1.0%  row = 6
gap =  4  TotalLC =  120  err =  3.3%  row = 7
gap =  3  TotalLC =   48  err =  6.3%  row = 8
gap =  1  TotalLC =   96  err =  1.0%  row = 9
gap =  9  TotalLC =   96  err =  9.4%  row =10
*********************************

Différence absolue (Final Gaps - BIP Gaps)

Δ =  0 rangée  1
Δ =  2 rangée  2
Δ = -1 rangée  3
Δ =  1 rangée  4
Δ = -1 rangée  5
Δ = -4 rangée  6
Δ =  2 rangée  7
Δ =  0 rangée  8
Δ =  0 rangée  9
Δ = -1 rangée 10
```