Optimisation et validation de la charge utile d'un satellite de télécommunication

Fawzi Cheik Bessaih

THÈSE

Présentée pour obtenir le grade de Docteur en Sciences de l'Université d'Avignon et des Pays de Vaucluse France

SPÉCIALITÉ : Optimisation et Recherche Opérationnelle

École Doctorale 536 « SCIENCES ET AGROSCIENCES »
Laboratoire d'Informatique d'Avignon (UPRES No 4128)

Optimisation et validation de la charge utile d'un satellite de télécommunication

par

Fawzi Bessaïh

Soutenue publiquement le 30 septembre 2013 devant un jury composé de :

M.	Christophe Lecoutre	Président
M.	Gérard Verfaillie	Rapporteur
M.	Van Dat Cung	Rapporteur
M.	Sylvain Leconte	Industriel
M.	Bertrand Cabon	Co-directeur
M.	Philippe Michelon	Directeur
M.	Dominique Feillet	Co-directeur

Laboratoire LIA, Avignon *EASD Astrium*

Remerciements

Ce travail a été réalisé dans le cadre d'une thèse CIFRE au sein d'EADS Astrium et en collaboration avec le Laboratoire d'Informatique d'Avignon. L'encadrement universitaire a été mené par Philippe Michelon et Dominique Feillet respectivement directeur et co-directeur de thèse, tous deux représentants du LIA. Bertrand Cabon était chargé de l'encadrement industriel.

Je tiens tout d'abord à remercier Christophe Lemort, Jean-Claude Laire et Sylvain Leconte pour avoir initié cette thèse et pour m'avoir fait confiance. Sylvain, merci pour ta confiance et ton soutien, que ce soit dans le cadre de la thèse et en dehors.

Merci à Van Dat Cung et Gérard Verfaillie d'avoir accepté de rapporter cette thèse. Merci pour vos précieux retours et conseils. Je remercie également tous les membres du jury et leur président Christophe Lecoutre. Christophe Lecoutre, merci pour l'aide précieuse que vous m'avez apporté durant la thèse sur la problématique de validation.

Dominique, Philippe, merci pour votre support, votre patience et vos encouragements tout au long de ces années de thèse.

Lucas, David et Caroline, je tiens à vous remercier pour votre aide, votre bonne humeur, votre patience et votre soutien. Un grand merci à vous trois.

Un grand merci à Jean-Michel Dussauze et à toutes les membres d'ACE84 au sein du quel j'ai évolué durant cette thèse. Merci pour votre bonne humeur et votre disponibilité.

Bertrand, ces quelques lignes ne suffiront pas à exprimer toute ma reconnaissance pour ton implication, ton encadrement, ton soutien, ta patience et ta bonne humeur tout au long de ces années de thèse. Les résultats de cette thèse te doivent énormément.

Merci à toute l'équipe de la BD Télécom et plus particulièrement à Jean-Michel Charbonnier et à Cédric Lode pour leur précieux soutien lors de la soutenance, ainsi qu'à Malcolm Wallbridge pour son support sur les aspects métiers. Cédric, merci pour ton support, tes retours précieux et ta pugnacité lors l'utilisation de l'outil de routage dont certains aspects ergonomiques étaient perfectibles.

Merci à Samuel Couzinet pour sa disponibilité, son implication et son support sur la problématique de routage. Ton aide a été très précieuse.

Je tiens également à remercier toute ma famille, ma mère, mon père et plus particulièrement mon épouse, Amina, merci infiniment pour ton soutien sans faille, ta patience, merci d'avoir supporté mes sauts d'humeurs.

Je dédie cette thèse au petit bonhomme, Adem, né le 01/10/10 en pleine tourmente de fin de thèse.

Merci à tous.

Résumé

Pour répondre à une demande sans cesse croissante en capacité et débit pour la télévision HD, la sécurité civile, les télécommunications haut débit etc., les opérateurs satellites doivent accroître de manière importante la capacité de leurs nouveaux satellites. Ces nouvelles capacités sont obtenues au prix d'une complexité accrue. La maîtrise de cette complexité impose la définition de nouvelles méthodes de conception et le développement de nouveaux outils informatiques associés.

La conception des charges utiles de télécommunication répond aux contraintes particulières à l'environnement spatial qui se traduisent par des exigences draconiennes quant à la fiabilité, la masse des équipements à embarquer ou encore la limitation de la puissance disponible. EADS Astrium a ainsi développé une gamme de produits logiciels visant à valider et à optimiser les designs des satellites. Ces logiciels sont basés sur des modèles et des méthodes standards d'optimisation de la recherche opérationnelle. Ils sont utilisés lors des phases de conception, notamment pour traiter deux problématiques d'optimisation :

– La validation de la robustesse d'une charge utile nécessite la validation de plusieurs centaines de millions, voire plusieurs dizaines de milliards de cas dégradés. Pour chacun des cas dégradés envisagés, il faut effectuer un calcul de reconfiguration de la charge utile de façon à valider la robustesse du design à cette anomalie. Pour permettre une validation exhaustive d'une charge utile en un temps raisonnable, un calcul de reconfiguration complet ne doit pas dépasser quelques millisecondes.
– L'aménagement des équipements sur les murs d'un satellite est un processus complexe très contraint. Il nécessite de respecter le positionnement des équipements selon des zones chaudes et des zones froides préalablement définies. Afin de garantir un niveau d'amplification optimal, la longueur des guides d'ondes reliant les équipements doit être limitées et des contraintes de montage/démontage des guides doivent être respectées.

Avec l'accroissement de la complexité des nouvelles générations de satellites, il devient critique pour EADS Astrium de définir de nouvelles méthodes d'optimisation de la charge utile. Nous nous proposons d'étudier ici ces deux problématiques. Une première problématique d'aménagement de guides d'ondes se rapporte à une problématique de multi-routage dans un espace continu en trois dimensions. L'objectif est de fournir une estimation des pertes radio-fréquentielles directement déduite des longueurs des guides d'ondes obtenues lors du routage. Cette problématique s'inscrit dans le cadre de problématique de type **Pipe Routing** (littéralement, routage de canalisation).

La seconde problématique concerne la validation de la robustesse d'une charge utile à un nombre de pannes préalablement fixé. Du fait d'une méthode de validation choisie, basée sur une énumération exhaustive des cas de pannes, cette problématique s'apparente à la résolution successive de millions voire de milliards de problèmes de satisfaction de contraintes. Avec l'augmentation de la complexité des charges utiles, il devient urgent de proposer des améliorations significatives de l'outil de validation existant, à savoir **SWITCHWORKS**.

Ce mémoire est organisé de la manière suivante :

Tout d'abord, le chapitre 1 introduit le contexte industriel, les problématiques traitées dans cette thèse ainsi que les notions métiers nécessaires à la compréhension des problématiques.

Ensuite, le mémoire est organisé en trois parties distinctes, une pour chaque problématique et une conclusion générale qui vient clore le mémoire.

La partie I correspondante à la problématique de routage est organisée de la manière suivante :

– Le chapitre 1 introduit la problématique de routage.

– Le chapitre 2 propose un état de l'art de la problématique de type **Pipe Routing**.
– Le chapitre 3 introduit une modélisation mathématique de la problématique de multi-routage et la méthode de résolution exacte associée mettant en oeuvre une méthodologie de type Branch and Price. Une description des instances de test et une présentation des résultats, ainsi qu'une conclusion clôtureront ce chapitre.
– Le chapitre 4 introduit deux approches heuristiques, toutes deux basées sur des ajustements de chemins valides. L'une met en œuvre un algorithme de type glouton associé à un mécanisme de réparation locale, l'autre est basée sur la génération de colonnes. Une présentation des résultats et une conclusion clôtureront ce chapitre.
– Le chapitre 5 introduit un modèle réaliste avec l'ajout de nouvelles contraintes. Le modèle réaliste sera développé exclusivement pour une méthode de résolution approchée de type de glouton, couplée à un mécanisme de réparation. Une présentation des résultats et une conclusion clôtureront ce chapitre.
– Le chapitre 6 fait office de conclusion et expose les perspectives envisagés.

La partie II correspondante à la problématique de validation est organisée de la manière suivante :

– Le chapitre 1 introduit la problématique de validation.
– Le chapitre 2 propose un état de l'art des techniques de résolution d'un Problème de Satisfaction de Contraintes.
– Le chapitre 3 expose les améliorations qui ont permis une réduction significative du temps de validation.
– Le chapitre 4 liste, pour des raisons documentaires, les tentatives infructueuses d'amélioration et de réduction de la combinatoire.
– Le chapitre 5 fait office de conclusion et expose les perspectives envisagés.

TABLE DES MATIÈRES

TABLE DES FIGURES

Liste des tableaux

Liste des Algorithmes

Chapitre 1

Introduction générale

EADS Astrium Satellites dispose d'un savoir-faire complet en matière de satellite de télécommunication : étude système, conception, fabrication et essais, services de lancement et exploitation en orbite, centre de contrôle et de communication. Il est maître d'oeuvre de plus de quatre-vingt-dix satellites de télécommunication géostationnaires. Il assure une exceptionnelle fiabilité et disponibilité opérationnelle des satellites au service des plus grands opérateurs dans le monde. Ces satellites sont utilisés pour des services de communication fixe et mobile, la diffusion en directe de télévision et de radio numérique, l'internet large bande...etc.

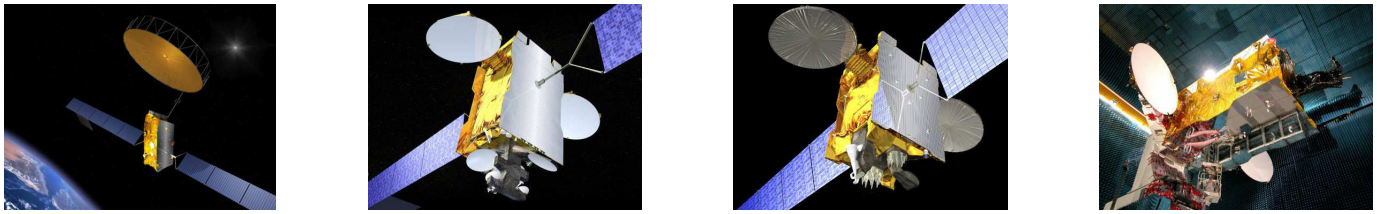


Figure 1.1 – Satellites de télécommunication

Pour répondre à une demande sans cesse croissante en capacité et débit les opérateurs de satellites de télécommunication doivent accroître significativement la capacité de leurs nouveaux satellites. Ces nouvelles capacités sont obtenues au prix d'une complexité accrue, qui se traduit notamment par une augmentation significative du nombre d'équipements et de connexions au sein des charges utiles qui assurent la transmission des signaux reçus par un satellite de télécommunication.

La maîtrise de cette complexité impose l'amélioration des méthodes d'optimisation existantes et la définition de nouvelles. Ces méthodes d'optimisation interviennent principalement lors de deux phases bien distinctes. Tout d'abord, lors de la phase de conception où elles permettent aux concepteurs d'optimiser le dimensionnement de la charge utile et d'en valider le design, et durant la phase de tests une fois le satellite assemblé. Cette dernière phase permet notamment de tester la charge utile dans une chambre à vide simulant le vide spatial et l'alternance entre phases chaudes et phases froides, simulant respectivement l'exposition au soleil et l'exposition à l'ombre de la terre.

Dans cette thèse on se propose d'étudier deux problématiques d'optimisation rencontrées lors de la phase de conception. Une première problématique est le multi-routage de guides d'ondes dans un espace continu en trois dimensions, dont le but est de fournir une estimation des pertes radio-fréquentielles, directement déduite des longueurs de guide d'ondes. Cette problématique s'inscrit dans le cadre de problématique de type **Pipe Routing** (littéralement, routage de canalisation).

La seconde problématique concerne la validation de la robustesse d'une charge utile à un nombre de pannes

préalablement fixé. Du fait d'une méthode de validation basée sur une énumération exhaustive des cas de pannes, cette problématique s'apparente à une résolution successive de millions voire de milliards de problèmes de satisfaction de contraintes.

Dans le cadre d'une Convention Industrielle de Formation par la Recherche (CIFRE) et de la collaboration entre EADS Astrium et le Laboratoire d'Informatique d'Avignon (LIA), l'objectif de cette thèse est de proposer des solutions innovantes permettant de traiter les deux problématiques. Cette thèse propose donc :
– d'une part, un outil d'optimisation du routage de guides d'ondes utilisable par des ingénieurs,
– d'autre part, des améliorations significatives des performances de l'outil de validation existant, à savoir **SWITCHWORKS**.
Nous allons à présent donner les bases nécessaires au lecteur, ceci afin de faciliter la compréhension des problématiques liées aux satellites de télécommunication.

Un satellite de télécommunication est constitué d'une plate-forme et d'une charge utile. La plate-forme assure le maintien à poste (en orbite) du satellite, le contrôle thermique, la régulation électrique, l'interface sol...etc. La charge utile, appelée **Payload** en anglais (littéralement charge payante) permet de recevoir, d'amplifier et de transmettre un signal large bande vers la terre. La charge utile constitue l'élément central d'un satellite de télécommunication.

Un signal large bande émis de la terre, parcourt 36 000 km pour atteindre un satellite de télécommunication géostationnaire. Le signal reçu, appelé signal montant ou entrant est alors très faible, de l'ordre de 10^{-10} watts du fait de l'atténuation subit tout au long des 36 000 km. Afin de transmettre le signal vers la terre dans les meilleurs conditions (signal exploitable), le signal doit être fortement amplifié pour atteindre 10^2 watts. Cette amplification est précédée d'une translation fréquentielle pour éviter tout phénomène d'interférence entre le signal montant et le signal émis, appelé signal descendant ou sortant. La figure 1.2 illustre le cheminement d'un signal.

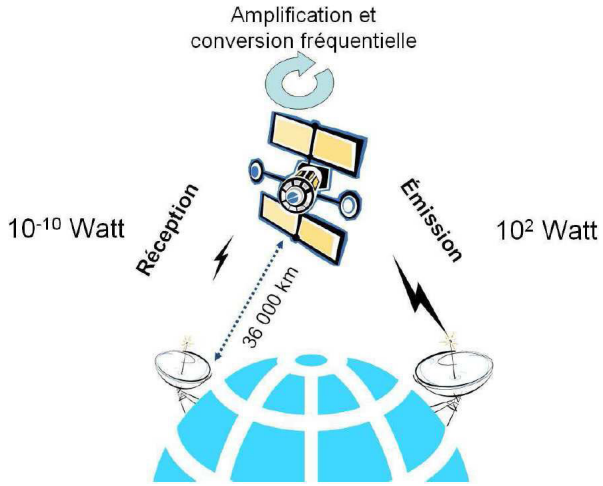


FIGURE 1.2 – Système de communication par satellite

La charge utile est constituée d'une multitude de composants. Chaque composant a un rôle bien précis dans la chaîne d'amplification du signal reçu.

1.1 La charge utile de télécommunication

Actuellement, la plupart des satellites de télécommunication ont une architecture assez similaire. La figure 1.3 résume l'acheminement d'un signal à large bande dans un satellite classique (ne comprenant pas de processeur

embarqué). Un signal à large bande est composé de plusieurs signaux à bande étroite qu'on nomme plus communément canaux. À la réception du signal à large bande, ce dernier est démultiplexé en plusieurs canaux. Les canaux sont amplifiés simultanément pour être ensuite multiplexés en un seul signal large bande et réemis vers la Terre.

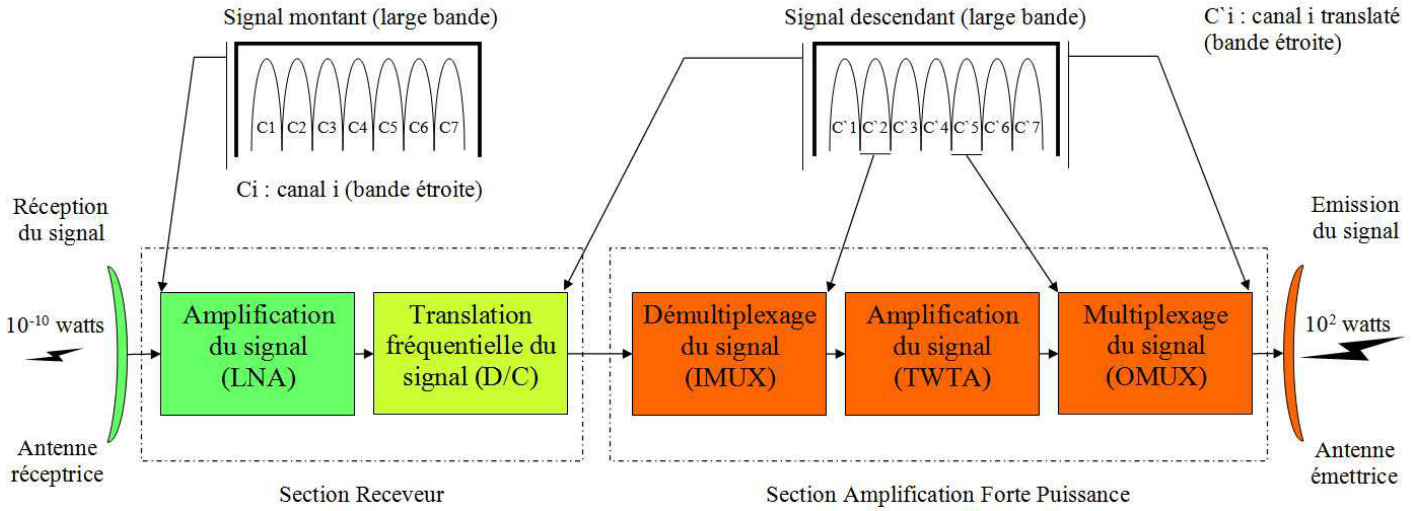


FIGURE 1.3 – Schéma simplifié de la charge utile

Les composants intervenant directement dans le traitement et l'amplification du signal reçu sont décrit ci-dessous :

IMUX (Input Multiplexer) et OMUX (Output Multiplexer) un IMUX permet de séparer et de filtrer un signal à large bande en plusieurs canaux (signaux à bande étroite – 30 MHz). Un OMUX, quant à lui, permet de recombiner plusieurs canaux en un seul signal à large bande. Cette opération est effectuée après amplification de chaque canal.

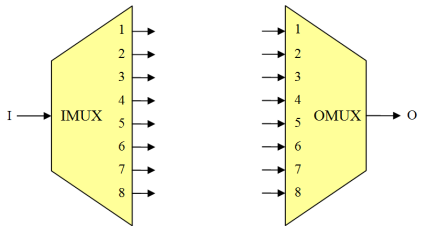


FIGURE 1.4 – 1-to-8 IMUX et 8-to-1 OMUX

LNA (Low Noise Amplifier) effectue une première amplification du signal avec un minimum de bruit. Le signal à large bande est de quelques centaines de picowatts en entrée du LNA et de quelques microwatts en sortie. Afin de minimiser la perte d'information, ces LNAs sont placés au plus près de l'antenne réceptrice.

D/C (Down Converter) assure une conversion fréquentielle basse du signal à large bande. Cette conversion permet de différencier le signal montant du signal descendant. Comme illustré par la figure 1.5, un mélangeur associe un signal utile de fréquence F_{utile} et un signal de fréquence F_{LO} pour obtenir un signal utile translaté en fréquence selon la fréquence LO. À titre d'exemple, l'ordre de grandeur de la conversion est de quelques GHz en bande C, une bande très largement utilisée en télécommunication.

TWTA (Travelling Wave Tube Amplifier) est un tube à ondes progressives. C'est l'équipement central de la charge utile. Il consomme à lui seul de 90% à 95% de la puissance électrique du satellite. Il permet d'amplifier fortement un canal par échange d'énergie avec un faisceau d'électrons. Cependant, il ne permet

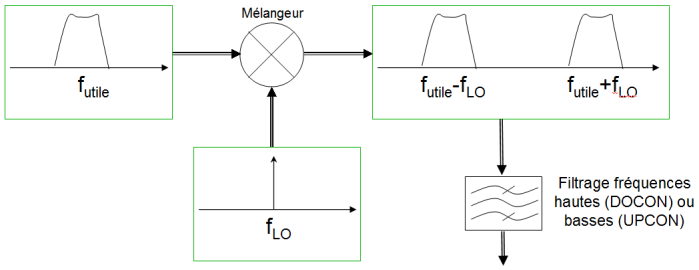


FIGURE 1.5 – Principe de la conversion fréquentielle

pas d'amplifier fortement un signal à large bande. C'est pour cette raison que le signal à large bande est démultiplexé en canaux. La figure 1.6 illustre un TWTA avec : (1) Cathode ; (2) Entrée du signal hyperfréquence à amplifier (ici, connecteur coaxial) ; (3) Aimants permanents ; (4) Atténuateur ; (5) Hélice ; (6) Sortie du signal amplifié (ici, connecteur coaxial) ; (7) Enveloppe ; (8) Collecteur d'électrons.

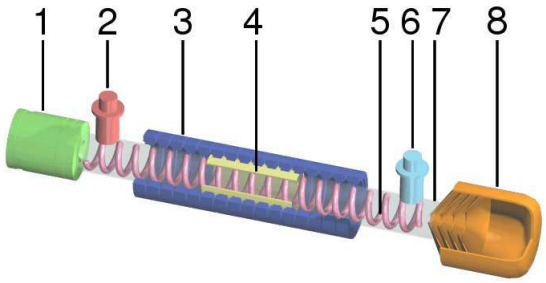
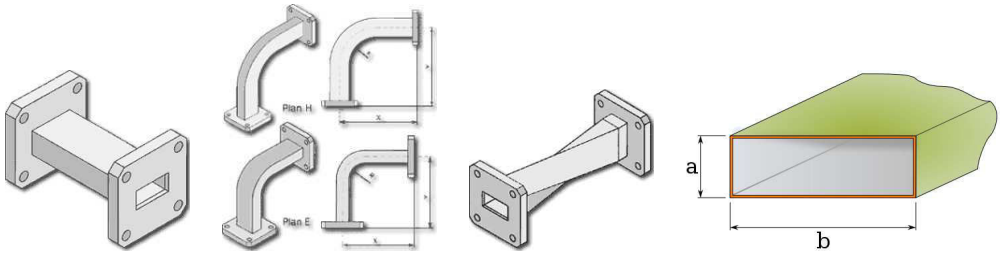


FIGURE 1.6 – Travelling Wave Tube Amplifier

Waveguide (littéralement Guide d'ondes) est un élément creux de coupe rectangulaire permettant de transmettre (de guider) une onde électromagnétique fortement amplifiée entre deux équipements de la charge utile. Dans la cas d'un signal de faible intensité, un cable coaxial est utilisé pour acheminer le dit signal. Les figures (1.7-a), (1.7-b), et (1.7-c) illustrent, respectivement un guide d'ondes en section droite, un coude et un twist. Les paramètres physiques d'un guide d'ondes sont la largeur a, la hauteur b (1.7-d) et un rayon de courbure (1.7-b). Ces paramètres dépendent de la puissance du signal à acheminer.



(a) (b) (c) (d)

FIGURE 1.7 – Waveguides : Guides d'ondes

Switche est un équipement permettant de relier plusieurs segments de guide d'ondes (ou de cable coaxial). Selon le type de switche utilisé et la position sélectionnée, il est possible d'acheminer un, deux ou trois signaux simultanément. Durant son fonctionnement, la position d'un switche peut être amenée à changer pour permettre la redirection des signaux le traversant. Ce cas de figure sera exposé plus en detail dans la suite du rapport. Il existe trois types de switches offrant différentes configurations d'acheminement de signaux, comme illustré par la figure 1.8.

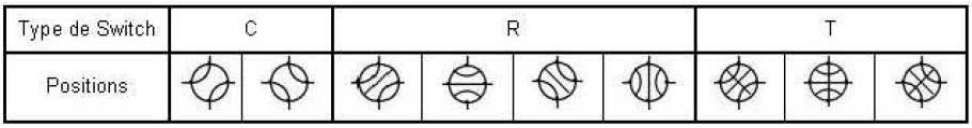


FIGURE 1.8 – Différents types de switches et leurs positions respectives

Chaque composant décrit ci-dessus est une partie intégrante de la charge utile. Le design de cette dernière répond à des besoins spécifiques par la mise en place de différents mécanismes qui permettent de passer d'une chaîne d'amplification simple à une charge utile pouvant traiter un signal à large bande.

1.2 D'une chaîne d'amplification simple vers la charge utile

La chaîne d'amplification illustrée schématiquement par la figure 1.9 permet d'amplifier un canal. Après réception du signal, une première opération consiste à lui faire subir une première phase d'amplification faible bruit (LNA) et de lui appliquer une translation fréquentielle (D/C). Le signal est ensuite filtré avant d'être fortement amplifié (TWTA), pour être ensuite filtré une seconde fois, acheminé au canal de sortie et retransmis vers la terre via l'antenne émettrice. Les composants se trouvant dans la section délimitée par l'antenne réceptrice et le TWTA sont interconnectés via des câbles coax. Quant aux composants se trouvant dans la section délimitée par le TWTA et l'antenne émettrice, ils sont interconnectés via des waveguides, du fait de la puissance du signal après amplification.

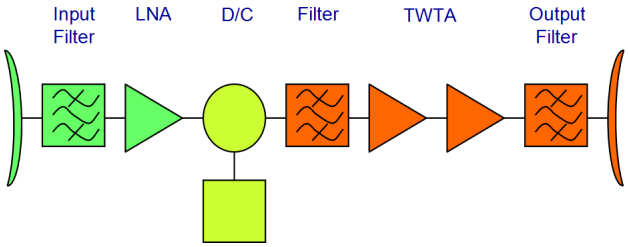


FIGURE 1.9 – Chaîne d'amplification

Contrairement à la chaîne d'amplification présentée par la figure 1.9, la figure 1.10 illustre un design multicanaux permettant de traiter simultanément plusieurs canaux issus de la décomposition d'un signal à large bande via un démultiplexeur (IMUX). Pour la phase d'amplification il est nécessaire d'avoir autant de TWTA que de canaux à amplifier. Une fois amplifiés, les canaux sont recombinés en un signal à large bande via un multiplexeur (OMUX). L'exemple de la figure 1.10 permet d'amplifier trois canaux du fait de la présence de trois TWTAs.

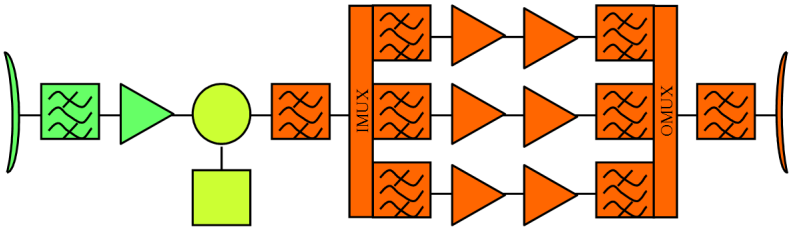


FIGURE 1.10 – Design multicanaux

Durant son existence, un satellite de télécommunication peut être sujet à un certain nombre de pannes, notamment celles des TWTAs. Pour y remédier, un mécanisme de redondance est mis en place. Au vue de la multiplication du nombre de TWTAs (design multicanaux), une redondance 2 :1 (un redondant pour chaque TWTA) n'est pas envisageable du fait du prix élevé des TWTAs, de l'ordre de plusieurs dizaines de milliers

d'euros. Généralement, de quatre à six TWTAs redondants sont prévus pour un ensemble de TWTAs nominaux. Ce mécanisme de redondance repose sur la présence de deux matrices de switches symétriques.

Une première matrice, entre les canaux d'entrée et les tubes amplificateurs (section d'entrée) garantie l'accès à n'importe quel TWTA redondant si besoin, voire à tous les TWTAs redondant simultanément.

Une seconde matrice, entre les tubes amplificateurs et les canaux de sortie (section de sortie) garantie la redirection des signaux sortant des tubes redondants vers les canaux de sortie. La figure 1.11 illustre une redondance de deux TWTAs pour trois TWTAs nominaux, avec deux matrices de redondance. Le niveau de redondance définit le degré de robustesse du satellite aux cas de panne. Ainsi, une matrice de redondance est conçue pour être robuste à k cas de pannes (généralement $k = 1..6$).

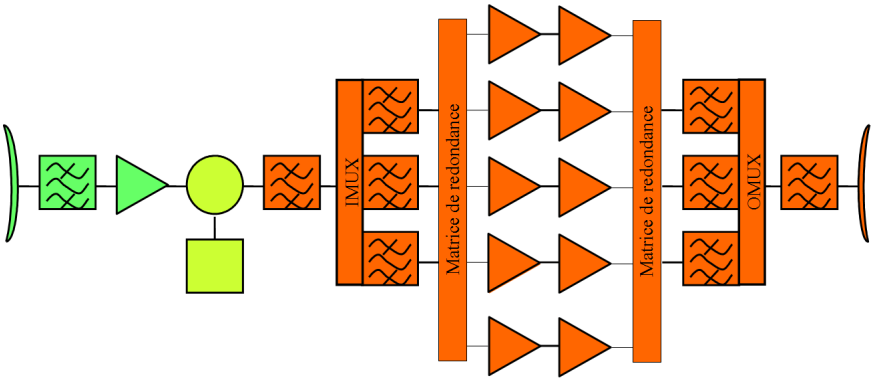


FIGURE 1.11 – Mécanisme de redondance : Matrice de switches

L'une des particularités d'un satellite de télécommunication est de pourvoir couvrir plusieurs zones géographiques. Cela se traduit schématiquement sur la figure 1.12 par la présence de deux antennes réceptrices, contrairement au schéma précédant qui n'en possédait qu'une seule. Traiter simultanément des signaux provenant de plusieurs zones géographiques nécessite la mise en place d'une matrice de switches dite de sélectivité. La matrice de sélectivité permet de sélectionner les canaux à traiter, appelés canaux actifs. Ces canaux actifs définissent la mission opérée par le satellite. La figure 1.12 illustre un design robuste à trois pannes de TWTA et couvrant deux zones géographiques. Comme pour le mécanisme de redondance, le mécanisme de sélectivité est composé de deux matrices, une première matrice de sélectivité en section d'entrée et une seconde en section de sortie. Dans la majorité des cas, les matrices de switches (redondance + sélectivité) de la section d'entrée et de sortie sont symétriques. Cette particularité aura son importance par la suite. Dans la suite du mémoire, une matrice de switches est composée d'une matrice de sélectivité et d'une matrice de redondance.

Ces différents aspects définissent la composition de la charge utile d'un satellite de télécommunication, une charge utile qui détermine le dimensionnement du satellite.

1.3 Problématiques liées aux satellites de télécommunication

Lors de la conception d'un satellite de télécommunication, le dimensionnement de ce dernier est une étape cruciale. Elle permet de définir les caractéristiques physiques et techniques du satellite répondant aux spécifications du cahier des charges. Ces caractéristiques concernent principalement la plate-forme et la charge utile embarquées. Pour la plate-forme, elles définissent les éléments permettant le maintien à poste (en orbite) du satellite, les éléments de contrôle thermique, de régulation électrique et d'interfaçage avec le sol...etc. En ce qui concerne la charge utile, les caractéristiques définissent essentiellement le nombre, le positionnement des équipements (notamment le nombre de TWTA) et la connectique entre les équipements.

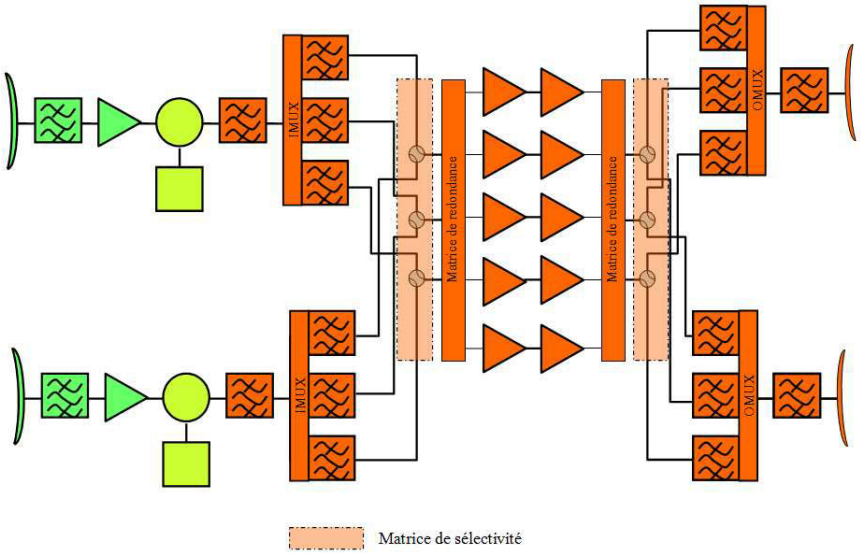


FIGURE 1.12 – Mécanisme de redondance et de sélectivité : Matrice de switches

Le design final de la charge utile est le résultat de plusieurs itérations. Une itération consiste à valider une proposition de design par rapport aux spécifications d'un point de vue technique, fonctionnel et budgétaire. En cas de non conformité, le design est modifié et revalidé, ainsi de suite, jusqu'à obtention de la conformité. À cet effet, un concepteur a à sa disposition des outils de conception et de validation lui permettant d'effectuer ces itérations en un temps réduit. Mais avec l'augmentation de la complexité des charges utiles, la lenteur de certains outils et l'apparition de nouveaux besoins en termes d'optimisation rendent la tâche plus difficile aux concepteurs, voire impossible dans un temps raisonnable. C'est notamment le cas de la validation de la robustesse d'une matrice de switches, qui, avec une complexité croissante nécessite dans certains cas plusieurs mois de validation, ce qui réduit considérablement le nombre d'itérations.

L'accroissement de la complexité a aussi créé de nouveaux besoins en termes d'optimisation. C'est le cas de la nécessité de disposer d'un outil permettant d'effectuer le routage de plusieurs centaines de guides d'ondes. Ceci permettrait d'estimer les pertes radio-fréquentielles et de proposer un routage préliminaire servant de base au routage final. Plusieurs itérations de routage sont nécessaires avant l'obtention d'un routage préliminaire conforme aux spécifications. L'une des variantes entre les différentes itérations est le positionnement des composants à connecter. Il est alors possible de modifier le positionnement des composants afin de réduire la longueur des guides.

Dans cette thèse nous nous proposons de répondre à l'accroissement de la complexité de la charge utile par des améliorations significatives de l'outil existant de validation de matrices de switches et par la définition d'un nouvel outil d'optimisation permettant d'effectuer un multi-routage en un temps raisonnable afin de favoriser la multiplication des itérations.

Pour traiter la problématique de multi-routage nous avons défini une méthode de routage de type heuristique basée sur un mécanisme de réparation des conflits. La particularité de la méthode que nous proposons réside dans l'utilisation d'un mécanisme de réparation et d'un algorithme de recherche de plus court chemin à voisinage étendu, ce qui nous permet de prendre en compte des contraintes liées à la structure même d'un guide d'ondes. Cette méthode est appliquée à un modèle de routage réaliste. Des expérimentations menées sur des satellites existants nous ont permis de prouver l'efficacité de notre approche. Cette efficacité a été mesurée selon l'écart entre les estimations de longueur fournies par l'approche et les longueurs réelles, ceci pour plusieurs centaines de guides d'ondes, et selon le temps d'exécution, de 5 à 10 minutes pour plusieurs centaines de guides d'ondes.

Pour cette même problématique, nous avons également proposé une approche de résolution exacte basée sur

la méthodologie Branch-and-Price, et deux heuristiques, l'une basée sur une méthode de réparation simple voisinage, la seconde sur la génération de colonnes, toutes les trois appliquées à un modèle de routage simplifié. Du fait de la modélisation, la méthode exacte, qui doit théoriquement converger vers une solution optimale en un temps fini, ne permet pas d'obtenir une première solution entière en un temps raisonnable, même dans le cas de routages simples. La génération de colonnes quant à elle fournit une solution relaxée dont la valeur de la fonction objective donne une borne inférieure à la solution entière optimale. Cette borne inférieure nous permet de démontrer la qualité des solutions fournies par la méthode de réparation simple voisinage.

La problématique de validation du design d'une charge utile s'apparente quant à elle à une succession de problèmes de satisfaction de contraintes fortement combinatoires. La problématique de validation consiste à valider le design d'une matrice de switches afin de déterminer les cas de pannes pour lesquels une reconfiguration de la charge utile est impossible. Le mécanisme de validation adopté est basé sur une énumération exhaustive des cas dégradés (cas de pannes), de sorte que chaque cas, appelé combinaison soit validé indépendamment des combinaisons antérieures. La validation consiste à identifier, si elles existent, l'ensemble des combinaisons pour lesquelles une reconfiguration de la charge utile n'est pas réalisable. Du fait de l'accroissement de la complexité, l'outil de validation existant **SWITCHWORKS** arrive à ses limites face à l'explosion de la combinatoire. Dans ce mémoire, nous nous proposons, dans un premier temps des améliorations algorithmiques permettant une réduction substantielle du temps de validation. Dans un second temps, nous exposerons, dans un souci documentaire, les tentatives non concluantes, notamment celles concernant la réduction de la combinatoire.

Première partie

Routage de guides d'onde dans un satellite de télécommunication

Chapitre 1

Problématique de routage de guides d'ondes

La capacité d'amplification d'un satellite de télécommunication est étroitement liée à son dimensionnement. Ce dimensionnement est défini pendant la phase de conception en réponse à un cahier des charges.

1.1 Dimensionnement d'un satellite

Le dimensionnement d'un satellite de télécommunication est une étape cruciale lors de la conception de ce dernier. Il permet de définir les caractéristiques physiques et techniques du satellite pour répondre au cahier des charges. Les caractéristiques définies sont celles de la plate-forme, que ce soit les éléments permettant le maintien à poste (en orbite) du satellite, les éléments de contrôle thermique, de régulation électrique, d'interfaçage avec le sol...etc., et celles de la charge utile, qui se résument essentiellement dans le nombre et le positionnement des équipements, notamment le nombre de TWTA. Ces différentes caractéristiques conditionnent entre autres les capacités de dissipation de la chaleur, de stockage de la puissance électrique et d'amplification. La mission principale d'un satellite de télécommunication étant la réception, l'amplification et la transmission de signaux à large bande, la puissance d'amplification devient une caractéristique prépondérante et qui plus est se trouve être étroitement liée au dimensionnement.

La capacité d'amplification doit être garantie malgré les pertes radio fréquentielles rencontrées tout au long du processus de traitement du signal, comprenant l'amplification. Afin d'estimer au plus juste le dimensionnement garantissant un niveau d'amplification suffisant à moindre coût, les concepteurs doivent disposer d'une évaluation fine des pertes radio fréquentielles. Ces pertes se traduisent par une dissipation de la puissance électrique dans les équipements sous forme de chaleur. Cette dissipation se produit lors de l'amplification sous la forme de pertes fixes et par l'atténuation de la puissance des signaux lors de leur acheminement entre les équipements via les guides d'ondes. Cette dernière est une perte proportionnelle à la longueur des guides.

Limiter ces pertes et les estimer au plus juste est primordial au bon dimensionnement du satellite, car des pertes trop importantes ou leur sous estimation entraîneront soit l'augmentation de la puissance des TWTAs, soit l'augmentation de leur nombre pour compenser la puissance perdue. De telles modifications nécessiteront l'accroissement des capacités de dissipation de la chaleur pour cause d'augmentation de la puissance dissipée sous forme de chaleur, et l'accroissement des capacités de stockage de la puissance électrique pour faire face à une demande de puissance électrique plus importante. En conséquence, dimensionnement et le coût de production du satellite devront être réévalués.

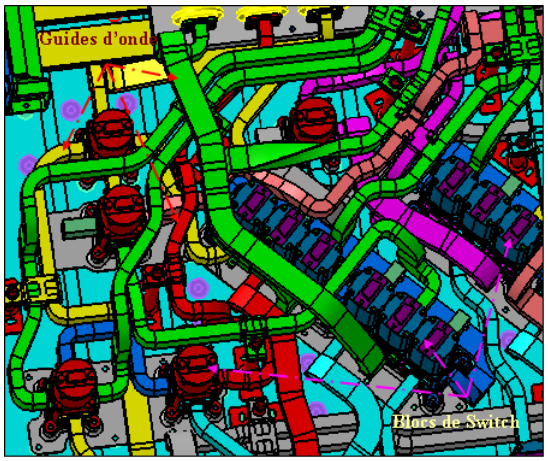


FIGURE 1.1 – Exemple de routage en 3D (CAO)

Les pertes radio fréquentielles peuvent être limitées en minimisant la longueur des guides d'ondes reliant les équipements, mais aussi en optimisant l'aménagement de ces mêmes équipements afin de réduire la distance qui les sépares. Or, pour un rendement optimal, l'ensemble des équipements du satellite doivent être maintenus dans une gamme de température. C'est une problématique majeure au niveau de l'accommodation du satellite, notamment pour les éléments fortement dissipatifs tels que les TWTAs qui consomment à eux seuls 90% à 95% de la puissance électrique du satellite, dont 60% est dissipée sous forme de chaleur. Des caloducs, éléments conducteurs de chaleur sont positionnés en surface et à l'intérieur des murs du satellite afin de créer trois zones de température, deux zones froides et une zone chaude. Les équipements y sont alors positionnés suivant leurs gammes de température de fonctionnement optimal. De ce fait, le positionnement des équipements se trouve être très contraint, ainsi la minimisation des pertes radio fréquentielles est exclusivement liée à la minimisation de la longueur des guides d'ondes.

A l'heure actuelle, les concepteurs quantifient ces pertes en mesurant manuellement, à l'aide d'une règle la longueur des guides d'ondes sur un diagramme logique. En parallèle, des équipes de CAO effectuent, sur une durée de 4 à 6 mois, une modélisation en trois dimensions de l'aménagement des équipements et des guides d'ondes. Ce délai s'explique par les multiples modifications d'aménagement qui sont effectuées afin d'optimiser la longueur des guides et par la même occasion les pertes radio fréquentielles. Un exemple de routage en trois dimensions est illustré par la figure 1.1, où un certain nombre de guides d'ondes connectant des blocs de switches sont représentés. Une fois la modélisation effectuée, après 4 à 6 mois d'attente, les concepteurs peuvent être amenés à reconsidérer le dimensionnement dans le cas où les performances ne sont pas atteintes. Avec l'accroissement de la complexité des satellites de télécommunication, il devient primordial de disposer rapidement d'estimations fines afin de faciliter et d'accélérer les échanges entre les différentes équipes de conception.

L'estimation et la minimisation de la longueur des guides d'ondes passe obligatoirement par le routage des ces derniers. Comme dit précédemment, les guides d'ondes permettent d'interconnecter les équipements se trouvant entre les TWTAs et l'antenne émettrice. Cette section regroupe les matrices de redondance et de sélectivité. Elles sont composées d'un nombre important de switches qui nécessitent un nombre important de connexions, donc de guides d'ondes. Cette caractéristique accentue la difficulté de la phase de routage. La figure 1.1 illustre une portion d'une matrice de redondance où l'on distingue très clairement l'enchevêtrement de guides d'ondes, ce qui illustre la difficulté du routage à accomplir.

La problématique étudiée est une problématique de multi-routage dans un espace continu en trois dimensions. Cette problématique s'inscrit dans le cadre de problématique de **Pipe Routing** (littéralement, routage de canalisation) dont un état de l'art est exposé dans le chapitre 2. Dans la suite de ce chapitre, dans la section 1.2, nous définirons la problématique de Routage de guide d'ondes **The Waveguide Routing Problem**

(WGRP).

1.2 La Problématique de routage de guides d'ondes

La problématique étudiée correspond à la recherche d'un ensemble de chemins disjoints de longueur totale minimale entre un ensemble de couples (source, destination). Un couple (source, destination) représente un guide d'ondes à router. Le routage doit respecter un certain nombre de contraintes, des contraintes métiers, telle la courbure, et des contraintes liées à la zone de routage, contournement des obstacles par exemple. La figure 1.2 illustre un exemple simplifié d'une telle zone comprenant 3 composants et 6 guides d'ondes, non routés sur la partie gauche et routés sur la partie droite de la figure. Comme illustré par la figure 1.1, le routage est effectué sur plusieurs niveaux. Généralement trois niveaux sont nécessaires.

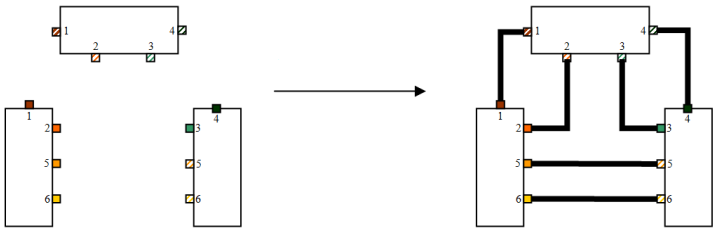


FIGURE 1.2 – Exemple de routage

Un guide d'ondes est un élément creux, rigide et de coupe rectangulaire comme illustré sur la figure ci-contre. Du fait de sa structure, il doit respecter une certaine courbure lors d'un changement de direction, ainsi qu'une distance minimale entre deux changements de direction successifs. Les guides d'ondes peuvent être de différentes épaisseurs et hauteurs. Pour donner un ordre d'idée, la hauteur (a) d'un guide d'ondes varie entre 10mm et 300mm, quant à la largeur (b) elle varie entre 10mm et 580mm.

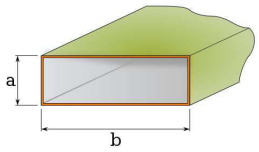


FIGURE 1.3 – Guide d'ondes

L'origine et la destination d'un guide d'ondes sont matérialisées par les connecteurs situés sur les équipements. Ces connecteurs peuvent être à différentes hauteurs et avoir des orientations différentes comme illustré par la figure 1.4. Selon l'orientation du connecteur l'épaisseur d'un guide d'ondes sur le plan varie (voir figure 1.4). De plus, les portions de départ et d'arrivée d'un guide d'ondes doivent suivre une direction spécifique (perpendiculaire au composant), et une distance minimale doit être respectée avant tout changement de direction. Ces portions de départ et d'arrivée permettent de garantir un libre accès aux vis de montage/démontage d'un guide d'ondes. Voir figures (1.5 - 1.6). Le respect de cette distance minimale peut être observée sur le figure 1.2 qui illustre un exemple de routage. De plus, lors du routage, il également nécessaire de respecter une distance minimale de $5mm$ entre les guides d'ondes.

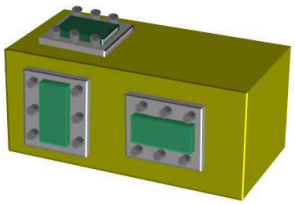


FIGURE 1.4 – Connecteurs

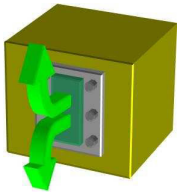


FIGURE 1.5 – Pas de distance minimale : Libre accès aux vis

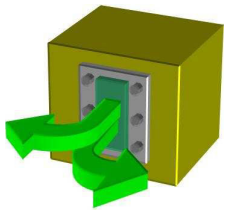


FIGURE 1.6 – Distance minimale nécessaire : libérer l'accès aux vis

CHAPITRE 2

ÉTAT DE L'ART

La problématique de **Pipe Routing** fait l'objet, depuis les années 80, de nombreuse recherches pour diverses applications industrielles : conception d'installation industrielle, conception de navire, l'aéronautique, l'électronique (VLSI design, pour Very Large Scale Integrated) et bien d'autres. Dans la suite du document, selon le domaine, un pipe (mot anglais signifiant tuyaux, canalisation,...) se réfère soit à une canalisation acheminant un liquide (eau, essence,...), soit à un câblage électrique, à une piste métallique connectant deux composants électroniques d'un circuit intégré (VLSI), ou alors à un guide d'ondes dans notre cas. Le **Pipe Routing** est une problématique de routage consistant à determiner un ensemble de routes pour un ensemble de pipes, où chaque pipe est défini par un ensemble de terminaux qui doivent être interconnectés en respectant un certain nombre de contraintes. Un terminal est une zone ponctuelle d'un équipement qui permet de le connecter à un ensemble d'équipements. Selon le domaine industriel et le type de pipe, l'ensemble des terminaux pour un pipe donné peut être soit composé d'une source et d'une destination ou d'une source et de plusieurs destinations. La problématique de routage est sujete à un ensemble de contraintes qui varient selon le domaine industriel. Ces contraintes ne se limitent pas à la minimisation de la longueur de la route attribuée à un pipe et à l'évitement d'obstacles. Généralement, différentes classes de contraintes doivent être respectées selon le domaine. Ci-dessous une liste non exhaustive des contraintes qui peuvent être rencontrées :

Contraintes physiques connecter les terminaux en évitant les obstacles (équipements et autres pipes),

Contraintes économiques minimiser la longueur des pipes et le nombre de changements de directions,

Contraintes de sécurité conserver une distance minimale par rapport à certain équipements,

Contraintes de production maximiser le nombre de canalisation par support (effet nappe),

Contraintes d'accommodation router les pipes le long des murs,

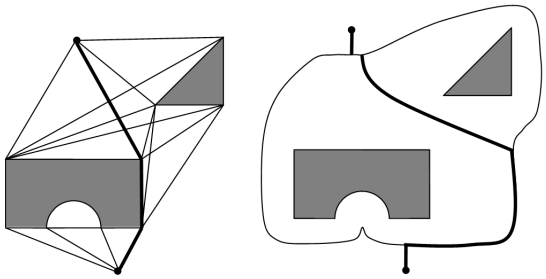
Contraintes de maintenabilité et d'opérabilité laisser libre accès aux elements permettant le montage/démontage des pipes (par exemple les vis) et aux vannes.

Dans ce chapitre, nous allons presenter les domaines d'application cités ci-dessus en proposant un état de l'art pour chaque domaine. Mais tout d'abord, nous allons nous pencher sur les aspects communs aux différents domaines d'application de la problématique de **Pipe Routing**, qui sont la discrétisation de l'espace de routage et le routage d'un pipe.

2.1 Discrétisation de l'espace de routage

Principalement deux méthode de discrétisation sont utilisées, l'approche **Skeleton** et la décomposition en cellules (en anglais, Cell Décomposition), le terme cellule ayant été introduit par Lee [101]. L'approche **Skeleton**,

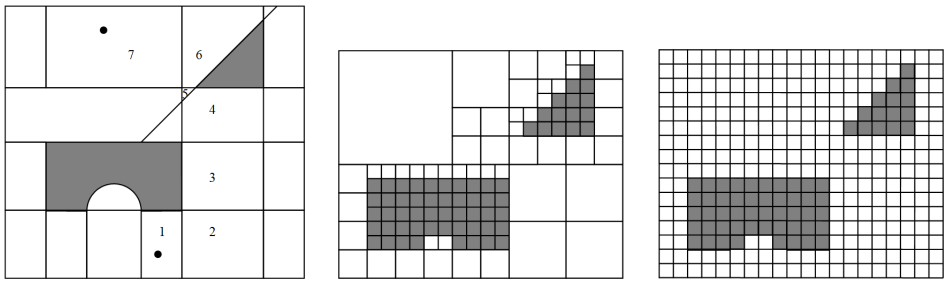
aussi appelée **Roadmap** ou **Highway** est basée sur le principe de la réduction de l'espace de routage continu en un graphe representant un ensemble de déplacements possibles dans cet espace. Le graphe de visibilité [88] et le diagramme de Voronoi [9, 88] sont les principaux **Squeleton**. Le graphe $G = (V, E)$ de visibilité est tel que l'ensemble V regroupe les sommets des obstacles et l'ensemble E est l'ensemble des arcs tel que $(i, j) \in E$ si le segment reliant $v_i \in V$ à $v_j \in V$ n'intersecte pas d'obstacle, figure 2.1(a). Le diagramme de Voronoi est un ensemble de points équidistants de deux obstacles ou plus, figure 2.1(b). Il existe d'autres approches **Skeleton**, tel que l'approche silhouette et l'approche **Subgoal** (littéralement, destination intermédiaire).



(a) (b)

FIGURE 2.1 – Graphe de visibilité (a), Diagramme de Voronoi (b)

La méthode de décomposition en cellules, qui est la plus utilisée, discrétise l'espace de routage en un ensemble de cellules homogènes ou hétérogènes. Dans la majorité des travaux, une cellule correspond à un rectangle dans un espace à deux dimensions ou à un parallélépipède rectangle dans un espace à trois dimensions. L'hétérogénéité (respectivement l'homogénéité) d'une décomposition signifie que les cellules n'ont pas les mêmes dimensions (respectivement ont les mêmes dimensions). Soit $G = (V, E)$ le graphe d'adjacence entre les cellules. V est l'ensemble des cellules et $(i, j) \in E$ si les cellules i et j sont adjacentes, c'est-à-dire qu'elles ont une frontière en commun. Ce graphe permet alors de relier la cellule contenant le point de départ à la cellule contenant le point d'arrivée par une succession de cellules adjacentes. Une décomposition peut être soit dépendante des obstacles ou indépendante. Dans le cas de la dépendance, la silhouette des obstacles et utilisée pour determiner la frontière des cellules et l'union des cellules est exactement égale à l'espace de routage libre (non occupé par un obstacle), figure 2.2(a). Il en résulte des cellules de formes diverses. L'avantage d'une telle décomposition réside dans le nombre restreint de cellules, mais ceci au dépend d'une complexité de décomposition élevée avec un calcul d'appartenance, d'interconnexion et d'adjacence des cellules très coûteux. En revanche, une méthode non dépendante aux obstacles permet de générer des cellules de forme géométrique simple et identique. L'intersection de chaque cellule avec les obstacles est vérifiée. Avec une telle méthode, la frontière des obstacles est plus approximative qu'avec la précédente. Cette approximation peut être réduite en augmentant le nombre de cellules. L'arbre quaternaire, figure 2.2(b) et la grille, figure 2.2(c) sont des exemples de décompositions en cellules non dépendantes aux obstacles, respectivement homogène et hétérogène en terme de taille des cellules.



(a) (b) (c)

FIGURE 2.2 – Décomposition en cellules

2.2 Routage de pipes

Dans la plupart des approches, déterminer une route pour un pipe revient à déterminer le plus court chemin minimisant le nombre de changements de directions et évitant les obstacles. De plus, lors du choix de la direction de déplacement, seules quatre directions sont disponibles, ce qui a pour effet de limiter le changement de direction à angle de 90°. Les approches font alors appel à des algorithmes de routage de type **Maze Routing** [101, 121, 103], dont le plus connu et le plus utilisé est sans nul doute l'algorithme de Lee [101], que ce soit sous sa forme originale ou l'une de ses nombreuses améliorations : Mikami et Tabuchi [113] proposent l'algorithme appelé line-search pour le routage de circuits imprimés dont l'une des couches contient des lignes horizontales, et l'autre des verticales, les deux couches étant reliées par des trous (vias). Cet algorithme est une alternative moins coûteuse en terme de temps d'exécution et d'occupation mémoire. Au lieu de travailler sur une grille de cellules, l'algorithme stocke des lignes déterminées par seulement trois coordonnées. La recherche s'effectue alternativement depuis la source et la destination, par l'extension de lignes dans les quatre directions. Les lignes originaires de la source sont comparées à celles originaires de la destination afin de détecter la fin de l'algorithme, lorsque deux lignes se croisent ; Soukup [151] exploite une recherche en profondeur d'abord. L'expansion se poursuit dans le sens de la destination, tant qu'aucun obstacle n'est rencontré. Lorsque cette expansion de type line-search n'est plus possible, l'algorithme de Lee prend le relais, jusqu'à ce que l'obstacle ait été contourné ; Alors que la plupart des algorithmes de Maze Routing effectuent une recherche en largeur d'abord, l'algorithme A^* est un algorithme bestFirst qui effectue une recherche en profondeur d'abord avec un mécanisme d'expansion basé sur la selection de prochaine cellule à étendre selon une évaluation de la distance potentielle au but. Cette évaluation est effectuée grâce à une heuristique. Cet algorithme a été proposé par Hart et al [125, 126]. La sélection de la prochaine cellule à étendre est effectuée selon un coût F qui est égal à la somme de la distance déjà parcourue G et l'évaluation H de la distance qui reste à parcourir pour atteindre le terminal destination. La cellule de plus faible coût est étendue. Le lecteur pourra se référer à [154] pour prendre connaissance d'autres variantes de l'algorithme de Lee.

Ces différents algorithmes peuvent être vue comme des implementations de l'algorithme de plus court chemin de Dijkstra [55] qui permet, s'il existe, de trouver le plus court chemin reliant une source à une destination.

Certaines approches font appel à des techniques de programmation mathématique, que ce soit des méthodes déterministes, comme la programmation linéaire ou à des méthodes non déterministes comme l'algorithme génétique ou le recuit simulé pour ne citer que les principaux. Cette utilisation est motivée par le besoin de prendre en compte un nombre de contraintes plus important ou de determiner un ensemble de routes possibles pour un même pipe. Plus de details sur l'utilisation des ces méthode seront fournis dans la suite du chapitre.

À present nous allons presenter un certain nombre de travaux effectués dans les différents domaines industriels cités précédemment : conception d'installations industrielles, conception de navire, l'aéronautique et l'électronique (VLSI design, pour Very Large Scale Integrated).

2.3 Conception d'installations industrielles

La conception en ingénierie d'installations industrielles chimique a fait l'objet d'un certain nombre de travaux, dont [139, 78, 77, 76] portant sur l'optimisation de l'aménagement des équipements industriels en respectant des contraintes de coût, de sécurité, d'opérabilité, de maintenabilité, etc. La problématique de **Pipe Routing** intervient dans le processus d'optimisation comme un indicateur de qualité du positionnement des équipements, c'est-à-dire que pour un positionnement donné, les routes des pipes reliant les composants sont déterminées afin d'évaluer la réalisabilité et l'optimalité du positionnement d'un point de vue économique. La figure 2.3 illustre un routage de pipe dans une installation industrielle.

FIGURE 2.3 – Routage de pipe dans une installation industrielle

Dans [139] les auteurs présentent une approche séquentielle de routage de pipes exploitant une base de connaissance **Knowledge Engineering**. Cette base de connaissance permet de définir l'ordre de routage des différents pipes, par exemple en donnant la priorité aux pipes de plus grand diamètre, et de définir le coût attribué à chaque route selon différents critères. Cette approche est basée sur une décomposition de l'espace de routage en cellules (grille de cellules) et l'utilisation d'une variante de l'algorithme de Lee [101]. Cette variante retourne un ensemble de routes réalisables grâce à un bruitage sur les coûts et les directions d'expansion lors du routage. Avant le routage, les positions des supports de pipe sont prédéfinis afin de favoriser un effet nappe, comme illustré par la figure 2.3. Un support est une structure permettant de router un ensemble de pipes côte-à-côte dans une même direction.

Dans [78] et [77] les auteurs présentent une approche de routage de pipes sur une grille rectangulaire basée sur deux algorithmes : l'algorithme de Lee [101] et un algorithme de type line-search. Avant le routage, chaque cellule de la grille est définie comme étant une cellule interdite, préférentielle (exemple, contenant un support de pipe) ou facultative. La largeur d'une cellule est égale au diamètre du pipe. Les pipes précédemment routés n'interviennent pas lors du routage mais sont pris en compte lors de l'évaluation du coût global de la solution d'aménagement des équipements.

Dans [76], les auteurs proposent différentes techniques permettant un aménagement automatique des équipements, dont un module de routage basé sur un graphe $G = (V, E)$, avec $v \in V$ un sommet représentant une jonction possible entre deux portions de route d'un même pipe ou entre deux supports, et l'arc $(i, j) \in A$ représentant un support ou une portion de route. Un sommet est défini par une position (x, y, z) et une direction. L'existence de chaque sommet et chaque arc est défini par un certain nombre de règles : par exemple, un sommet, dit primaire est associé à chaque terminal et pour chaque paire de sommets primaires dont les directions (direction défini par la droite passant par un terminal et perpendiculaire à l'équipement) sont perpendiculaires deux sommets dit secondaires sont positionnés pour completer la route rectangulaire reliant le couple de sommets primaires ; un arc existe entre un sommet et le sommet qui lui est le plus proche. Pour determiner une route de plus petite distance pour un pipe, il est alors possible d'appliquer, soit un Programme Linéaire en Nombre Entier, soit l'algorithme de Dijkstra mais avec l'ajout de sommets et d'arcs supplémentaires.

2.4 Conception de navire

Le routage de pipe dans un navire est une phase très complexe qui n'est pas complètement automatisée à ce jour. Cette phase nécessite l'intervention d'ingénieurs spécialisés, qui grâce à leurs experiences arrivent à mener à bien cette tâche mais au prix d'un investissement en temps considérable. Un certain nombre d'approches de routage ont été basées sur l'acquisition de connaissances métiers et leur exploitation pour automatiser la phase de routage en minimisant l'intervention humaine. Dans [143] les auteurs proposent une approche basée sur un système expert permettant de reproduire les mécanismes cognitifs d'un expert (d'un ingénieur). La base de connaissance est constituée en identifiant les objets intervenant dans le routage, les relations entre ces objets, et en ajoutant des méthodes à chaque objet. Les routes associées aux pipes, les elements (valve, support, terminal,...) d'un pipe, les équipements et les espaces libres sont considérés comme des objets. Une fois la base de connaissance définie, le routage des pipes est effectué par ordre de priorité et par agencement des différents objets de la base de connaissance grâce aux méthodes associées. L'utilisateur peut alors visualiser le résultat, modifier la base de connaissance si le résultat ne lui convenait pas et relancer l'itération. Dans l'article, les bases de connaissances constituées regroupent 167 règles et 106 méthodes.

Dans [89, 90, 91], l'auteur propose une approche permettant de fournir un ensemble de routes candidates pour un pipe afin de permettre à l'utilisateur de sélectionner la meilleure route d'après sa propre expérience. L'approche est basée sur un algorithme génétique. L'espace de routage est décomposé en cellules rectangulaires de forme homogène. Un chromosome, caractérisant une route candidate, est défini comme un vecteur d'entiers compris dans l'intervalle [0,4], l'etat 0 indiquant l'arrivé à la destination et les états 1,2,3 et 4 indiquant les directions de déplacement, respectivement droite, haut, gauche et bas. Afin d'obtenir une population initiale de routes diversifiées, un certain nombre de mécanismes sont mis en place : un principe de directions préférentielles basé sur l'attribution à des zones préalablement définies de vecteurs de priorité de choix de la direction à suivre ; des points de passage obligé positionnés afin de dévier une route de la trajectoire la plus directe pour atteindre le terminal destination ; l'utilisation du concept d'énergie potentielle localisée qui attribue une valeur d'énergie à chaque cellule permettant de guider le comportement d'un pipe. Par exemple, si une contrainte implique que la route associée à un pipe doit longer les obstacles, une faible valeur d'énergie sera attribuée aux cellules proches de l'obstacle alors que celles contenant un obstacle se verront attribuer une forte valeur.

Les routes sont ensuite générées par sélection aléatoire d'une direction selon le vecteur de priorité de choix. Une fonction d'évaluation, integrant les contraintes permet de mesurer la qualité d'une solution. En fin d'execution, l'approche fournit à l'utilisateur une route optimisée selon les critères saisis mais aussi les routes générées tout au long du processus. L'utilisateur sélectionne alors la meilleure route d'après sa propre expérience. La figure 2.4 illustre le résultat de l'exécution de l'approche pour un pipe.

Certaines approches tentent d'automatiser le processus de routage de pipes. Dans [131], les auteurs proposent une approche permettant un routage automatique de pipes dans la salle de machine d'un navire et prenant en compte la plupart des contraintes d'accommodation. Dans un premier temps, des routes candidates sont fournies par une méthode de génération de cellules, basées sur des modèles de routage : modèle basique (figure 2.5), variantes de modèles basique (figure 2.5), modèles de contournement d'obstacles (figure 2.6) et des modèles de connexion entre cellules adjacentes (figure 2.7). Les routes candidates pour chaque pipe sont déterminées durant le même appel de la méthode de génération de cellules. Cependant, la priorité est donnée aux pipes de plus grand diamètre qui doivent adopter un routage utilisant les modèles de base. Un parcours d'arbre est ensuite effectué pour sélectionner le meilleur ensemble de routes vérifiant l'ensemble des contraintes. Le respect des contraintes non géométriques est vérifié par une fonction d'évaluation integrant ces contraintes. Les contraintes géométriques sont prises en compte lors de la génération de cellules.

La méthode de génération de cellule est une alternative à la méthode de décomposition en cellules. Les cellules sont générées au fil de la résolution. La figure 2.8(a) l'illustre pour trois pipes de 1, 2 et 4 inches de diamètre avec des directions de sortie ou d'entrée pour chaque terminal (flèches et cercles noir sur la figure, respectivement).

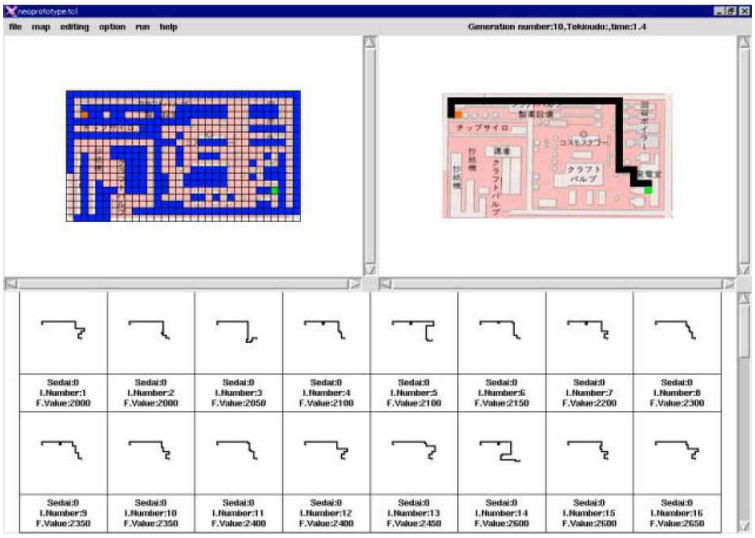


FIGURE 2.4 – Exemple de routage avec une route optimale et les routes candidates générées

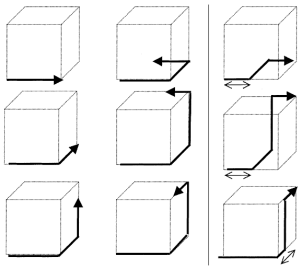


FIGURE 2.5 – Modèles basiques de traversée de cellule et variantes des modèles basiques

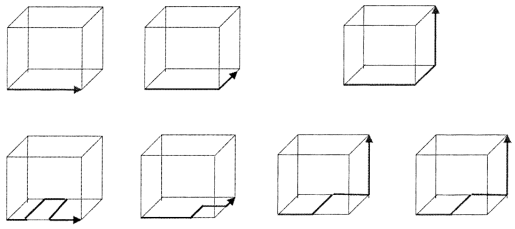


FIGURE 2.6 – Modèles de contournement d'obstacles

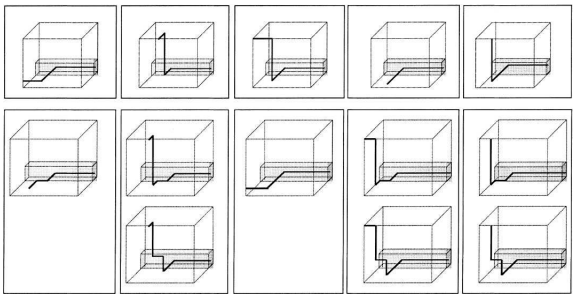


FIGURE 2.7 – Modèles de connexions

Dans un premier temps, une cellule de forme rectangulaire (aux dimensions du pipe) est générée pour chaque terminal, avec une direction spécifique, figure 2.8(b). Après avoir groupées les cellules 10 et 20, la cellule 12 est générée pour atteindre la destination du pipe de 4 inch, qui a la priorité du fait de son diamètre, figure 2.8(c). La cellule 22 est ensuite générée pour le pipe de 2 inch, formant une forme basique en **L** avec la cellule 12, figure 2.8(d). Les cellules 32 et 33 sont générées pour le pipe de 1 inch selon le même principe, figure 2.8(e). À présent, un modèle de route basique a été généré pour chaque pipe. L'algorithme va maintenant tenter de générer des variantes afin de diversifier l'allure des routes des pipes de 1 et 2 inch. Pour ce faire, les cellules 41 et 42 sont générées pour accéder à la cellule 33, et la cellule 43 pour accéder à la cellule 32 selon le modèle de connexion, figure 2.8(f). Cela permet au pipe de 2 inch d'avoir une variante. De la même manière, les cellules 41, 42 et 43 permettent au pipe de 1 inch d'avoir une variante, figure 2.8(e). Au final, les pipes de 1 et 2 inch ont deux

routes candidates. En revanche, le pipe de 4 inch n'a qu'une seule route, ce qui s'explique par la priorité qui lui a été attribuée. Il reste alors à effectuer un parcours d'arbre afin d'obtenir une solution dont la qualité est mesurée grâce à une fonction d'évaluation integrant les contraintes considérées.

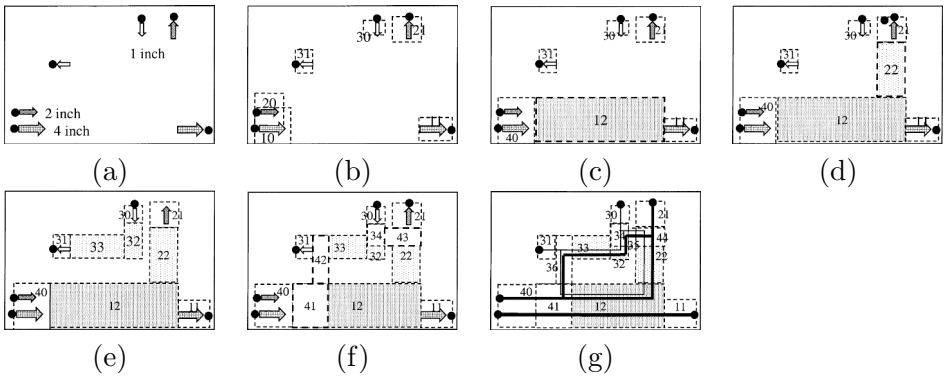


FIGURE 2.8 – Méthode de génération de cellules

Dans [6, 7] les auteurs proposent une automatisation du routage de pipe combinant un algorithme déterministe, Dijkstra [55] et un algorithme non déterministe, l'Optimisation par Essaims Particulaires (PSO en anglais) introduit par Kennedy et Eberhart [56] et Eberhart et Kennedy [96]. Les pipes routés dans cette approche nécessitent le positionnement d'éléments que l'ont nomme branchements et qui permettent de relier un terminal source à plusieurs terminaux destination.

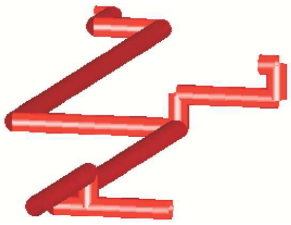
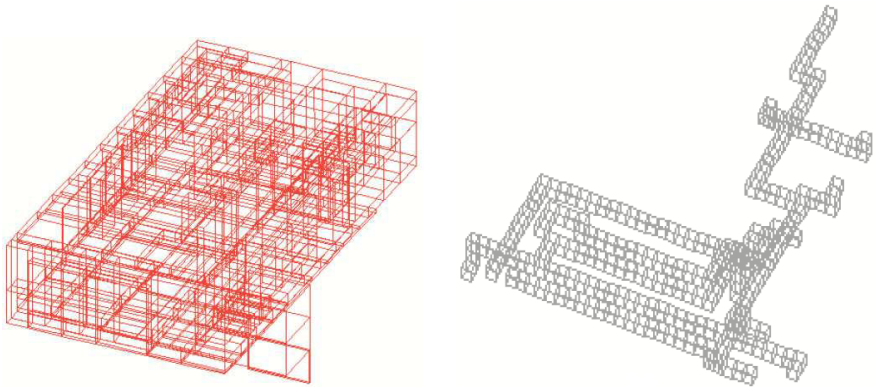


FIGURE 2.9 – Exemples de branchements

L'algorithme de plus court chemin Dijkstra est exécuté dans le graphe $G = (V, E)$ d'adjacence des cellules issues de la decomposition en cellules de l'espace de routage. Cette décomposition a été effectuée en deux étapes. Dans un premier temps, la zone de routage est décomposée en un ensemble de cellules rectangulaires de grande taille ($\approx 1m$), figure 2.10(a). L'algorithme de Dijkstra est exécuté pour tous les pipes successivement sans considerer, à une itération donnée les pipes routés aux itérations précédentes. Successivement, pour chaque pipe et à l'intérieur de chaque cellule utilisée, une nouvelle cellule est crée de taille égale au diamètre du pipe, après quoi l'algorithme de Dijkstra est exécuté une seconde fois en éliminant les cellules utilisées du graphe $G = (V, E)$ d'adjacence, figure 2.10(b). Cette opération est répétée jusqu'à ce que tous les pipes soit routés. Pour relier un terminal source à un ensemble de terminaux destinations, les auteurs proposent une approche basée sur l'algorithme de Dijkstra. Pour un tel pipe, deux terminaux sont sélectionnés pour determiner le plus court chemin les reliant. Ensuite, itérativement, un terminal non connecté est sélectionné. L'algorithme de Dijkstra est exécuté pour déterminer les plus courts chemins reliant le terminal sélectionné aux cellules utilisées par le plus court chemin initialement calculé. Le chemin ayant la plus petite distance détermine la position du branchement.

L'originalité de cette approche tient au fait que l'ordre de routage des pipes, l'ordre de positionnement des branchements, les routes à conserver ou à remettre en question sont déterminés par une méthode d'optimisation : **l'Optimisation par Essaims Particulaires Discrete** (OEPD) [97]. La méthode utilisée est inspirée des



(a) (b)

FIGURE 2.10 – Les deux étapes de décomposition en cellules

travaux de Clerc [36] qui applique l'OEPD à la problématique du voyageur de commerce. Cette méthode d'optimisation se base sur la collaboration des individus entre eux, un individu correspondant à un pipe dans [6, 7]. Elle a d'ailleurs des similarités avec les algorithmes de colonies de fourmis qui s'appuient eux aussi sur le concept d'auto-organisation. Cette idée veut qu'un groupe d'individus peu intelligents peut posséder une organisation globale complexe. Ainsi, grâce à des règles de déplacement très simples (dans l'espace des solutions), les particules peuvent converger progressivement vers un minimum local.

2.5 Aéronautique

La problématique de **Pipe Routing** dans l'aéronautique concerne le routage de câblages électriques et de canalisations. Comme pour les domaines précédemment cités, le routage est une partie intégrante de la phase de conception d'un avion. Il doit respecter de nombreuse contraintes techniques et de sécurité pour obtenir une certification. Les circuits électriques et hydrauliques d'un avion sont composés de centaines de pipes (selon la taille de l'avion : A380), dont le routage est généralement effectué manuellement par des ingénieurs, qui se servent de leurs connaissances et experiences du metier pour déterminer le routage idéal. Cette tâche nécessite un temps considérable.

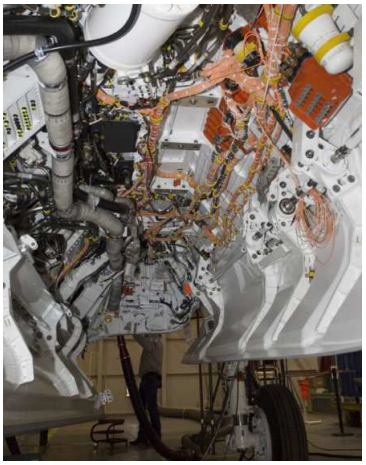


FIGURE 2.11 – Exemple de routage réel d'une soute d'armement d'un F-35 Joint Strick Fighter

Dans [47, 48, 49, 50], les auteurs proposent un système de routage intelligent pour automatiser le design des faisceaux de câblages électriques et des pipes dans un avion. Le système emploie une méthode d'ingénierie à

base de connaissance **Knowledge Based Engineering** (KBE) qui consiste à identifier et implementer les règles et les connaissances concernant les processus d'ingénierie du routage manuel. Ce système se base sur une discrétisation de l'espace de routage continu en trois dimensions en un ensemble de cellules homogènes. Une interface permet à un utilisateur de transmettre l'espace de routage (modèle en trois dimensions issu d'une conception assistée par ordinateur : CAO) et de définir le type de pipe à router, soit un cable ou une canalisation hydraulique par exemple. À la fin de l'execution, une visualisation lui est transmise afin qu'il puisse valider le résultat ou redéfinir le routage. Le routage est effectué séquentiellement, un pipe à la fois mais en considérant les pipes précédemment routés, par exemple en privilégiant un effet nappe entre un ensemble de pipes passant par la même zone et se dirigeant vers une même direction. Dans cette approche, les règles issues de la base de connaissances qui sont prises en compte concernent la courbure des changements de direction et l'espacement entre un pipe et les équipements présents sur l'espace de routage (un composant, la structure du fuselage ou un pipe précédemment routé). La règle d'espacement peut être soit une règle de répulsion soit une règle d'attraction.

L'algorithme de routage utilisé est l'algorithme de recherche de plus court chemin **A*** qui repose sur une fonction d'évaluation de coût $F(n) = G(n)+H(n)$. Les auteurs intègrent les contraintes citées précédemment en ajoutant des fonctions de coût supplémentaires à la fonction d'évaluation qui s'exprime alors $F(n) = G(n) + H(n) + \sum I(n)$, avec $I(n)$ une fonction de coût associée à une contrainte.

La figure 2.12 illustre le résultat de routage d'une soute d'armement d'un F-35 Joint Strick Fighter. L'exemple de la soute d'armement est considéré comme étant complexe et est utilisé par les auteurs comme base de tests.

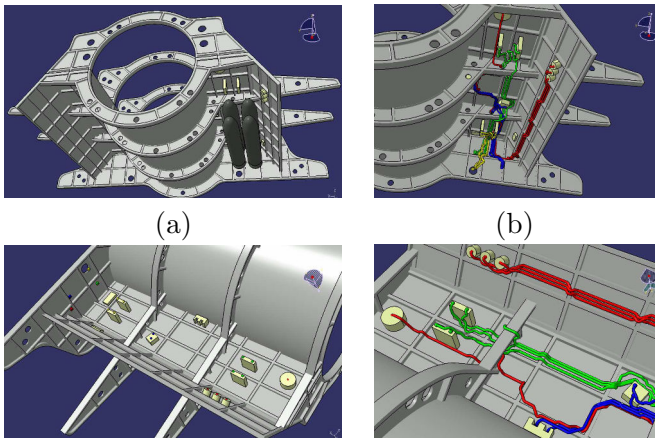


FIGURE 2.12 – Résultats de routage d'une soute d'armement d'un F-35 Joint Strick Fighter

2.6 Conception de circuits intégrés : Very Large Scale Integrated

La conception de circuit de grande dimension (VLSI design, pour Very Large Scale Integrated) est un processus complexe divisé traditionnellement en une succession de problèmes théoriques et pratiques hiérarchisés. Cette complexité se retrouve dans l'étape de routage de l'ensemble des réseaux entre les différents composants, précédée d'une étape de positionnement des composants sur un support.

Un circuit est défini par un ensemble de composants et un ensemble de réseaux. Les composants sont considérés comme des boîtes noires pouvant schématiser les composants de base (transistors, condensateurs...) ou des macro-composants (processeurs, sous-circuits,...). À chaque composant correspond un ensemble de points terminaux (pins, en anglais). Un point terminal est une zone ponctuelle du composant qui le connecte au reste

du circuit. Un réseau est un ensemble de points terminaux devant être électriquement connectés par des pistes conductrices (wires, en anglais), le nombre de points terminaux étant le degré du réseau. On appelle liste des réseaux (netlist, en anglais) l'ensemble de tous les réseaux d'un circuit. Par extension, on appelle également réseau la collection des pistes qui connectent électriquement les points terminaux d'un même réseau. Dans ce cas, un réseau est donc un ensemble de lignes droites reliant plusieurs composants. La figure 2.13 illustre un circuit.

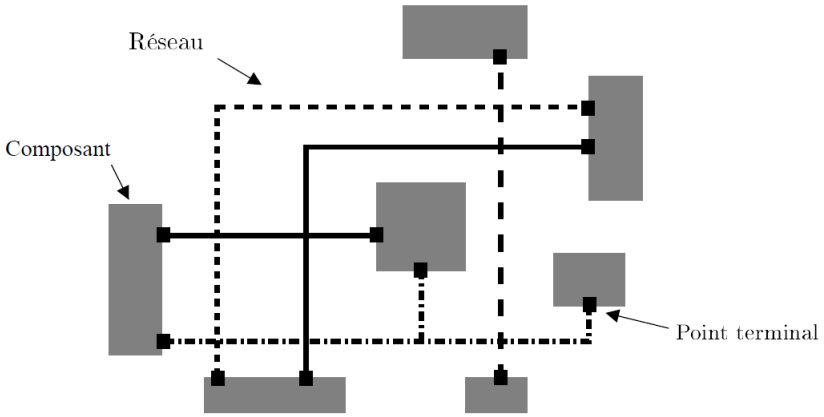


FIGURE 2.13 – Un exemple de circuit

Étant donné une liste de points terminaux T et une liste de réseaux N, on appelle étape de routage le problème consistant à déterminer l'ensemble des pistes reliant tous les points terminaux $t_i \in T$ d'un même réseau $n_j \in N$ sans que deux pistes de réseaux différents ne soient connectées. Le routage est effectué sur plusieurs niveaux, avec une direction spécifique des pistes par niveau, typiquement une direction horizontale ou verticale, avec une alternance des directions entre les niveaux (HVHVH...). Le changement de niveau est possible grâce au positionnement d'un via (un trou) qui est un point de contact entre deux niveaux adjacents. La figure 2.14(a) illustre un schéma logique d'un circuit comprenant trois réseaux qui nécessite deux couches. La figure 2.14(b) illustre une vue en trois dimensions d'une solution sur deux niveaux, avec l'utilisation d'un via pour relier les deux sous réseaux affectés à deux couches différentes.

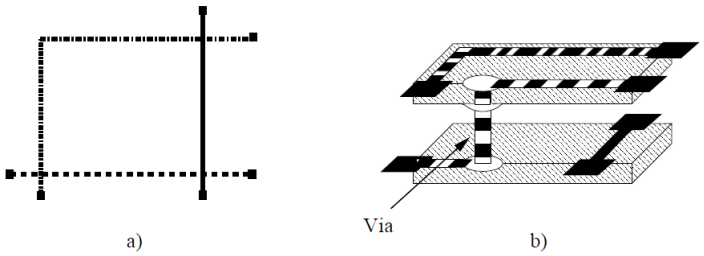


FIGURE 2.14 – Un exemple de circuit sur deux couches avec vias

Afin de réduire la complexité du routage, le circuit VLSI est partitionné en plusieurs cellules, appelée *G-Cell* (généralement en rectangles) qui peuvent être traitées indépendamment. Ainsi, le routage de chaque zone peut être effectué par des heuristiques efficaces, voire des des méthodes exactes. Un tel routage est appelé routage détaillé (routage local). Néanmoins, un routage détaillé d'une cellule donnée doit être compatible avec les routages détaillés des cellules adjacentes. Cette compatibilité est garantie par une étape appelée routage global, antérieure aux routages détaillés. Les travaux de Korte, Prömel et Steger [98] et de Grötschel, Martin et Weismantel [110, 75] donnent un aperçu des rapports entre le routage global et le routage local. La figure 2.15 illustre un circuit partitionné en cellules rectangulaires.

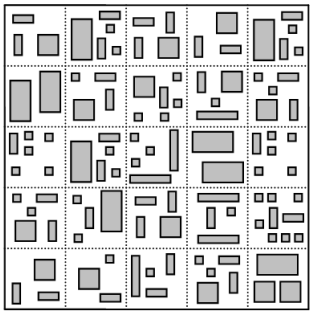


FIGURE 2.15 – Circuit partitionné en une succession de rectangles : frontières en pointées

2.6.1 Routage global

La problématique du routage global est habituellement modélisée en tant que problématique de la théorie des graphes. Un circuit VLSI est formalisé sous forme d'un graphe $G(V, E)$ d'adjacence des cellules issues de la decomposition du circuit. La décomposition est soit hétérogène, figure 2.16(a), soit homogène, figure 2.16(b). Ces deux décompositions représentent le graphe associé au circuit de la figure 2.15. Chaque sommet $v_i \in V$ représente une cellule rectangulaire de la zone de routage et chaque arc $e_{i,j} \in E$ représente une frontière entre les cellules i et j, avec une capacité correspondant au nombre de réseaux $n_i \in N$ pouvant être acheminés entre les deux cellules, et un coût l_{ij} d'utilisation égale à la longueur de l'arc, matérialisé par exemple par la distance séparant le centre de gravité des cellules i et j. Le routage global nécessite un ensemble de réseaux $N = \{n_1, n_2, \cdots, n_k\}$ qui doivent être routés dans le graphe $G = (V, E)$. Un reseau $n_i \in N, 1 \leq i \leq k$ est un ensemble de terminaux $\{v_{i0}, v_{i1}, v_{i2}, \cdots\} \subseteq V$, où v_{i0} est le terminal source et les autres terminaux sont les destinations. Dans le routage global, il est admis que tout terminal est situé au centre de la cellule le contenant. Le difficulté du routage pour un réseau $n_i \in N$ est d'identifier un ensemble de sommets de Steiner, noté $V_{i,Steiner} \subset V$ et un ensemble d'arcs $E_i = \{e_{i1}, e_{i2}, \cdots\} \subset E$ formant un arbre couvrant rectilinéaire de poids minimum $T_i = (V_i, E_i)$, où $V_i = \{n_i\} \cup V_{i,Steiner}$. Dans la suite du document, une route associée à un réseau sera appelée arbre si le réseau est de degré supérieur à deux.

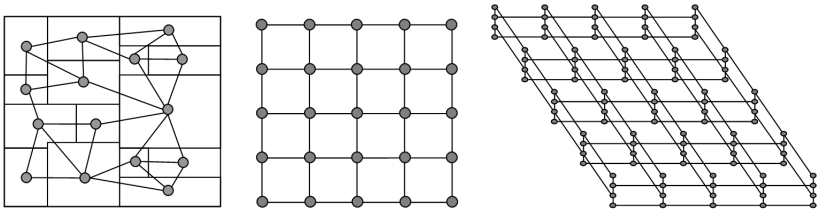
L'objectif du routage global est de définir grossièrement le comportement des réseaux autour des composants, sous contrainte de la capacité des arcs et en minimisant la longueur des réseaux. De plus, le routage doit être effectué selon deux autres critères d'optimisation : minimiser le nombre de vias et minimiser les situations de congestion (d'encombrement) afin de limiter les situations de blocage. La congestion λ est définie par exemple de la façon suivante : Soit un arc $e \in E$, la congestion associée est défini par le ratio $f(e) \backslash c(e)$, $f(e)$ étant le flot porté par l'arc et c(e) sa capacité.

Un via permet à un reseau de changer de niveau dans un circuit. Soit l'affectation des réseaux aux différents niveaux est effectuée à posteriori, après le routage, on parle alors de routage en deux dimensions comme illustré par les exemples (a)-(b) de la figure 2.16. Soit elle l'est durant le routage grâce à la prise en compte des vias directement dans le graphe $G(V, E)$, on parle alors de routage en trois dimensions, comme illustré par la figure 2.16(c) : Le graphe est composé de quatre niveaux (HVHV), avec un certain nombre d'arcs (de vias) reliant les quatre niveaux et qui permettent d'en changer. Une capacité, représentant le nombre maximum de vias par cellule est associée à chaque arc.

Dans la littérature, le routage global est traité par différentes méthodes qui mettent en oeuvre un certain nombre de techniques de base pour résoudre les sous-problèmes du routage global : Maze Routing, Pattern Routing, Arbre de Steiner, Programme Linéaire En Nombre Entier et Modèle de Flot.

Maze Routing voir section 2.2.

Pattern Routing Une alternative aux algorithmes de la classe Maze Routing et line-search, utilisés habi-



(a) (b) (c)

FIGURE 2.16 – Formalisation sous Forme de Graphe

tuellement pour traiter la problématique de routage global, est proposée par Kastner et al [135], le « Pattern Routing ». L'approche proposée utilise trois modèles de routage de base, le modèle en 'I' , en 'L' et le modèle en 'Z' pour déterminer une route reliant deux terminaux. Ozdale et Wong [124] exploite cette méthode et ajoute un nouveau modèle, le modèle en 'U' qui permet de contourner les zones congestionnées durant une procédure de « Rip-Up and Reroute » (littéralement supprimer et dérouter), expliquée ultérieurement dans ce chapitre. Le « Pattern Routing » a l'avantage d'etre moins coûteux que les autres algorithmes et de nécessiter un nombre fixe de vias, mais il ne permet pas toujours d'obtenir une route optimale quand elle existe, car il ne peut explorer toutes les solutions. Pour rappel, un via est nécessaire lors d'un changement de direction, par exemple pour le modèle de routage en 'L'.

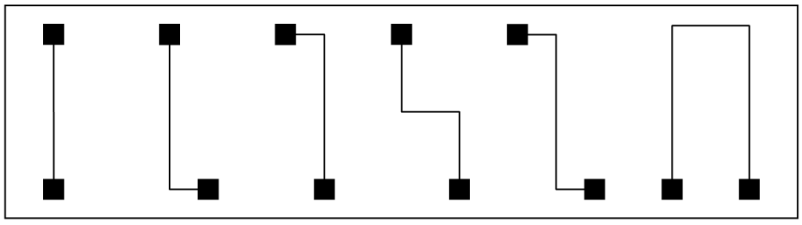


FIGURE 2.17 – Modèles de routage : I, L, Z, U

Arbre de Steiner Les algorithmes de la classe Maze Routing ne permettent de connecter que deux terminaux d'un même reseau de manière optimale. En pratique, un réseau est composé d'un terminal source et de plusieurs terminaux destinations qu'il faut atteindre. Une solution serait de déterminer l'arbre couvrant de poids minimum sur les terminaux et de router chaque paire de terminaux correspondant au extrémités d'un arc de l'arbre couvrant. La figure 2.18 (a) illustre la solution obtenue pour un reseau de trois terminaux. Dans [6, 7] les auteurs proposent une solution basée sur l'algorithme de Dijkstra, dont une explication est donnée dans la section 2.4.

Il est aisé de constater que la solution obtenue n'est pas optimale. En introduisant un sommet secondaire et en déterminant l'arbre couvrant de poids minimum, on obtient le résultat de la figure 2.18 (b), qui est optimal. Le sommet ajouté est un sommet de Steiner.

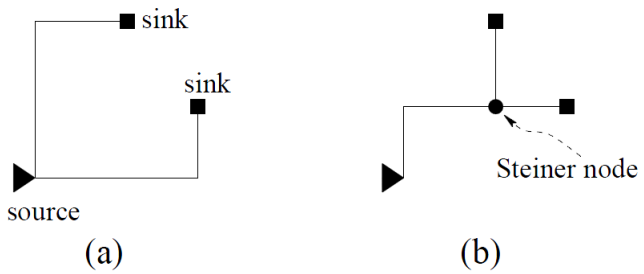


FIGURE 2.18 – Une solution par arbre couvrant (a) et par arbre de Steiner (b) pour un reseau de trois terminaux

Dans la majorité des problématiques VLSI, les réseaux sont constitués de pistes horizontales et verticales.

De ce fait, seuls les arbres de Steiner rectilinéaires sont considérés (Rectilinear Steiner Tree : RST), plus précisément les arbres de Steiner rectilinéaires de poids minimum (Rectilinear Steiner Minimal Tree : RSMT). L'arbre de Steiner rectilinéaire de poids minimum est le plus souvent utilisé afin de décomposer les réseaux en un ensemble de réseaux de degré égal à deux qui seront routés avec un algorithme de la classe Maze Routing.

Au vue de la complexité de la construction d'un arbre de Steiner rectilinéaire de poids minimum, problème NP-Difficile, un certain nombre d'algorithmes d'approximation ont été développés [27, 37, 95, 35, 34], dont l'approche FLUTE : Fast Lookup Table Based WireLenght Estimation Technique [35, 34] qui est à ce jour la méthode la plus rapide pour construire un arbre de Steiner rectilinéaire de poids minimum. Dans FLUTE, l'ensemble des réseaux de degré $n < 9$ sont partionnés en $n!$ catégories selon la position relative des réseaux. Pour chaque catégorie, un ensemble de vecteurs, nommées Potentially Optimal Wirelenght Vector (POWV) sont stockés dans un tableau nommé « Lookup Table ». Un vecteur POWV représente une combinaison linéaire de distances entre les lignes d'une grille de Hanan [81]. Soit un réseau, une grille de Hanan est construite de telle sorte que la position de chaque terminal représente un point d'intersection entre une ligne verticale et une ligne horizontale sur la grille. La méthode repose sur la fait que tout arbre de Steiner, dans une grille de Hanan peut être décomposé en un ensemble d'arcs (verticaux et/ou horizontaux) de la grille de Hanan. Ainsi, la longueur de tout arbre de Steiner peut toujours être définie par une combinaison linéaire de longueurs d'arcs, sous contrainte que les coefficients soient des entiers positifs. Soit un réseaux de degré n, x_i est la coordonnée sur l'axe des abscisses de la i-ème ligne verticale de la grille de Hanan de sorte que $x_1 \leq x_2 \leq \cdots \leq x_i$. De même, y_i est la coordonnée sur l'axe des ordonnés de la i-ème ligne horizontale de la grille de Hanan de sorte que $y_1 \leq y_2 \leq \cdots \leq y_i$. Ces notations sont illustrées par la figure 2.19.

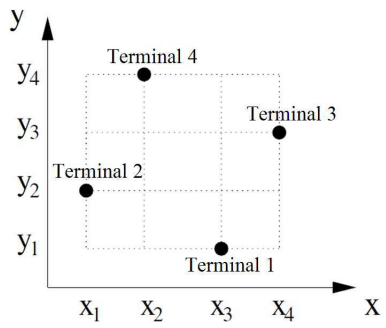


FIGURE 2.19 – Illustration d'un réseaux de 4 terminaux

Soit un arc $e_i \in E$ horizontal (respectivement vertical), sa longueur, dans une grille de Hanan est égale à $h_i = x_{i+1} - x_i$ (respectivement, $v_i = y_{i+1} - y_i$) pour $1 \leq i \leq n-1$. La figure 2.20 illustre ces longueurs. Une combinaison linéaire de longueurs d'arc peut alors être définie de la façon suivante : soit l'exemple illustré par la figure 2.21, les combinaisons linéaires des trois arbres de Steiner de la figure 2.21(a)-(c), correspondant aux trois réseaux de la figure 2.19 s'écrivent respectivement $h1 + 2h2 + h3 + v1 + v2 + 2v3$, $h1+h2+h3+v1+2v2+3v3$, et $h1+2h2+h3+v1+v2+v3$. Pour plus de simplicité, ces combinaisons linéaires sont stockées sous forme de vecteurs de coefficients, appelé vecteurs de longueurs. Ainsi, pour les trois arbres de Steiner 2.21(a)-(c), les vecteurs de longueur correspondant sont $(1, 2, 1, 1, 1, 2)$, $(1, 1, 1, 1, 2, 3)$, et $(1, 2, 1, 1, 1, 1)$ respectivement.

L'approche FLUTE est basée sur le pré-calcul d'une table contenant les configurations de longueur pour un réseau donnée de degré inférieur à 9. Ces configurations seront utilisées pour determiner l'arbre de Steiner rectilinéaire de poids minimum. Une méthode de décomposition de réseaux est aussi proposée afin de permettre l'utilisation de la méthode pour des réseaux de degré supérieur à 9.

L'approche FLUTE est utilisée par exemple par [30, 33, 32, 128, 127, 166, 164, 31, 116] pour générer un arbre de Steiner rectilinéaire de poids minimum permettant de décomposer les réseaux en un ensemble de

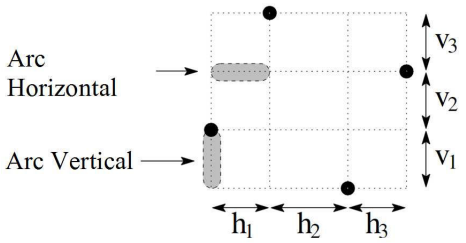


FIGURE 2.20 – Illustration de la longueur des arcs verticaux et horizontaux

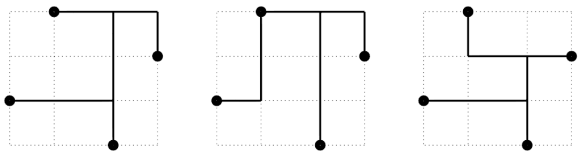


FIGURE 2.21 – Illustration des trois possibilités de routage pour le réseau de la figure 2.19

réseaux de degré égale à deux.

Programme Linéaire en Nombre Entier Le routage global peut être formulé sous la forme d'un PLNE [106]. Soit un ensemble $\tau_i = \{T_{i1}, T_{i2}, \cdots\}$ d'arbres candidats pour le réseau n_i, avec une variable de décision $x_{ij} \in \{0, 1\}$ associée au reseau i et qui indique si l'arbre T_{ij} est sélectionné pour le réseau n_i. Le problème de routage global est alors formulé de la maniére suivante :

$$\text{minimiser } \hat{\lambda} \tag{2.1}$$

sujet à

$$\sum_{T_{ij} \in \tau_i} x_{ij} = 1, \forall n_i \in N \tag{2.2}$$

$$\sum_{(i,j), e_{ij} \in T_{ij}} x_{ij} \leq \hat{\lambda} c(e), \forall e \in E \tag{2.3}$$

$$x_{ij} \in \{0, 1\}, \forall n_i \in N, \forall T_{ij} \in \tau_i \tag{2.4}$$

Les contrainte (2.2) et (2.4) assurent qu'un seul arbre sera sélectionné pour chaque réseau $n_i \in N$. La contrainte (2.3) et la fonction objective (2.1) garantissent la minimisation de la congestion maximale.

Une approche simple pour traiter la problématique est de résoudre la relaxation linéaire du PLNE en relaxant la contrainte (2.4) d'intégrité des x_{ij}. La solution fractionnaire obtenue doit alors être transformée en une solution entière.

Modèle de Flot L'objectif du routage global est d'affecter un ensemble de ressources afin de satisfaire un ensemble de demandes. La nature du problème est équivalente à la recherche d'un flot optimal dans un graphe [99, 3]. Quand il n'est pas possible de modéliser la problématique de routage global sous forme d'une problématique de multiflot, des sous-problèmes du routage global sont formalisés sous forme de problématiques de flot [112], ce qui peut donner des solutions de qualité. La problématique multiflot permet de modéliser le routage global entièrement, où chaque reseau est une commodité avec une source et une destination. L'avantage d'une telle formalisation est la possibilité de modéliser la problématique sous forme de PLNE. Du fait de la complexité du routage global (taille, nombre de réseaux,...) les approches [148, 137, 147, 4] proposées sont basées sur des heuristiques et des algorithmes d'approximation combinatoire [159].

Ces différents algorithmes de base sont mis en oeuvre dans différentes approches : résolution séquentielle, résolution globale, routage hiérarchisé et algorithme génétique.

Une approche séquentielle, comme son nom l'indique, route les réseaux itérativement. L'inconvenient d'une telle approche est l'existence d'un lien entre la qualité de la solution obtenue et l'ordre de routage. Quelque

soit l'ordre de routage choisi, il est toujours très difficile de router les derniers réseaux. Abel [2] est arrivée à la conclusion qu'il n'existe pas de méthode d'ordonnancement des réseaux meilleure que toutes les autres méthodes.

Dans [84] les auteurs proposent un algorithme de routage s'inspirant des mouvements de particules dans un champs de force, s'inspirant de la méthode de champ potentiel développée pour la planification de route pour un robot [88]. Durant le routage, une force d'attraction, le long de la ligne reliant la source et la destination et une force de répulsion, générée par les sources et destinations des réseaux non routés sont prises en compte sous forme de coûts dans le calcul de la trajectoire. Ainsi, la solution est moins dépendante de l'ordre de routage. Dans [26] les auteurs proposent un algorithme séquentiel minimisant le plus grand taux de congestion et la longueur des réseaux. La méthode route itérativement les réseaux en utilisant un algorithme d'approximation générant un arbre de Steiner de coût minimum.

Les deux exemples cité précédemment ont une approche gloutonne du routage, il n'y a pas de remise en cause de routages précédemment effectués. Une méthode nommé « Rip-Up and Reroute » (littéralement supprimer et dérouter) a été proposée par [155] pour atténuer la problématique d'ordonnancement du routage des réseaux lors d'une approche séquentielle. Le principe est d'identifier les zones à forte congestion et de dérouter les réseaux concernés (traversant ces zones) vers des zones moins congestionnées. Dans [123] l'auteur propose une variante du « Rip-Up and Reroute ». Dans cette variante tous les réseaux sont déroutés, passant ou pas par les zones congestionnées. Cela est justifié par le fait qu'un réseau ne traversant pas une zone congestionnée peut être positionné sur une zone plus éloignée pour laisser libre place à un reseau se trouvant sur une zone adjacente congestionnée. De plus les réseaux sont toujours re-routés dans le même ordre. Dans [112], les auteurs quant à eux proposent de re-router les réseaux simultanément en utilisation une formulation permettant de determiner une solution entière pour un flot maximal à coût minimal en un temps polynômial du fait de l'intégrité des capacités [3]. D'autres approches « Rip-Up-and-Reroute » sont décrites dans [87].

Les approches globales qui consistent à router les réseaux simultanément sont basées sur une formulation multiflot [3] de la problématique. Soit un graphe $G(V, E)$, où $V = v_1, v_2, \cdots, v_n$ l'ensemble des sommets et $E = e_1, e_2, \cdots, e_m$ l'ensemble des arcs. Dans un tel graphe, un ensemble de k commodités doivent être acheminées entre un certain nombre de sommets. Du point de vue du routage global dans un circuit VLSI, l'ensemble des réseaux $N = n_1, n_2, \cdots, n_k$ est l'ensemble des commodités. À chaque commodité n_i correspond un ensemble de terminaux et une demande d_i qui est égale à 1. À chaque arc $e \in E$ sont associés une capacité $c(e)$ et un coût $l(e)$. Le routage peut être exprimé par une formulation basée sur les arcs ou par une formulation basée sur les arbres de Steiner. Pour une formulation basée sur les arcs, le flot $f(e)$ représente la quantité de flot traversant l'arc $e \in E$. En revanche, pour la formulation basée sur les arbres, un ensemble $T_i = t_{i1}, t_{i2}, \cdots$ d'arbres possibles est associé à chaque réseau $n_i \in N$. Soit la variable $x_{ij} \in \{0, 1\}$, $x_{ij} = 1$ si l'arbre t_{ij} est sélectionné pour le réseau $n_i \in N$, 0 sinon. Typiquement, il y a deux contraintes qui doivent être satisfaites dans une problématique de multiflot : la quantité de flot transmise à une commodité i doit être égale à la demande d_i ; la totalité du flot $f(e)$ passant pas l'arc $e \in E$ ne peut excéder la capacité $c(e)$ de l'arc e. Du fait de la complexité de la problématique de routage global et de la complexité inhérente au PLNE, la contrainte d'intégrité de la problématique de multiflot est relaxée et une solution entière est obtenue en appliquant des méthodes d'arrondi sur la solution optimale fractionnaire.

La modélisation de la problématique de routage global sous forme de problématique multiflot a été initiée par Shragowitz et Keel [148]. Soit $G = (V, E)$ un graphe orienté, la formulation proposée ne s'applique qu'à des réseaux de degré 2. Pour tout $n_i \in N$, $v_s \in V$ est la source du réseau n_i, c'est-à-dire que $d_i(v_s) = 1$ et $v_t \in V$ est la destination, donc $d_i(v_t) = -1$. Pour plus de simplicité l'arc orienté $e \in E$ est noté par le couple de sommets (v', v). La formulation de routage global par Shragowitz et Keel s'écrit alors :

$$\text{minimiser} \sum_{e \in E} \sum_{n_i \in N} l(e)|f_i(e)| \tag{2.5}$$

sujet à

$$\sum_{v',(v,v') \in E} f_i(v,v') - \sum_{v',(v',v) \in E} f_i(v',v) = d_i(v), \forall n_i \in N, \forall v \in V \tag{2.6}$$

$$\sum_{n_i \in N} |f_i(e)| \leq c(e), \forall e \in E \tag{2.7}$$

$$f_i(e) \in \{0, \pm 1\}, \forall n_i \in N, \forall e \in E \tag{2.8}$$

L'approche proposée se déroule en deux étapes : tout d'abord une premiere solution est calculée en ignorant la contrainte de capacité (2.7), ce qui fournit une borne inférieure. Ensuite, itérativement l'approche tente de réduire la violation de la contrainte de capacité en identifiant les arcs ayant le plus grand dépassement de capacité afin d'en détourner une partie des réseaux qui l'empruntent tout en minimisant l'augmentation de la fonction de coût, qui est la longueur des réseaux. Cette approche est similaire à l'approche « Rip-Up-and-Reroute ».

Raghavan et Thompson [137] proposent quant à eux un modèle de flot restreint aux réseaux de degré trois dans un graphe $G(V, E)$, néanmoins, les auteurs déclarent que la généralisation est possible. L'objectif est de minimiser le flow maximum, c'est-à-dire minimiser $\hat{f} = max(f(e), \forall e \in E)$. Soit un ensemble de réseaux $n_i \in N$ de degré trois, v_{i1}, v_{i2} et v_{i3} étant les trois terminaux. Dans la formulation de Raghavan et Thompson, tout réseau de degré trois nécessite un sommet de Steiner. Soit la variable $s_i(v) \in \{0, 1\}$ indiquant si un sommet $v \in V$ est considéré comme un sommet de Steiner pour le réseau n_i. La problématique de routage global est alors formulée sous forme d'une problématique de multiflot où $s_i(v)$ unité de flot doivent être soustraites à v_{i1}, v_{i2} et v_{i3} pour être envoyées à $v \in V$. Les auteurs proposent alors de résoudre la relaxation du programme linéaire et de converger vers une solution entière par arrondi.

La dernière application du modèle de multiflot à la problématique de routage global est attribuée à Albrecht [4]. Cette approche se base sur l'algorithme d'approximation de multiflot de Garg et Könemann[62], plus simple et plus rapide que celle de Shahrokhi et Matula[147] dont une application au problème de routage global a été proposée par Carden et al [134]. L'approche d'Albrecht considère à la fois la congestion et la longueur des réseaux. L'approche s'applique à un graphe $G = (V, E)$ à deux niveaux, similaire à celui illustré par la figure 2.16(c), avec une direction par niveau (horizontale - verticale). Les arcs entre les niveaux représentent les vias et la capacité de chaque arc est égale au nombre maximum de vias par cellule, une cellule étant associée à un sommet $v \in V$. L'une des particularité de cette approche est la prise en compte de réseaux d'épaisseurs différentes. Soit $w_i(e)$ l'épaisseur d'un réseau $n_i \in N$ sur l'arc $e \in E$. En admettant que $d_i = 1, \forall n_i \in N$, le PLNE s'écrit de la façon suivante :

$$\text{minimiser } \hat{\lambda} \tag{2.9}$$

sujet à

$$\sum_{T_{ij} \in \tau_i} x_{ij} = 1, \forall n_i \in N \tag{2.10}$$

$$\sum_{(i,j),e_{ij} \in T_{ij}} w_i(e) x_{ij} \leq \hat{\lambda} c(e), \forall e \in E \tag{2.11}$$

$$x_{ij} \in \{0, 1\}, \forall n_i \in N, \forall T_{ij} \in \tau_i \tag{2.12}$$

Dans l'optique de réduire la complexité de la problématique du routage global, des approches de routage hiérarchisées ont été proposées. Ces approches sont basées sur le partitionnement des circuits VLSI par des méthodes de « floorplanning » générant un arbre hiérarchisé. À Chaque nœud de l'arbre est associé un problème de routage global résolu séparément par un algorithme exact de programmation linéaire en nombres entiers. Les solutions ainsi obtenues sont combinées pour donner une solution globale à la problématique de routage, sachant que les solutions partielles sont optimales. Cette approche a été initiée par Bursetin et Pelavin [21] pour un type de topologie des circuit VLSI et a été par la suite généralisée par Luk et al [161] à différentes topologies. Il s'en suit de nombreuse améliorations dont un aperçue est donnée dans [102].

Ces cinq dernières années ont vu émerger de nouvelles approches de routage global initiées à la faveur de concours organisés pendant The International Symposium on Physical Design (ISPD) en 2007 et 2008 : BoxRouter 2.0 [33, 32], DpRouter [30], Archer [124], FastRoute 4.0 [128, 127, 166, 164], FGR [141], MaizeRouter [116], NTHU-Router [31], NTUgr-Router [79] pour ne citer qu'eux. Ces approches sont basées, pour la plupart sur une méthodologie de résolution séquentielle associée à une méthode « Rip-Up and Reroute ». Une étape initiale consiste à décomposer les réseaux de degré $n > 2$ en un ensemble de réseaux de degré égal à deux. La décomposition est effectuée grâce à la construction d'un arbre de Steiner rectilinéaire de poids minimum pour chaque réseau. Après quoi, la méthode « Rip-Up and Reroute » est mise en oeuvre de différentes manières, mais avec comme seul objectif de dérouter les réseaux se trouvant dans une zone congestionnée, l'objectif principal étant de router tous les réseaux en minimisant les situations de congestion. Ci-dessous une description de deux approches BoxRouter 2.0 [32], et MaizeRouter [116] est proposée. Une description des autres approches est donnée dans [117].

BoxRouter 2.0 L'idée de base de l'approche BoxRouter 2.0 [32], initiée par BoxRouter 1.0 [33] est de progressivement étendre une boîte englobant la région sujete à la plus forte congestion après le routage d'initialisation (FLUTE) et d'appliquer, aux réseaux appartenant à la boite, un PLNE avec comme objectif de maximiser le nombre de réseaux routés sans congestion. Les réseaux non routés sont quant à eux reroutés (Rip-Up and Reroute) en utilisant un algorithme $A*$ dynamique dont la fonction de coût permet de détourner les réseaux des zones congestionnées à l'itération courante et de celles congestionnées aux itérations précédentes, principe inspiré de [111]. Ce principe est le suivant : à chaque itération, caractérisée par l'expansion de la boite englobante, le coût de congestion des arcs est mis à jour de sorte qu'un arcs successivement congestionné sur plusieurs itérations se verra attribuer un coût de congestion plus élevé. Ce détournement doit être effectué sans modifier la forme générale de l'arbre de Steiner. À la fin de la procédure, les réseaux son affectés aux diffèrent niveaux grâce à un PLNE.

MaizeRouter L'approche MaizeRouter est basée sur trois mécanismes : Extreme Edge Shifting, Edge Retraction et Garbage Collection. Une étape initiale consiste à déterminer un arbre de Steiner rectilinéaire de poids minimum pour chaque réseau indépendamment des autres réseaux. Après quoi, l'approche s'attelle à corriger les situations de congestion en effectuant des opérations sur les arcs composants l'arbre de Steiner. Les trois mécanismes cités ci-dessus constituent ces opérations. L'Edge Shifting consiste à déplacer un arc d'une zone fortement congestionnée vers une zone dont la capacité n'a pas été atteinte, comme illustré par les figures 2.22(a)-(b) où deux arcs ont été déplacés, un arc vers le coté gauche pour la figure 2.22(a) et un arc vers le haut pour la figure 2.22(b). Ceci permet de réduire le taux de congestion. L'Extreme Edge Shifting est inspiré du mécanisme de Edge Shifting initié dans [128, 127, 166, 164], et qui à l'inverse de l'Extreme Edge Shifting n'augmente pas la longueur de l'arbre de Steiner lors du déplacement d'un arc. Ces déplacements peuvent créer des arcs superflus qu'il faut supprimer. Le Garbage Collection est chargé de cette opération comme illustré par la figure 2.22(c). La figure 2.22(d) illustre le résultat final. La qualité d'un routage étant évaluée du point de vue de la longueur des réseaux et du taux de congestion, il peut arriver que l'opération d'Extreme Edge Shifting produise des arcs parallèles trop longs lors du déplacement d'un arc d'une zone congestionnée vers une zone qui ne l'est pas, comme illustré par la figure 2.22(a)-(c). Ceci a pour effet de déséquilibrer la fonction de coût. Afin de rétablir l'équilibre, un mécanisme de Edge Retraction entre en jeu. Il a pour objectif de réduire la longueur de l'arbre de Steiner en réduisant la

longueur des arcs rallongés par une étape d'Extreme Edge Shifting, comme illustré par la figure 2.22(f). Une étape de Garbage Collection est aussi nécessaire après une étape d'Edge Retraction, figure 2.22(e). Une fois les réseaux routés, une étape d'affectation des réseaux aux couches est effectuée. Pour les figures 2.22(a-g), la couleur gris foncé indique une zone à forte congestion, gris clair indique une zone dont la capacité est atteinte ou presque, et la couleur blanche indique une zone avec peu de demandes.

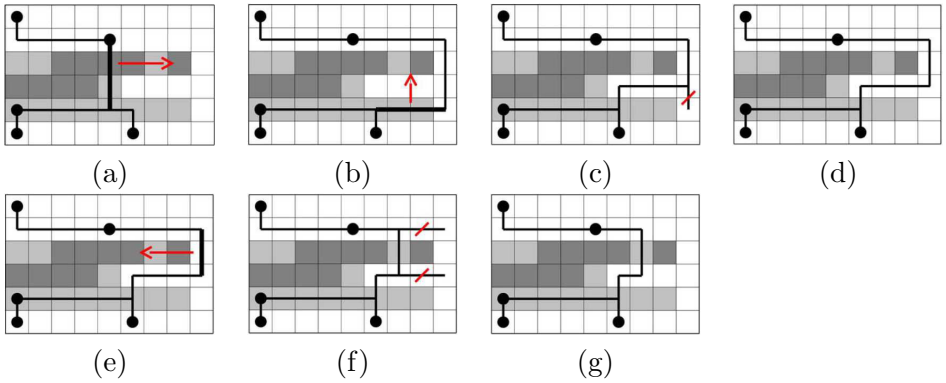


FIGURE 2.22 – Extreme Edge Shifting, Edge Retraction et Garbage Collection

2.6.2 Routage détaillé

Il existe un nombre important de modèles de routage utilisés pour traiter la problématique de routage détaillé. Le modèle le plus utilisé est le routage sur une grille, un modèle auquel va se limiter cette section. Le lecteur pourra se référer à [104] pour plus de details sur l'ensemble des modèles.

Les problématiques de routage global sont définies par la forme de la zone de routage et la position des terminaux. Comme dit précédemment, le routage global permet de définir grossièrement le comportement des réseaux autour des composants : quel réseau traverse quelle cellule de la zone de routage. Il est difficile d'indiquer quel point du réseaux est en contact avec la frontière d'une cellule. Les points d'intersection des réseaux avec les frontières des cellules sont appelés pseudo-terminaux. Afin d'effectuer les routages détaillés, il est nécessaire de positionner ces pseudo-terminaux. Cette opération est effectuée grâce à des heuristiques.

Une fois les pseudo-terminaux positionnés, il est possible de distinguer trois problématiques de routage détaillé dans une grille : routage de type Canal, routage de type SwitchBox et routage de type Standard.

Canal Soit une grille rectangulaire discrétisant une cellule d'un circuit VLSI, les pseudo-terminaux appartenant à cette cellule, dans un routage de type canal, sont tous situés sur le bord supérieur et inférieur de la grille. Ce positionnement est illustré par la figure 2.23 (terminaux représentés par des numéros).

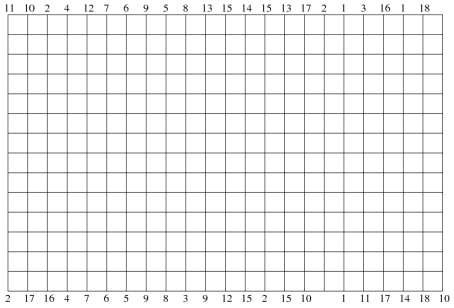


FIGURE 2.23 – Routage de type Canal

SwicthBox Pour un routage de type SwitchBox, les pseudo-terminaux sont positionnés sur les quatre bords de la grille rectangulaire, comme illustré par la figure 2.24.

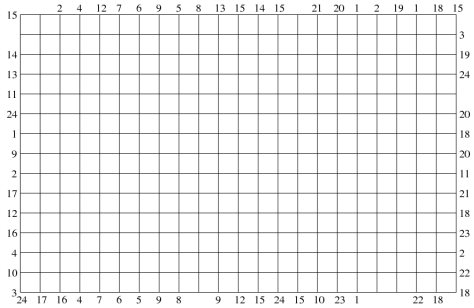


FIGURE 2.24 – Routage de type SwitchBox

Standard Pour un routage de type Standard, les pseudo-terminaux peuvent être positionnés n'importe ou sur la grille, avec la présence de composants, comme illustré par la figure 2.25.

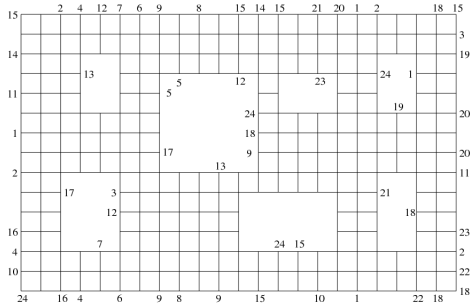


FIGURE 2.25 – Routage de type Standard

La problématique de routage détaillé est traitée par les mécanismes de base cités précédemment dans la section 2.6.1 mais aussi par des heuristiques très spécialisées. Ainsi, dans [165, 80], les auteurs proposent des algorithmes de routage de type Canal et SwitchBox respectivement. Les deux algorithmes sont basés sur l'extension de lignes horizontales sur un niveau et de lignes verticales sur un niveau adjacent. Le contact entre les lignes verticales et horizontales est effectué grâce au positionnement de vias. Dans [165], les auteurs proposent un algorithme de routage de type canal sur deux niveaux : un niveau avec des pistes horizontales et un second niveau avec des pistes verticales. Le principal objectif est de minimiser le nombre de pistes horizontales pour effectuer le routage. Ce routage doit être effectué sans que deux réseaux ne soient en contact sur une piste verticale ou horizontale. La figure 2.26 illustre un routage de type canal, avec un routage utilisant trois pistes horizontales, figure 2.26(a) et un routage n'utilisant que deux pistes horizontales grâce à l'introduction de coude (dogleg, en anglais) pour le réseau numéro 2, figure 2.26(b). L'utilisation de coude permet non seulement de minimiser le nombre de pistes horizontales utilisées, mais aussi de réparer les situations de conflit rencontrées sur des pistes verticales. Comme illustré par la figure 2.27, où le réseau numéro 2 est en conflit vertical avec la piste numéro 1 sur la portion de ligne en pointillé (figure 2.27(a)), ce coude permet de réparer cette situation de conflit, figure 2.27(b).

Afin de déterminer l'ordre de routage des lignes verticales et horizontales, un graphe de contraintes verticales et une représentation en zone, des segments horizontaux sont définis. Le graphe de contraintes verticales permet d'ordonnancer le routage des réseaux sur les colonnes. Soit le routage de type Canal illustré par la figure 2.28(a), les segments numérotés indiquent les terminaux de chaque réseau. La figure 2.28(b) représente le graphe $G(V, E)$ de contraintes verticales associées à l'exemple de la figure 2.28(b). $v_i \in V$ est le sommet associé au réseau $n_i \in N$ et l'arc $(i, j) \in A$ indique que tous les segments horizontaux du réseau n_i doivent être positionnés au dessus de ceux du réseau n_j. Cela permet d'éviter les conflits dans une même colonne. Soit deux réseaux ayant une seule

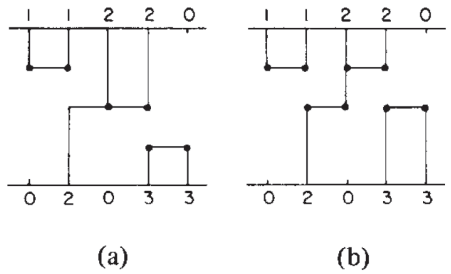


FIGURE 2.26 – Routage de type canal

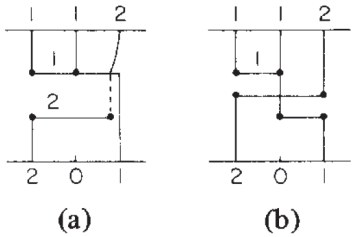


FIGURE 2.27 – Illustration de la nécessité d'utiliser un coude

ligne horizontale. Le réseau connecté aux terminaux situés au bord supérieur du canal dans une colonne donnée doit forcement être au dessus d'un réseau connecté aux terminaux situés au bord inférieur du canal dans cette même colonne. Les réseaux 1 et 2 de la figure 2.28(a) illustrent ce cas.

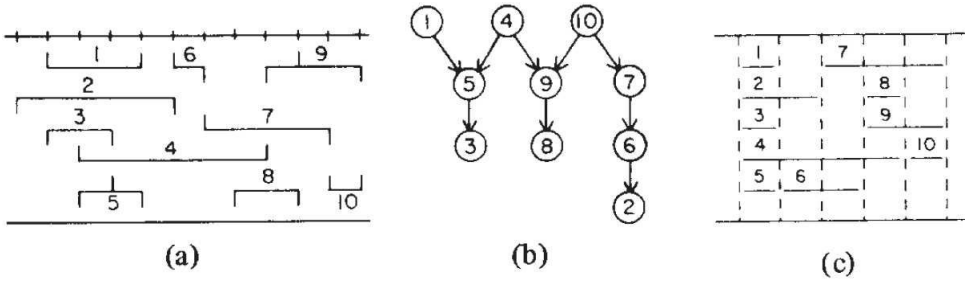


FIGURE 2.28 – (a) exemple de canal de routage, (b) graphe de contraintes verticales, (c) représentation en zone

La figure 2.28(c) illustre une représentation du canal de routage en zone. Le canal de routage est partitionné en cinq zones representant cinq répartitions des réseaux sur les colonnes. Par exemple, la zone numéro 1 regroupe les réseaux 1, 2, 3, 4 et 5 car ces réseaux sont présents dans les cinq premières colonnes du canal de routage, comme illustré par la figure 2.28(a). Cette représentation traduit le fait que les réseaux d'une même zone ne doivent pas être présents sur une même piste horizontale, ce qui doit être le cas des réseaux 1, 2, 3, 4 et 5. Le nombre de réseaux par zone représente la densité de la zone. Un dernier mécanisme permet d'unir un ensemble de réseaux dans la représentation en zone du moment que ces réseaux peuvent être sur la même ligne, comme pour les réseaux 6 et 9 de la figure 2.28(a). Les figures 2.29(a)-(b), respectivement graphe de contraintes verticales et représentation en zones illustrent le résultat d'une telle union. Après une succession d'unions, le graphe de contraintes verticales obtenu est illustré par la figure 2.29(c), avec une répartition par piste (track en anglais) horizontale des réseaux. Le résultat final du routage est illustré par la figure 2.29(d).

Ainsi, une instance de routage de type Canal est entièrement caractérisée par le graphe de contraintes verticales et la représentation en zones. Cela permet de déterminer un ordre optimal des réseaux respectant les contraintes verticales du graphe et utilisant un nombre de pistes horizontales égal à la densité maximale déduite de la représentation en zones. Pour l'exemple illustré par la figure 2.29(d), le densité maximale est égale à 5.

Dans [80], les auteurs présentent un algorithme permettant de traiter un routage de type SwitchBox avec obstacle. Cette approche est basée sur un algorithme glouton proposé par Rivest et Fiducia [140], permettant de router des instances de type Canal. L'algorithme effectue un balayage de la gauche vers la droite en avançant de colonne en colonne. La passage d'une colonne i à la colonne $i+1$ n'est effectué que si la colonne i est entièrement routée. Le routage d'une colonne consiste à relier un terminal se trouvant sur le bord supérieur (respectivement inférieur) à une piste horizontale attribuée au même réseau ou à la premiere piste horizontale libre, ceci grâce à une piste verticale d'une longueur minimale. Lors du routage, un réseau peut occuper plusieurs pistes horizontales. L'un des objectifs de la phase de routage est de réunir plusieurs pistes d'un même réseau sur une même piste afin de minimiser leur nombre. Par exemple, sur la figure 2.30 l'algorithme, à la colonne numéro

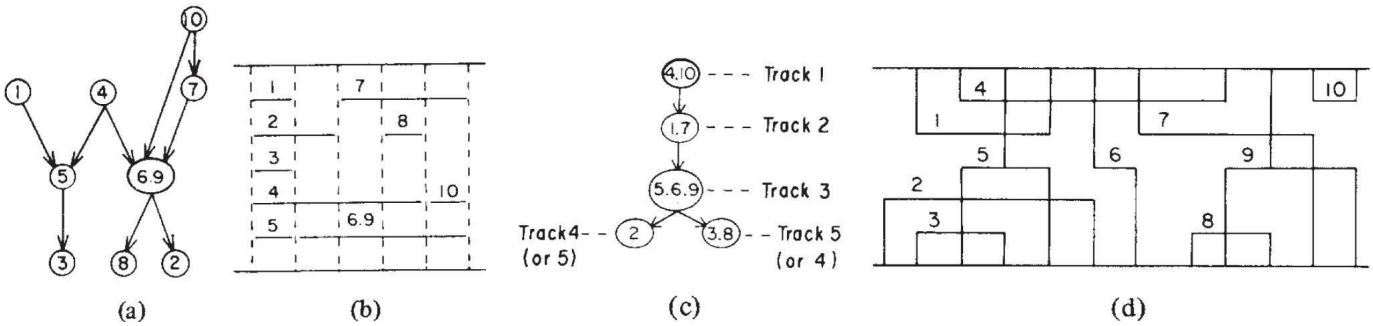


FIGURE 2.29 – (a) exemple de canal de routage, (b) graphe de contraintes verticales, (c) représentation en zones

6 a eu recours à une piste verticale pour réunir les deux pistes horizontales appartenant au réseau numéro 2. La minimisation de ce nombre passe aussi par l'utilisation de coude permettant d'atteindre une piste horizontale déjà occupée par le même réseau, par exemple le réseau 4 à la colonne numéro 6.

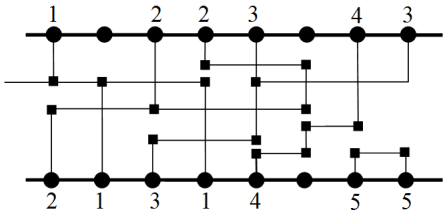


FIGURE 2.30 – Exemple de canal de routage

L'approche proposée par Rivest et Fiducia [140] permet de connecter des réseaux ayant leurs terminaux sur le bord supérieur et/ou inférieur du canal, mais aussi sur le bord gauche du canal qui est le point de depart du routage. Dans [80], les auteurs adaptent cette approche pour connecter des réseaux ayant un ou plusieurs terminaux sur le bord droit, ce qui est le cas pour un routage de type SwitchBox. De plus, l'approche proposée route une instance de type SwitchBox avec obstacle. Les modifications apportées consistent en la possibilité de réserver une piste horizontale sur un certain nombre de colonnes pour un réseau ayant un ou plusieurs terminaux sur le coté droit du canal et en relaxant la contrainte imposant une direction par niveau, ceci pour permettre le contournement d'obstacles. Un mécanisme de partitionnement du SwitchBox en deux sous canaux est mis en place pour limiter la congestion des pistes horizontales et verticales. Le partitionnement est effectué dans le sens de la verticale et au niveau de la colonne dont la densité est maximale.

2.7 Conclusion

La problématique de routage de guides d'ondes est assimilable aux problématiques de pipe routing, cependant, elle se distingue par un espace de routage plus ouvert et une liberté de routage plus importante. Les méthodes proposées, que ce soit la méthode exacte ou les méthodes heuristiques sont inédites pour ce type de problématique, à l'image des heuristiques simple et grand voisinage avec réparation. Ces deux heuristique s'inspirent des méthodes « Rip-Up and Reroute » (littéralement supprimer et dérouter), mais en appliquant une méthode de réparation qui permet de s'approcher l'optimalité.

La problématique VLSI est assez particulière du fait des contraintes imposées. Les méthodes utilisées sont très spécifiques à la problématique VLSI. Elles proposent des concepts très intéressant qui mériterait d'être étudiés pour la problématique de routage, tel que le routage global et le routage détaillé.

Chapitre 3

Formalisation du WGRP et Résolution Exacte

Dans ce chapitre, nous présentons tout d'abord la modélisation adoptée pour traiter la problématique de multi-routage. Nous présentons ensuite, sur la base de la modélisation choisie, une méthode de résolution exacte mettant en oeuvre une méthodologie de type Branch and Price. Dans la section 4.3 nous présentons le principe de la génération de colonnes, avec une presentation de la relaxation linéaire d'un modèle de couverture, ainsi qu'une description du sous problème associé, qui dans notre cas se réduit à une problématique de recherche de plus court chemin. Dans la section 3.3, une description de la méthodologie de Branch and Price est donnée, comprenant le principe de fonctionnement et les schémas de branchement.

3.1 Formalisation du WGRP

La première étape dans la modélisation du problème abordé consiste à proposer une représentation de l'espace de routage, qui est un espace continu en trois dimensions. Nous adoptons la méthode de décomposition en cellules rectangulaires homogènes pour discrétiser l'espace de routage en une grille en deux dimensions sur plusieurs niveaux, le passage d'un niveau à l'autre ne pouvant se faire qu'entre cases de mêmes coordonnées. Ce choix (plutôt qu'une modélisation en trois dimensions par exemple) s'explique par les méthodes de conception actuelles des guides d'ondes, qui se font niveau par niveau. Pour chaque niveau, nous proposons un maillage en grille rectangulaire, choisi du fait de sa simplicité de mise en œuvre, contrairement à un maillage en hexagone ou en triangle par exemple. La figure 3.1 illustre ces trois type de maillage.

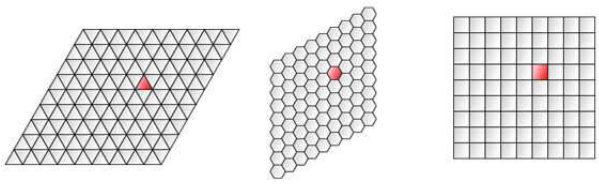


Figure 3.1 – Grille : différentes sortes de maillage

Une case de la grille peut être un espace libre, un obstacle (élément du satellite positionné à cet endroit), un point de départ ou un point d'arrivée de guide d'ondes. Chaque case a au plus 10 cases adjacentes (8 cases adjacentes sur le plan incluant les diagonales et 2 cases pour le changement de niveau). Avec cette modélisation, un guide d'ondes est représenté par un chemin reliant une case origine à une case destination par une ensemble

de cases adjacentes (voir figure 3.2). Afin de simplifier la problématique, l'épaisseur d'un guide d'ondes est supposée égale à la dimension d'une case de la grille.

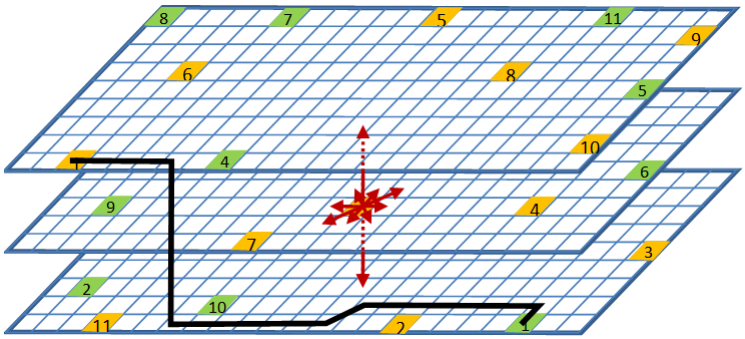


FIGURE 3.2 – Discrétisation de la zone de routage en une grille rectangulaire

Soit $G = (V, A)$ le graphe non orienté construit de la manière suivante. L'ensemble des sommets V est obtenu en introduisant un sommet par case de la grille à l'exception des cases occupées par un obstacle. A est l'ensemble des arcs, où l'arc $(v_i, v_j) \in A$ si les sommets $v_i \in V$ et $v_j \in V$ représentent des cases adjacentes. Soit W l'ensemble des guides d'ondes à router. Un guide d'ondes $w \in W$ est défini par deux sommets terminaux, un sommet source $s_w \in V$ et un sommet destination $t_w \in V$. $V_s = \{s_w, w \in W\} \subset V$ est l'ensemble des sommets origines, $V_t = \{t_w, w \in W\} \subset V$ l'ensemble des sommets destinations. Le positionnement d'un guide d'ondes $w \in W$ sera représenté par un chemin dans le graphe G reliant son sommet origine s_w à son sommet destination t_w. Enfin, à chaque arc $(v_i, v_j) \in A$ est associé un coût positif C_{ij} indiquant sa longueur. Ici, ces longueurs ont la particularité de ne prendre que trois valeurs, correspondant respectivement à des déplacements non-diagonaux dans le plan, diagonaux dans le plan ou entre niveaux. La matrice des longueurs est supposée satisfaire l'inégalité triangulaire, ce qui revient à s'assurer que le coût d'un déplacement diagonal ne dépasse pas le coût de deux déplacements non-diagonaux. Le coût d'un chemin est défini par la somme des coûts des arcs utilisés par ce dernier.

En dehors des limites de cette modélisation liée à la discrétisation de l'espace, une difficulté provient de l'impossibilité physique pour deux guides d'ondes de se chevaucher. Interdire le passage de deux guides d'ondes par un même sommet du graphe n'est pas suffisant pour gérer cette contrainte du fait de la possibilité que deux chemins se croisent en diagonale sur la grille (voir figure 3.3). Afin de prendre en compte cette contrainte, nous introduisons l'ensemble D des points $d \in D$ de croisement de la grille. Sur la figure 3.3, un point de croisement est représenté par la croix (rouge).

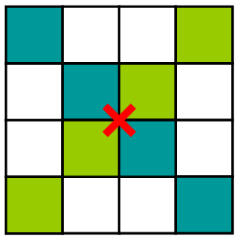


FIGURE 3.3 – Point de croisement sur une grille rectangulaire

Dans un premier temps une version simplifiée de la problématique est étudiée afin de mieux l'appréhender. Seules les contraintes interdisant le chevauchement et le croisement entre guides d'ondes sont prises en compte. Ces deux contraintes sont formalisées par une capacité unitaire respectivement sur les sommets, les arcs et sur les points de croisement $d \in D$. Les contraintes de courbure, d'épaisseur et de portion de départ et d'arrivée seront considérées dans le chapitre 5 où un modèle réaliste du WGRP sera traité. La problématique WGRP simplifiée peut alors être décrite par une premiere modélisation sous forme d'un Programme Linéaire en Nombres Entiers

faisant intervenir des variables de flot. Soit la variable de flot $x_{ij}^w \in \{0,1\}$ associée à un arc $(v_i, v_j) \in A$ et au guide d'ondes $w \in W$. Le modèle PLNE est le suivant :

$$\text{minimiser} \sum_{w \in W} \sum_{(v_i,v_j) \in A} C_{ij} \cdot x_{ij}^w \tag{3.1}$$

sujet à :

$$\sum_{w \in W} \sum_{\{v_j | (v_i,v_j) \in A\}} x_{ij}^w \leq 1 \quad (v_i \in V) \tag{3.2}$$

$$\sum_{\{v_i | (v_i,v_k) \in A\}} x_{ik}^w - \sum_{\{v_j | (v_k,v_j) \in A\}} x_{kj}^w = 0 \quad (w \in W, v_k \in V \backslash (V_s \cup V_t)) \tag{3.3}$$

$$\sum_{w \in W} \sum_{(v_i,v_j) \in A} \delta_{ij}^d \cdot x_{ij}^w \leq 1 \quad (d \in D) \tag{3.4}$$

$$\sum_{\{v_j | (v_s,v_j) \in A\}} x_{sj}^w = 1 \quad (w \in W, v_s = s_w) \tag{3.5}$$

$$\sum_{\{v_i | (v_i,v_t) \in A\}} x_{it}^w = 1 \quad (w \in W, v_t = t_w) \tag{3.6}$$

$$x_{ij}^w \in \{0,1\} \quad ((v_i,v_j) \in A, w \in W) \tag{3.7}$$

où $x_{ij}^w = 1$ si l'arc $(v_i, v_j) \in A$ est utilisé par le guide d'ondes $w \in W$, 0 sinon ; $\delta_{ij}^d = 1$ si l'arc $(v_i, v_j) \in A$ représente deux cases adjacentes de la grille situées en diagonale et traversant le point de croisement $d \in D$, $\delta_{ij}^d = 0$ sinon. Les contraintes de capacité sur les sommets (3.2) et sur les points de croisement (3.4) impliquent respectivement qu'une case ou qu'un point de croisement peuvent être occupées par au plus un guide d'ondes (capacités unitaires). La contrainte (3.3) permet la conservation du flot et les contraintes (3.5)-(3.6) assurent que les terminaux $s_w \in V$ et $t_w \in V$ seront bien visités.

L'inconvénient du modèle (3.1)-(3.7) réside dans le nombre très important de variables lié à la taille de la grille, le nombre de niveaux et de guides d'ondes. La génération de colonnes permet cependant de calculer sa relaxation linéaire en considérant de manière implicite les variables, ce qui lui permet de résoudre efficacement les programmes linéaires de grande taille. Le modèle (3.1)-(3.7) n'est cependant pas exploitable sous cette forme par la génération de colonne. Il nécessite une reformulation préalable.

3.2 Génération de Colonnes

La décomposition de Dantzig-Wolfe appliquée au modèle (3.1)-(3.7) nous permet d'obtenir le modèle de couverture (3.8)-(3.12).

Soit $\Omega = \{r_1, r_2, ..., r_{|\Omega|}\}$ l'ensemble des chemins r_k (aussi appelés routes) réalisables pour les guides d'ondes $w \in W$. $\Omega^w \subset \Omega$ est le sous-ensemble de Ω correspondant aux chemins réalisables de w, c'est-à-dire reliant s_w à t_w. La longueur c_k de la route $r_k \in W$ est égale à la somme des longueurs des arcs empruntés ; $a_{ik} = 1$ si la route r_k contient le sommet $v_i \in V$, 0 sinon ; $b_{ij}^k = 1$ si la route r_k utilise l'arc $(v_i, v_j) \in A$, 0 sinon ; $\delta_{dk} = 1$ si la route $r_k \in \Omega$ traverse le point de croisement $d \in D$, 0 sinon.

Le modèle de couverture est alors le suivant :

$$\text{minimiser} \sum_{r_k \in \Omega} c_k \cdot \theta_k \tag{3.8}$$

sujet à

$$\sum_{r_k \in \Omega} a_{ik} \cdot \theta_k \leq 1 \quad (v_i \in V), \tag{3.9}$$

$$\sum_{r_k \in \Omega} \delta_{dk} \cdot \theta_k \leq 1 \quad (d \in D), \tag{3.10}$$

$$\sum_{r_k \in \Omega^w} \theta_k \geq 1 \quad (w \in W), \tag{3.11}$$

$$\theta_k \in \{0,1\} \quad (r_k \in \Omega), \tag{3.12}$$

où $\theta_k = 1$ si la route r_k est sélectionnée dans la solution, 0 sinon. Les contraintes de capacités (3.9) et (3.10) impliquent respectivement qu'une case ou qu'un point de croisement peuvent être occupés par au plus un guide d'ondes. La contrainte (3.11) assure qu'une route $r_k \in \Omega$ est attribuée à tout guide d'ondes $w \in W$.

3.2.1 Principe

Le principe de la génération de colonnes a été suggéré en premier par Ford et Fulkerson [58] en 1958, pour être ensuite présenté deux ans plus tard, en 1960 par Dantzig et Wolfe [40] pour les problèmes linéaires avec des structures décomposables. Finalement, Gilmore et Gomory [67] l'ont utilisée pour résoudre le problème de découpage, **Cutting-Stock**. Une synthèse des principaux travaux portant sur la génération de colonnes est présentée par Soumis [152].

Dans la suite du mémoire, nous appellerons problème maitre MP (pour Problème Maître) la relaxation linéaire de (3.8)-(3.12) et problème maitre restreint MP(Ω_1) la restriction du problème maitre au sous ensemble $\Omega_1 \subset \Omega$. Soit MP(Ω_1) :

$$\text{minimiser} \sum_{r_k \in \Omega_1} c_k \cdot \theta_k \tag{3.13}$$

sujet à

$$\sum_{r_k \in \Omega_1} a_{ik} \cdot \theta_k \leq 1 \quad (v_i \in V) \tag{3.14}$$

$$\sum_{r_k \in \Omega_1} \delta_{dk} \cdot \theta_k \leq 1 \quad (d \in D) \tag{3.15}$$

$$\sum_{r_k \in \Omega_1^w} \theta_k \geq 1 \quad (w \in W) \tag{3.16}$$

$$\theta_k \geq 0 \quad (r_k \in \Omega_1) \tag{3.17}$$

Soit $D(\Omega_1)$ le dual de $MP(\Omega_1)$:

$$\text{maximiser} \quad -\sum_{v_i \in V} \lambda_i - \sum_{d \in D} \mu_d + \sum_{w \in W} \sigma_w \tag{3.18}$$

sujet à

$$-\sum_{v_i \in V} a_{ik}\lambda_i - \sum_{d \in D} \delta_{dk} \cdot \mu_d + \sigma_w \leq c_k \quad (w \in W, r_k \in \Omega^w) \tag{3.19}$$

$$\lambda_i \geq 0 \quad (v_i \in V) \tag{3.20}$$

$$\mu_d \geq 0 \quad (d \in D) \tag{3.21}$$

$$\sigma_w \geq 0 \quad (w \in W) \tag{3.22}$$

Dans ce modèle, λ_i est la variable duale positive associée à la contrainte (3.14), μ_d est la variable duale positive associée à la contrainte (3.15) et σ_w la variable duale positive associée à la contrainte (3.16).

Soit une solution optimale du $MP(\Omega_1)$ fournie par le simplexe. Il en découle une solution optimale $(\lambda^*, \mu^*, \sigma^*)$ pour $D(\Omega_1)$, le problème dual de $MP(\Omega_1)$.

Proposition 3.2.1 *Soit $\Omega_1 \subset \Omega$ et $(\lambda^*, \mu^*, \sigma^*)$ la solution optimale de $D(\Omega_1)$. Si $(\lambda^*, \mu^*, \sigma^*)$ est réalisable pour $D(\Omega)$ alors, $(\lambda^*, \mu^*, \sigma^*)$ est optimale pour $D(\Omega)$.*

Cette propriété découle du fait que l'espace des solutions réalisables de $D(\Omega)$ est inclus dans celui de $D(\Omega_1)$. Si $(\lambda^*, \mu^*, \sigma^*)$ est réalisable pour $D(\Omega)$ cette propriété et le théorème de la dualité permettent donc de conclure sur la valeur de la solution optimale de $MP(\Omega)$.

Dans le cas où la solution $(\lambda^*, \mu^*, \sigma^*)$ n'est pas réalisable pour $D(\Omega)$, une ou plusieurs contraintes portant sur les routes $r_k \in \Omega \backslash \Omega_1$ ont été violées. Le principe de la génération de colonnes est d'identifier une ou plusieurs de ces contraintes à l'aide d'un sous-problème, ceci dans le but d'ajouter les routes (les variables) correspondantes à Ω_1. Ansi résoudre alternativement $MP(\Omega_1)$ et le sous-problème permet de converger vers une solution duale réalisable. La génération de colonnes se termine lorsque le sous-problème ne trouve plus de contraintes violées (plus aucune colonne, route n'est ajoutée à Ω_1). La convergence de l'algorithme est garantie par le caractère fini de l'ensemble Ω. La génération de colonnes est donc une méthode itérative résolvant MP(Ω_1) puis enrichissant Ω_1 si besoin à l'aide d'un sous-problème. L'objectif du sous-problème est de détecter s'il existe dans $\Omega \setminus \Omega_1$ une ou plusieurs colonnes de coût réduit négatif. L'algorithme de génération de colonnes est décrit par la figure 3.4.

Générer un ensemble initial de colonnes Ω_1
Faire
    Résoudre MP(Ω_1)
    $\Gamma \leftarrow$ *colonne*(s) retournée(s) par le sous problème
    $\Omega_1 \leftarrow \Omega_1 \cup \Gamma$
Tant que $\Gamma \neq \emptyset$

FIGURE 3.4 – Algorithme de génération de colonnes

L'algorithme de génération de colonnes fonctionne donc sur le principe d'enrichissement d'un ensemble initial Ω_1 permettant d'obtenir une premiere solution réalisable. Or, pour le modèle (3.8)-(3.12) il n'est pas toujours aisé de trouver un ensemble initial Ω_1 fournissant une solution réalisable. Soit la variable auxiliaire $z_w \in \{0, 1\}$, $z_w = 1$ si aucune route n'a été attribuée au guide d'ondes $w \in W$, 0 sinon. Soit le grand nombre M définissant la pénalité associé au non-routage d'un guide d'ondes et assurant que n'importe quelle solution réalisable pour MP sera meilleure qu'une solution ne routant pas un ou plusieurs guides d'ondes. L'introduction de la variable

auxiliaire z_w permet de garantir l'existence d'une solution réalisable quelque soit l'ensemble initial Ω_1 fourni. Il en résulte les modifications suivantes du modèle : une modification de la fonction objectif (3.8), avec l'introduction des z_w pénalisées d'un coût M, comme illustré par l'équation (3.23) ; la modification de la contrainte (3.11), avec l'ajout de la variable z_w, comme illustré par l'equation (3.26) ; l'ajout de la contrainte d'intégrité (3.28) de la variable z_w, qui est relaxée dans le nouveau modèle (3.23)-(3.28), décrit ci-dessous :

$$\text{minimiser} \sum_{r_k \in \Omega_1} c_k \cdot \theta_k + M \cdot \sum_{w \in W} z_w \tag{3.23}$$

sujet à

$$\sum_{r_k \in \Omega_1} a_{ik} \cdot \theta_k \leq 1 \quad (v_i \in V) \tag{3.24}$$

$$\sum_{r_k \in \Omega_1} \delta_{dk} \cdot \theta_k \leq 1 \quad (d \in D) \tag{3.25}$$

$$\sum_{r_k \in \Omega_1^w} \theta_k + z_w \geq 1 \quad (w \in W) \tag{3.26}$$

$$\theta_k \geq 0 \quad (r_k \in \Omega_1) \tag{3.27}$$

$$z_w \geq 0 \quad (w \in W) \tag{3.28}$$

Notons que ces modifications n'affectent pas l'écriture du dual (3.18)-(3.22), si ce n'est l'ajout de nouvelles contraintes induites par les variables z_w. En particulier, les contraintes 3.19 sont inchangées. Il est dit précédemment que lors d'une itération de la génération de colonnes, l'objectif du sous-problème est de détecter s'il existe dans $\Omega \setminus \Omega_1$ une ou des routes de coût réduit négatif, c'est-à-dire susceptibles d'améliorer la solution courante. En se basant sur le dual du modèle (3.23)-(3.28), la condition indiquant si le coût réduit d'une route $r_k \in \Omega \setminus \Omega_1$ est négatif est la suivante :

$$c_k + \sum_{v_i \in V} a_{ik} \lambda_i + \sum_{d \in D} \delta_{dk} \mu_d - \sigma_w < 0, \tag{3.29}$$

En rappelant que λ_i, μ_d et σ_w sont les variables duales positives respectivement associées aux contraintes (3.24), (3.25) et (3.26), et $w \in W$ est le guide d'ondes associé à $r_k \in \Omega^w$.

De manière équivalente, et pour mettre en évidence la nature du sous-problème, la condition (3.29) peut également s'écrire :

$$\sum_{(v_i,v_j) \in A} b_{ij}^k (c_{ij} + \lambda_i) + \sum_{(v_i,v_j) \in A} b_{ij}^k \sum_{d \in D} \delta_{ij}^d \mu_d < \sigma_w - \lambda_{t_w} \tag{3.30}$$

Le sous-problème revient ainsi à déterminer, pour chaque guide d'ondes $w \in W$ s'il existe un chemin reliant s_w à t_w de coût strictement inférieur à $\sigma_w - \lambda_{t_w}$, où le coût de l'arc (v_i, v_j) est fixé à $\tilde{c}_{ij} = c_{ij} + \lambda_i + \sum_{d \in D} \delta_{ij}^d \cdot \mu_d$. En notant que les coûts définis sur les arcs sont positifs, il s'agit donc simplement, pour chaque guide d'ondes, d'un problème de plus court chemin classique.

3.2.2 Résolution du sous problème

Pour résoudre le sous-problème, nous proposons d'utiliser l'algorithme A* [125, 126]. Afin de générer dans la mesure du possible plusieurs chemins par itération, nous proposons une phase de post-traitement présentée dans la sous-section 3.2.2.2.

3.2.2.1 A*

L'algorithme **A*** [125, 126] peut être vu comme une extension de l'algorithme de Dijkstra [55] pour laquelle, au lieu de traiter les sommets dans l'ordre de leur distance par rapport au point de départ, le critère de sélection du prochain sommet prend en compte une évaluation (optimiste) de la distance jusqu'au point de destination.

Plus précisément, pour un guide d'ondes $w \in W$ donné, le choix du prochain sommet v_i à étendre est effectué selon un critère $f(i)$ qui est une combinaison linéaire de deux coût $g(i)$ et $h_w(i)$: le coût $g(i)$ correspond à la distance déjà parcourue entre la source s_w et le sommet v_i ; le coût $h_w(i)$ est une estimation de la distance qui reste à parcourir pour atteindre la destination t_w. Notons $f(i) := g(i) + h_w(i)$ la fonction d'évaluation du sommet v_i. L'optimalité de l'algorithme A^* [46] est garantie sous condition de monotonie de h_w, à savoir $h_w(i) \leq \tilde{c}_{ij} + h_w(j)$ pour tout arc (v_i, v_j).

Afin de proposer une fonction h_w donnant une évaluation aussi précise que possible de la distance restant à parcourir, nous définissons $h_w(i)$ comme étant égal au coût du plus court chemin reliant v_i à la destination v_t avec les distances initiales (coûts c_{ij}). Du fait que $c_{ij} \leq \tilde{c}_{ij}$ et du fait que la matrice des coûts c respecte l'inégalité triangulaire, la condition de monotonie est vérifiée. Pour tout guide d'ondes $w \in W$, la valeur de la fonction h_w est calculée en pré-traitement. Pour cela, l'algorithme de Dijkstra est utilisé en prenant comme source le sommet t_w.

Il existe d'autre heuristiques, décrites en annexe permettant d'estimer la distance restante :
- Distance de Manhattan, l'heuristique standard du **A***,
- Distance Euclidienne,
- Distance Diagonale,

3.2.2.2 Post-traitement

Afin d'accélérer la convergence de la génération de colonne, il est préférable de générer plusieurs colonnes par appel au sous-problème. Le post-traitement proposé exploite le fait qu'à la fin de l'algorithme **A***, il reste un certain nombre de portions de chemin optimales, reliant s_w aux sommets v_i. Il est alors possible d'avoir $f(i) = g(i) + h_w(i) < \sigma_w - \lambda_{t_w}$ pour certains de ces sommets v_i. Les valeurs $g(i)$ et $h_w(i)$ sont respectivement associées aux chemins reliant s_w à v_i et v_i à t_w, dont la concaténation pourrait ainsi permettre d'obtenir un chemin reliant s_w à t_w de coût réduit négatif. Le principe du post-traitement est de parcourir l'ensemble de ces chemins et de calculer leur coût vis-à-vis de la fonction de coût $\tilde{c}$. Précisons que seuls les chemins reliant les sommets v_i à t_w ont en fait besoin d'être parcourus, puisque leurs coûts $h(i)$ ont été calculés avec la matrice de coût initiale. Lorsqu'un chemin de coût réduit négatif est trouvé, il est inséré dans Ω_1.

Afin de limiter les temps de calcul et de générer, dans la mesure du possible des chemins assez différents et de bonne qualité, un filtrage est effectué. Premièrement, les sommets adjacents au plus court chemin fourni par l'algorithme A* sont exclus. Deuxièmement, les sommets v_i tels que le plus court chemin permettant d'atteindre v_i se dirige dans une direction opposée à celle du vecteur reliant s_w à t_w sont aussi supprimés.

3.3 Branch and Price

Les domaines d'applicabilité préférentiels de la génération de colonnes sont les problèmes d'optimisation combinatoire liés au transport et à la planification (de transport, d'horaires de personnel . . .). Elle est souvent utilisée pour résoudre des problèmes linéaires dont la contrainte d'intégrité a été relaxée. Associée à une méthode d'énumération implicite, elle permet de trouver une solution entière optimale. On parle alors du **Branch and Price**.

Pour traiter notre problématique, trois méthodes différentes d'exploitation de la génération de colonnes s'offraient à nous. La première méthode, dite Off-Line, résout le PLNE sur un ensemble restreint de colonnes, initialement générées (voir par exemple Hoffman and Padberg [85]). La seconde méthode, dite dynamique, résout la relaxation continue du PLNE, pour ensuite appliquer un algorithme d'énumération implicite sur les colonnes générées afin d'obtenir la meilleur solution entière possible. Cette méthode a été mise en œuvre pour résoudre le problème du pairage d'equipages pour les longs courriers par Barnhart et al. [23].

La troisième méthode, le Branch and Price, la seule exacte, consiste à appliquer la génération de colonnes dynamique au travers d'un algorithme d'énumération implicite. Elle garantie l'optimalité de la solution entière obtenue car une borne inférieure exacte est calculée à chaque nœud de l'arbre de recherche. Malheureusement cette optimalité est obtenue au prix d'un effort considérable du fait du grand nombre de colonnes générées et de nœuds traités.

Le risque avec les deux premieres approches est de trouver une solution très éloignée de la solution optimale, ou de ne pas trouver de solution, même avec un plus grand nombre de colonnes. La troisième méthode, celle que nous avons retenue, nous garantit de trouver une solution en un temps fini, qui plus est optimale.

3.3.1 Principe

La génération de colonnes fournit soit une solution entière, soit une solution fractionnaire qui nécessite un mécanisme de séparation et d'évaluation afin de converger vers une solution entière. Un tel mécanisme va partitionner un ensemble S de solutions du Programme Linéaire en Nombres Entiers MP en plusieurs sous-ensembles à travers un arbre de recherche. Chaque nœud u de l'arbre correspond à un $PLNEMP^u$ dont l'ensemble des solutions réalisables est noté $S^u \subset S$. La phase d'évaluation est effectuée par la génération de colonnes qui est appliquée à la relaxation linéaire RMP^u du PLNE MP^u. Deux états distincts peuvent être obtenus, soit on obtient une solution entière et dans ce cas le processus se termine, soit on obtient une solution fractionnaire qui nécessite une méthode de séparation qui, appliquée au problème MP^u va séparer son ensemble de solutions (S^u) en deux sous-ensembles disjoints S^v et S^w correspondant aux deux problèmes fils MP^v et MP^w. Il est alors possible qu'une solution entière puisse être trouvée dans les deux fils ou plus profondément dans l'arbre de recherche. Reste à définir la politique de parcours de l'arbre en définissant les règles de branchement. Lors de l'évaluation d'un nœud u, si la valeur de la relaxation, celle de RMP^u est supérieure ou égale à la meilleure solution entière connue, le nœud est rejeté.

3.3.2 Règles de branchement

Un schéma de branchement classique consiste à créer deux fils avec deux contraintes opposées portant sur une variable fractionnaire, la variable θ_k dans notre cas. Ainsi, une variable θ_k fractionnaire, associée à une route r_k est sélectionnée et deux fils sont créés :

- $\theta_k = 1$

- $\theta_k = 0$

Imposer $\theta_k = 1$, c'est-à-dire imposer l'utilisation d'une route r_k est assez aisé, contrairement au fait d'interdire l'utilisation d'une route r_k. La variable θ_k est une variable artificielle qui n'est pas representative de la décision locale prise à chaque nœud de l'arbre. Il est communément admis que les variables de flot de la formulation initiale (3.1)-(3.7) sont plus représentatives et doivent être utilisées pour le branchement. Dès lors, il est possible de créer un branchement, soit sur un sommet, soit sur un arc.

Sommet Soit f(i) la fonction de flot définie de la façon suivante, $f(i) = \sum_{r_k \in \Omega_1} a_{ik} \cdot \theta_k$. Le sommet $v_i \in V$ est sélectionné pour un branchement si la valeur du flot qui lui est associé est fractionnaire.

Deux fils sont alors crées :

- Une branche où le sommet $v_i \in V$ ne peut appartenir à la solution entière ($f(i) = 0$),
- Une branche où le sommet $v_i \in V$ doit obligatoirement appartenir à la solution entière ($f(i) = 1$),

Il est assez aisé d'appliquer la contrainte $f(i) = 0$. Dans le Problème Maître, il suffit de supprimer les routes r_k utilisant le sommet v_i. Dans le sous problème il faut interdire l'utilisation de v_i par les autres routes. Pour la contrainte $f(i) = 1$, le sommet v_i doit obligatoirement être utilisé par une route r_k faisant partie de la solution entière. Dans le Problème Maître, la contrainte (3.14) doit être modifiée de la façon suivante :

$$\sum_{r_k \in \Omega_1} a_{ik} \cdot \theta_k = 1 \tag{3.31}$$

Par contre aucune modification ne doit être apportée au sous problème.

Arc Soit f(i,j) la fonction de flot définie de la façon suivante, $f(i,j) = \sum_{r_k \in \Omega_1} b_{ij}^k \cdot \theta_k$. L'arc $(v_i, v_j) \in A$ est sélectionné pour un branchement si la valeur du flot qui lui est associée est fractionnaire. Sur le même principe que le sommet, deux fils sont créés :

- Une branche où l'arc $(v_i, v_j) \in A$ ne peut appartenir à la solution entière ($f(i,j) = 0$),
- Une branche où l'arc $(v_i, v_j) \in A$ doit obligatoirement appartenir à la solution entière ($f(i,j) = 1$),

Comme pour un sommet, il est assez aisé d'appliquer la contrainte $f(i,j) = 0$. Dans le Problème Maître, il suffit de supprimer les routes r_k utilisant l'arc (v_i, v_j). Dans le sous problème il faut interdire l'utilisation de (v_i, v_j) par les autres routes. Concernant la contrainte $f(i,j) = 1$, l'arc (v_i, v_j) doit obligatoirement être utilisé par une route r_k faisant partie de la solution entière. Dans le Problème Maître, la contrainte (3.32) doit être ajoutée.

$$\sum_{r_k \in \Omega_1} b_{ij}^k \cdot \theta_k = 1, (v_i, v_j) \in A \tag{3.32}$$

La aussi, le sous problème ne nécessite pas de modifications, excepté d'intégrer la variable duale associée à la nouvelle contrainte sur l'arc (v_i, v_j).

Malgré sa simplicité de mise en oeuvre, le schéma précédemment décrit ne peut être retenu, car les contraintes (3.31) et (3.32) peuvent faire intervenir des coûts duaux négatifs. Ceci peut potentiellement générer des cycles de coût négatif. À y regarder de plus prêt, on se rend compte qu'il suffirait d'interdire l'utilisation d'un nœud ou d'un arc pour tous les guide d'ondes $w \in W$ sauf un. L'avantage d'un tel schéma est de nous garantir que quelque soit la contrainte ajoutée, le graphe $G = (V, A)$ ne contiendra jamais de cycle de coût négatif. Un tel schema est proposé par Barnhart et al. dans [22] pour traiter une problématique de multiflot entier. Cependant, l'inconvénient d'un tel schéma est le nombre de fils crées. Contrairement au schéma précédent, dont l'arbre de

recherche est binaire, ce nouveau schéma nécessite la création d'autant de fils que de guides d'onde utilisant un sommet (ou un arc). Ansi, l'arbre de recherche associé à ce nouveau schéma est n-aires.

De la même manière que le schéma précédemment défini, le nouveau schéma est défini pour un sommet et pour un arc, mais cette fois si pour un guide w spécifique :

Sommet par guide d'ondes $w \in W$ Pour chaque guide d'onde $w \in W$ utilisant le sommet $v_i \in V$ sélectionné, l'utilisation du sommet v_i est interdite pour l'ensemble des guides d'ondes $\{w' \in W, w' \neq w\}$ utilisant v_i, et est autorisée pour le guide d'ondes w. Ce qui se traduit par :

- Dans le Problème Maître, il suffit de supprimer l'ensemble des routes $\{r_k \in \Omega_1^{w'}, w' \in W, w' \neq w\}$ utilisant le sommet v_i,
- Dans le sous problème, il faut interdire l'utilisation de v_i à tous les guides d'ondes $w' \neq w$.

Arc par guide d'ondes $w \in W$ Pour chaque guide d'onde $w \in W$ utilisant l'arc $(v_i, v_j) \in A$ sélectionné, l'utilisation de l'arc (v_i, v_j) est interdite pour l'ensemble des guides d'ondes $\{w' \in W, w' \neq w\}$ utilisant (v_i, v_j), et est autorisée pour le guide d'ondes w. Ceci se traduit par :

- Dans le Problème Maître, il suffit de supprimer l'ensemble des routes $\{r_k \in \Omega_1^{w'}, w' \in W, w' \neq w\}$ utilisant l'arc (v_i, v_j),
- Dans le sous problème, il faut interdire l'utilisation de l'arc (v_i, v_j) à tous les guides d'ondes $w' \neq w$.

Le sommet v_i et l'arc (v_i, v_j) sont sélectionnés de la manières suivante : l'élément (arc ou sommet) ayant un flot fractionnaire et faisant intervenir le plus grand nombre de guides d'ondes (traiter les zones de conflit les plus encombrées en premier) est sélectionné en priorité. Du fait de l'équivalence entre les deux types de branchement, dans la suite du mémoire, les branchements seront effectués selon les sommets.

3.4 Expérimentations

3.4.1 Description des instances de test

Les expérimentations sont effectuées sur deux séries d'instances, une première série d'instances représentatives des différents cas de figure qui peuvent être rencontrés lors du routage des guides d'ondes. Ces cas de figure sont : le croisement de deux guides d'ondes sur le plan (figure 3.5) ; le goulot d'étranglement qui oblige les guides d'ondes à passer dans un espace restreint (figure 3.6) ; le virage en S qui est une succession de deux virages où un guide d'ondes est successivement à l'intérieur et à l'extérieur de ces derniers (figure 3.7). La seconde série regroupe des instances où chaque instance est composée des cas de figure précédemment cités.

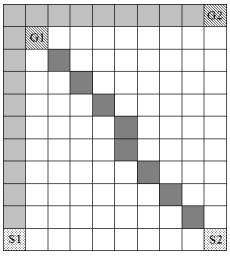


FIGURE 3.5 – Croisement

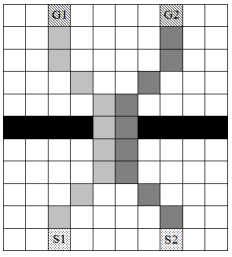


FIGURE 3.6 – Goulot d'étranglement

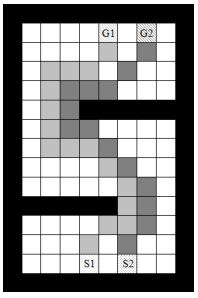


FIGURE 3.7 – Virage en S

Une instance est définie par :

(X, Y, Z) la taille de la grille, où X est le nombre de ligne, Y le nombre de colonne et le nombre de niveaux autorisés,

Nwg le nombre de guide d'ondes à router,

Occ le taux d'occupation égal au ratio entre le nombre de cases occupées et le nombre total de cases.

Dans la suite du mémoire, nous notons **GC** la bornes inférieure fourni par la génération de colonnes et B&B la méthode exacte Branch and Price.

3.4.2 Résultats

Les tableaux 3.1 et 3.2 illustrent les résultats d'expérimentations effectuées sur les instances décrites ci-dessus. La méthode est évaluée selon la borne obtenue, le nombre de guides d'ondes fixés et le temps CPU en seconde. Le temps d'exécution a été limité à 20 min, soit 1200 secondes, ce qui constitue un temps raisonnable pour résoudre les instances de tests. Lorsque, pour une instance donnée, cette limite est dépassée, le temps CPU indique $\sim$. Parmi les 63 instances, la méthode B&B en a résolu 10 dans le temps limite accordé. Pour chaque type d'instances, les instances résolues sont les plus simples en terme de taille de zone de routage (X,Y,Z) et en nombre de guides d'ondes à router. Quand une solution entière n'est pas trouvée, la méthode fixe très peu de variable (de guide d'ondes). Il lui est difficile de traiter les cas de croisement, de virage en S et de goulet d'étrangement pour des zones de routage dépassant les dimensions $X = 15$ et $Y = 15$. On observe également une incapacité à traiter les changements de niveau.

Pour les instance réelles, la méthode B&B fixe très peu de variables, au maximum 30% d'entre elles sont fixées.

3.4.3 Conclusion

Une méthode exacte est en théorie très séduisante et promet l'optimalité des solutions fournies, ceci pour des problématiques complexes. Néanmoins, dans la pratique cette méthode se révèle souvent innapropriée. Pour la problématique de routage, la difficulté de cette méthode est le schéma de branchement choisi qui, conjugué à la taille des zones de routage a engendré une complexité affectant la capacité de convergence de la méthode B&B, malgré notre tentative d'accélérer la convergence en ajoutant plus de colonnes par itération.

Dans la suite du mémoire nous proposons deux méthodes heuristiques permettant de traiter la problématique plus efficacement que la méthode B&B.

(X,Y,Z)	Nwg	Occ	GC	B&B	Nb wg fixée	CPU (s)
Croisement						
(5,5,1)	2	0	9,54	10,83	2	14,25
(10,10,1)	2	0	25,27	29,31	2	32,11
(30,30,1)	2	0	83,55	-	0	∼
(50,50,1)	2	0	141,84	-	0	∼
Goulet d'étranglement						
(15,15,1)	4	4	65,11	65,11	4	72,77
(15,15,1)	4	9	66,53	66,53	4	65,24
(15,15,1)	5	22	85,67	-	2	∼
(30,30,1)	5	8	163,95	-	1	∼
(30,30,1)	6	9	201,43	-	1	∼
(30,30,1)	7	5	225,21	-	2	∼
(30,30,1)	7	7	217,17	-	2	∼
(50,50,1)	11	3	593,26	-	3	∼
(50,50,1)	11	7	618,12	-	4	∼
(50,50,2)	16	16	353,78	-	6	∼
Virage en S						
(15,20,1)	2	6	53,18	53,18	2	55,02
(15,20,1)	3	6	77,08	77,08	3	58,12
(15,20,1)	4	6	98,23	-	2	∼
(30,30,1)	2	3	85,98	85,98	2	117,48
(30,30,1)	3	3	124,37	-	1	∼
(30,30,1)	4	3	156,07	-	1	∼
(30,30,1)	5	3	210,25	-	1	∼
(50,50,1)	2	3	185,77	-	0	∼
(50,50,1)	3	3	267,78	-	0	∼
(50,50,1)	4	3	344,37	-	0	∼
(50,50,1)	5	3	447,21	-	0	∼
Croisement / Goulet d'étranglement / Virage en S						
(10,10,1)	2	56	27,90	27,90	2	30,15
(10,10,1)	3	18	27,31	27,31	3	37,14
(10,10,1)	4	22	36,97	36,97	4	36,15
(10,10,2)	6	42	37,56	-	3	∼
(25,25,1)	5	11	141,81	-	2	∼
(25,25,1)	6	14	206,02	-	2	∼
(25,25,1)	5	8	256,92	-	2	∼
(25,25,1)	6	17	175,54	-	2	∼
(30,30,2)	15	19	203,84	-	6	∼
(30,30,2)	23	24	395,79	-	8	∼
(30,30,2)	28	22	286,53	-	8	∼
(50,50,2)	14	34	491,25	-	4	∼
(50,50,2)	31	20	563,26	-	8	∼
(50,50,2)	38	24	794,12	-	10	∼
(200,120,1)	42	28	1384,76	-	12	∼
(200,120,1)	42	30	1416,55	-	11	∼
(100,100,1)	23	14	836,74	-	6	∼
(100,100,1)	27	18	1025,16	-	7	∼

TABLE 3.1 – B&B : Résultats d'expérimentations

(X,Y,Z)	Nwg	Occ	GC	B&B	Nb wg fixée	CPU (s)
(215,140,3)	80	24	1932,26	-	23	∼
(215,140,3)	37	21	832,05	-	11	∼
(215,140,3)	138	34	3791,39	-	32	∼
(215,140,3)	112	31	2277,68	-	27	∼
(215,140,3)	140	34	3266,65	-	29	∼
(215,140,3)	96	27	2470,16	-	22	∼
(250,230,2)	13	7	1034,00	-	5	∼
(250,230,2)	48	2	3078,69	-	12	∼
(250,230,3)	91	23	5650,09	-	17	∼
(300,200,3)	37	18	1154,86	-	11	∼
(300,200,3)	74	22	1780,69	-	16	∼
(300,200,3)	74	23	1890,74	-	16	∼
(300,200,3)	80	21	2715,13	-	20	∼
(300,200,3)	96	25	3601,32	-	27	∼
(300,200,3)	112	28	3137,49	-	26	∼
(300,200,3)	123	27	5211,58	-	26	∼
(300,200,3)	138	32	5369,17	-	29	∼
(300,200,3)	140	32	4572,77	-	28	∼
(300,200,3)	148	30	5912,13	-	29	∼
(300,200,3)	168	35	6846,32	-	27	∼

TABLE 3.2 – B&B : Résultats d'expérimentations - instances réelles

CHAPITRE 4

RÉSOLUTIONS APPROCHÉES DU WGRP

Dans ce chapitre, nous présentons deux approches heuristiques pour la résolution du WGRP, toutes deux basées sur des ajustements de chemins valides. L'une met en œuvre un algorithme de type glouton associé à un mécanisme de réparation locale, l'autre est basée sur la génération de colonnes. Une présentation des résultats et une conclusion clôtureront ce chapitre.

4.1 Heuristique gloutonne avec réparation locale

L'heuristique proposée est basée sur une approche séquentielle, l'algorithme glouton, où les guides d'ondes sont routés itérativement sous contraintes d'obstacles et des guides précédemment routés. Cette heuristique est présentée dans l'algorithme 1. Le routage d'un guide d'ondes est au plus effectué en trois temps. Une premiere étape consiste à déterminer le plus court chemin, grâce à l'algorithme **A***, du guide d'ondes $w \in W$ de l'itération courante. Le routage est effectué en relâchant la contrainte de capacité des cases (sommets) occupées par des guides d'ondes $w' \in W$ précédemment routés. Pour ces cases, la capacité n'est plus unitaire mais égale à deux, c'est-à-dire qu'en plus du guide d'ondes $w' \in W$, ces cases peuvent être occupées par le guide $w \in W$ de l'itération courante si nécessaire. Ces cases sont dites en situation de conflit. Cette premiere étape consiste donc à déterminer le plus court chemin du guide d'ondes $w \in W$ en l'autorisant à entrer en conflit avec un ou plusieurs guides précédemment routés. Il est clair, qu'un tel guide d'ondes ne peut être en conflit, sur une case donnée, qu'avec un seul guide d'ondes précédemment routé. Soit r le plus court chemin associé au guide d'ondes $w \in W$. Une succession de case (une chaîne) appartenant à la route r est appelée zone de conflit, si toutes les cases sont en situation de conflit.

Cette approche ressemble au mécanisme de **Rip-Up and Reroute** utilisé par les approches de routage global de circuit VLSI pour supprimer les zones de congestion. Malgré tout, ici seul une ou plusieurs portions d'une route sont remises en question lors de la réparation et non pas son intégralité comme dans le mécanisme **Rip-Up and Reroute**. Une remise en cause complète n'est effectuée que lors de l'échec de la réparation.

Une fois le routage de terminé, trois cas de figures peuvent être rencontrés :
- s'il n'y a pas de route possible, le guide d'ondes w est ignoré,
- s'il n'y a pas de zone de conflit, le plus court chemin est retenu
- s'il y a une ou plusieurs zones de conflit, la fonction de réparation est exécutée consécutivement sur chaque zone. Si toutes les zones de conflit sont réparées, le passage à l'itération suivante est effectué, dans le cas contraire, le guide d'ondes w est routé une seconde fois, mais cette fois-ci sans relâcher les contraintes de capacité (pas de conflit autorisé).

Le passage à l'itération suivante est effectué quelque soit le résultat du routage.

Bien que selon Abel [2] il n'existe pas de méthode d'ordonnancement des réseaux meilleure que toutes les autres, dans notre approche, les guides d'ondes ne sont pas routés dans un ordre quelconque. Ils sont ordonnés selon un ordre croissant des longueurs des routes associées aux guides d'ondes. Ces routes ont été déterminées lors d'un routage initial où les contraintes de capacité sur les cases et sur les points de croisement n'ont pas été considérées. Ce type de routage est dit routage indépendant. Cet ordonnancement permet de router d'abord les guides d'ondes les plus court afin de privilégier les connexions directes.

Différents mécanismes sont mis en jeu lors du routage et la réparation locale. Ces mécanismes sont décrits dans la section qui suit.

Algorithme 1: Glouton + réparation

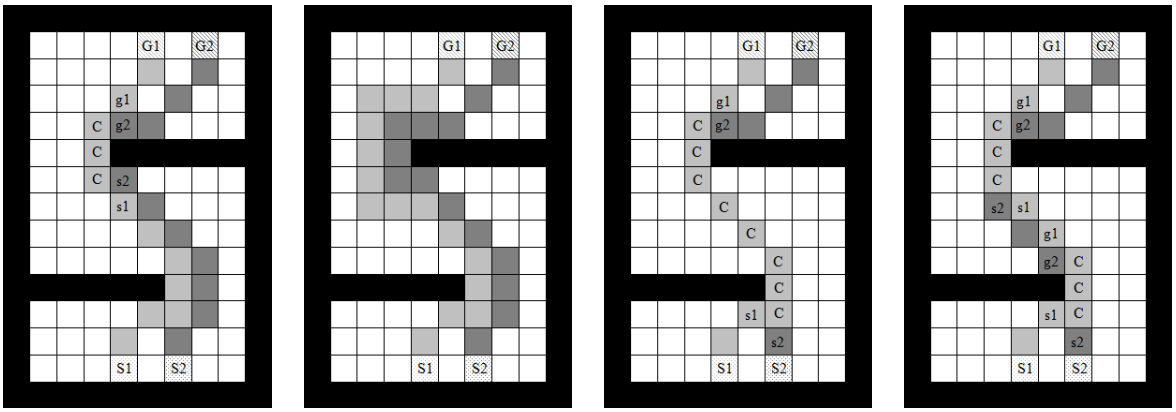
Trier W, dans le sens croissant des distances A^*.
pour tout $w \in W$ **faire**
  $r_w = A^*(w, 1)$ { Retourne un chemin r_w si il existe, null sinon. L'entrée en conflit est autorisée (1).}
  si $r_w \neq null$ **alors**
    si r_w est en conflit avec un ou plusieurs guides d'ondes **alors**
      $r'_w = reparer(r_w)$ {Retourne un chemin r'_w en cas de succès de la réparation, null sinon.}
      si $r'_w \neq null$ **alors**
        r'_w est associé à w.
      sinon
        $r_w = A^*(w, 0)$ { Retourne un chemin r_w si il existe, null sinon. L'entrée en conflit n'est plus autorisée (0).}
        r_w est associé à w.
      finsi
    sinon
      si $r_w \neq null$ **alors**
        r_w est associé à w.
      finsi
    finsi
  finsi
fin pour

4.2 Réparation locale

Les différents mécanismes régissant la réparation locale vont être exposés dans cette section. Prenons l'exemple d'un routage sur un seul niveau, illustré par la figure 4.1(a). Nous sommes en présence d'une grille representant une zone de routage : une case noire est un obstacle ; une case blanche est une case libre ; les cases libellées $S1$ et $G1$ sont respectivement la source et la destination du guide d'ondes numéro 1 ; $S2$ et $G2$ la source et la destination du guide d'ondes numéro 2. La configuration de routage de cet exemple est ce que nous appelons un virage en S.

Dans cet exemple, le guide n°2 est en conflit avec le guide d'ondes n°1 précédemment routé. La zone de conflit est représentée par les cases libellées C. Afin de réparer cette situation de conflit, des points de départ et d'arrivée secondaires, $s1$, $g1$, $s2$, $g2$ sont positionnés aux frontières de la zone de conflit, comme illustré par la figure 4.1(a). La frontière d'une zone de conflit est délimitée par une case appartenant à une route, qui n'est pas en situation de conflit et qui est adjacente à une case en situation de conflit. La réparation consiste alors à determiner les plus courts chemins reliant les points de départ et d'arrivée secondaires en respectant les contraintes de capacité (entrée en conflit interdite). Le résultat est illustré par la figure 4.1(b).

Afin de garantir le succès de la réparation, deux règles doivent être impérativement respectées. La première est liée à la taille de la zone de conflit. Cette taille doit être limitée au maximum. C'est à dire qu'un chemin associé à un guide d'ondes ne rentrera en conflit avec un autre chemin qu'en dernier recours. La figure 4.1(c) illustre le cas où cette taille n'est pas limitée. On peut constater que la zone de conflit est très étendue par rapport à l'exemple de la figure 4.1(a). Une zone de conflit trop grande empêche toute tentative de réparation car les chemins reliant les points secondaires $s1$ à $g1$ et $s2$ à $g2$ se bloqueront mutuellement. La limitation de cette zone de conflit est effectuée en pénalisant l'utilisation d'une case en situation de conflit par un coût supplémentaire. Ce coût d'occupation est pris en compte dans la fonction de coût $f(i)$, de l'algorithme A^* par l'ajout d'une fonction de coût supplémentaire $c(i)$, comptabilisant les coûts d'occupation. Ainsi, la nouvelle fonction de coût $f(i)$ s'écrit : $f(i) = g(i) + h(i) + c(i)$.



(a) (b) (c) (d)

FIGURE 4.1 – Routage de deux guides d'ondes

La seconde règle concerne le croisement durant une situation de conflit entre les chemins associés aux guides d'ondes, plus précisément, durant la sortie d'une situation de conflit. Sur notre exemple, lors du routage du guide d'ondes n°2, il est impératif d'interdire le croisement avec le guide d'ondes n°1. La figure 4.1(d) illustre le cas où cette règle n'est pas respectée. On y observe deux zones de conflit et un croisement entre les deux chemins, ce qui empêche toute tentative de réparation. Un calcul des produits vectoriels $\overrightarrow{s1C_1} \cdot \overrightarrow{s2C_1}$ et $\overrightarrow{C_2g1} \cdot \overrightarrow{C_2g2}$ permet de détecter un croisement lors d'une situation de conflit, et donc de l'interdire. La figure 4.2 illustre ce calcul : le guide d'ondes n°2 est en conflit avec le guide n°1 et il s'apprête à sortir de cette situation de conflit par la case $s2$, qui représente la frontière de la zone de conflit. En effectuant le calcul des produits vectoriels $\overrightarrow{s1C_1} \cdot \overrightarrow{s2C_1}$ et $\overrightarrow{C_2g1} \cdot \overrightarrow{C_2g2}$ et en comparant leurs signes, on arrive à détecter que l'utilisation de la case $s2$ entraînerait une situation de croisement entre les deux guides d'ondes. Le test de croisement est effectué à chaque sortie d'une situation de conflit.

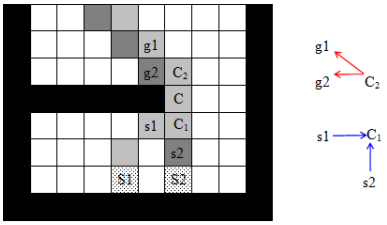
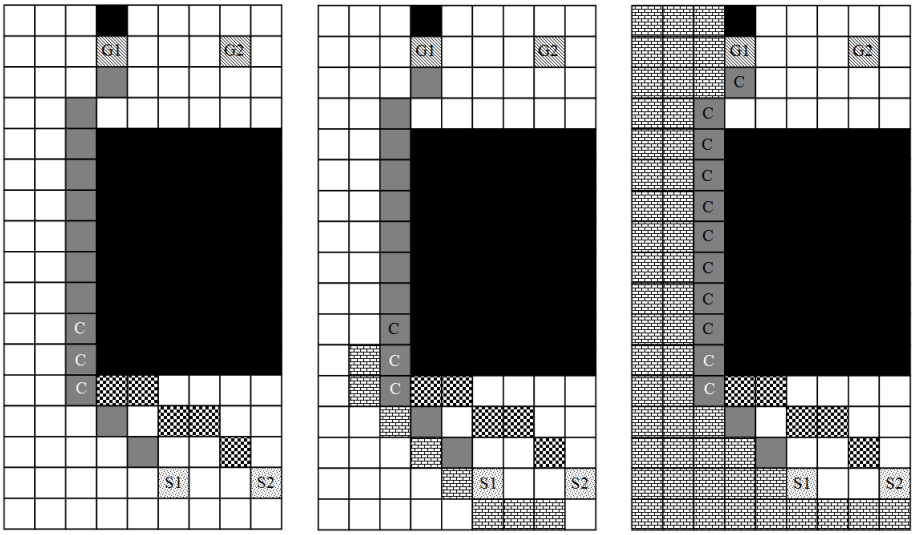


FIGURE 4.2 – Produits Vectoriels

Malheureusement, l'interdiction de croisement rompt l'optimalité de l'algorithme A*. Prenons l'exemple de la figure 4.3(a) qui illustre deux guides d'ondes, le guide d'ondes n°1 précédemment routé (l'ensemble des cases gris foncé) et le n°2 qui est en train de l'être. L'algorithme **A*** effectue une premiere extension du chemin associé au guide d'ondes n°2, représentée par les cases en damier. Durant cette extension, il entre en conflit avec le guide

d'ondes n°1 par le coté droit. Cette situation de conflit est matérialisée par les cases marquées d'une lettre C de couleur blanche. L'algorithme **A*** étend une deuxième extension par le coté gauche, représentée par les cases en brique, cas illustré par la figure 4.3(b). À son tour, la seconde extension entre en conflit avec le guide d'ondes n°1 au niveau de la case marquée d'une lettre C noire, case déjà occupée par la premiere extension mais qui va être affectée à la seconde extension à la faveur d'un coût plus faible. Cette situation aurait été impossible si l'on avait uniquement considéré les coûts de déplacement, or le coût d'occupation servant à limiter la zone de conflit est pris en compte, ce qui a pour effet de favoriser l'extension du coté gauche. Une fois cette domination effective, l'extension du coté droit est perdue et il ne reste plus que celle du coté gauche qui ne peut croiser le chemin associé au guide d'ondes n°1 afin d'atteindre sa destination. Le routage devient alors impossible, comme le montre la figure 4.3(c) où toutes les cases ont été explorées en vain.



(a) (b) (c)

FIGURE 4.3 – Multi-labels

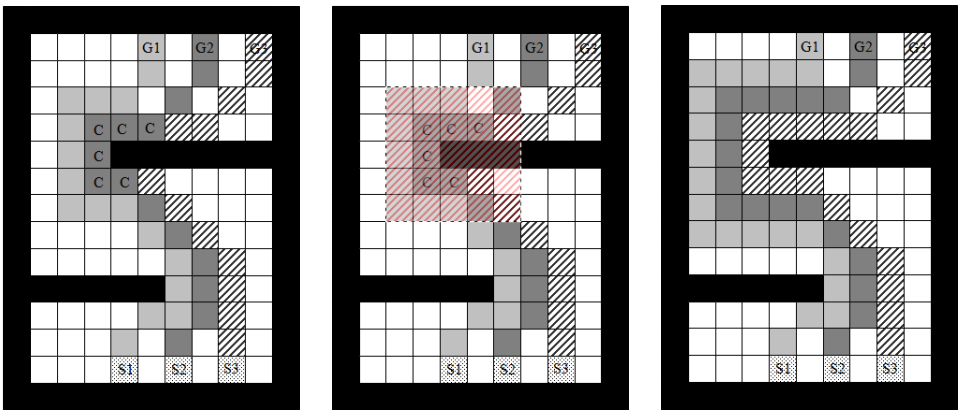
La solution adoptée pour rétablir l'optimalité de l'algorithme **A*** est de créer un label gauche et un label droit en plus du label utilisé jusqu'à présent, le label neutre. Lors de l'expansion, si une situation de conflit est détectée, le label correspondant au coté d'entrée en conflit, gauche ou droit est étendu. Quant au label neutre, ce dernier est utilisé si aucun conflit n'est détecté. Les règles de dominance adoptées sont les suivantes : un label neutre ne peut dominer qu'un label neutre. Les labels droit et gauche ne peuvent dominer qu'un label du même type ou un label neutre. Lors de l'expansion, tous les labels d'une case doivent être étendus. L'algorithme de recherche de plus court chemin A* devient alors multi-label.

Dans l'optique d'améliorer l'efficacité de l'algorithme **A***, l'extension d'un label n'est pas effectuée pour toutes les directions réalisables. En effet, une analyse simple montre qu'un déplacement dans le plan provoquant un changement d'angle supérieur à l'angle droit ramène sur une case adjacente à la case visitée précédemment, ce qui à un coût nécessairement supérieur à un déplacement direct. Ainsi, par exemple, le chemin reliant successivement les cases de coordonnées $(0,0,0)$, $(1,0,0)$ et $(0,1,0)$, formant ainsi un angle de 135°, sera dominé par le chemin reliant directement $(0,0,0)$ à $(0,1,0)$.

La solution proposée jusqu'à present permet de réparer une situation de conflit entre deux guides d'ondes. Pour un nombre plus important de guides d'ondes, une nouvelle notion doit être introduite. Prenons l'exemple de trois guides d'ondes, illustré par la figure 4.4(a) : les guides d'ondes n°1 et n°2 précédemment routés (figures 4.1(a)-(b)) et le guide d'ondes n°3 en conflit avec le n°2. Si on appliquait le même raisonnement que précédemment, seul le routage des guides d'ondes n°2 et n°3 serait remis en cause. Or, pour avoir l'espace nécessaire au routage des trois guides d'ondes, le routage du guide n°1 doit être remis en cause malgré le fait qu'il ne soit pas concerné par

la situation de conflit entre les guides d'ondes n°2 et n°3. Nous introduisons alors la notion de boite englobante qui permet de détecter les guides d'ondes proches de la zone de conflit afin de les remettre en question, le guide d'ondes n°1 dans notre exemple.

La boite englobante fonctionne de la façon suivante : un rectangle est placé autour de la zone de conflit afin de l'inclure entièrement, ensuite il est agrandi selon un paramètre, le **radius**. La figure 4.4(b) illustre cette opération, avec un radius égal à une case. Une fois positionnée, les cases qui y sont incluses sont examinées dans l'optique de détecter l'ordre d'apparition des guides d'ondes. La détection est effectuée grâce à un balayage de droite à gauche et de bas en haut. Ce dernier détermine l'ordre de routage des portions de guides d'ondes à réparer. Les portions sont définies par les points de départ et d'arrivée secondaires, positionnés à la frontière extérieure de la boite englobante. Le résultat de la réparation est illustré par la figure 4.4(c).



(a) (b) (c)

FIGURE 4.4 – Routage de trois guides d'ondes

La multiplication des guides d'ondes dont le routage doit être remis en question durant la réparation, implique la prise en compte d'un nouvel aspect concernant le positionnement des points de départ et d'arrivée secondaires. Un positionnement à la frontière de la boite englobante n'est pas toujours optimal, car il peut provoquer l'échec de le réparation. Un positionnement dit optimal est celui où les points de départ secondaires (les points d'arrivée secondaires) sont placés en retrait (en cascade) les uns par rapport aux autres. Appliqué à l'exemple précédent, cela donne le résultat illustré par la figure 4.5, mais aussi schématiquement par la figure 4.6. Le point d'arrivée secondaire du guide d'ondes n°1 est confondu avec le point d'arrivée réel. Un tel positionnement permet de limiter les situations de blocage lors de la reparation. Sur la figure 4.6, le positionnement des points secondaires du guide d'ondes n°1 lui permet d'être routé sans risquer de bloquer le guide d'ondes n°2, idem pour le n°2 et n°3.

4.3 Heuristique basée sur la génération de colonnes

Cette heuristique tente de construire de manière assez simpliste une solution entière à partir de la borne inférieure présentée en section 4.3 et obtenue par la génération de colonnes. Si la borne inférieure correspond à une solution réalisable, c'est-à-dire que les variables ont des valeurs égale à 1, l'heuristique retourne cette solution qui s'avère d'ailleurs être optimale. Dans le cas contraire, un processus séquentiel est utilisé pour converger vers une solution entière complete (tous les guides d'ondes sont routés) dans le meilleur des cas ou partielle (certain guides d'ondes n'ont pas été routés) dans le pire cas. Dans un premier temps, une variable fractionnaire est sélectionnée pour être fixée à 1. Après quoi, la génération de colonne est relancée pour calculer une nouvelle borne inférieure en prenant en compte les variables fixées. La variable sélectionnée n'est pas seule à être fixée un 1, toutes les variables fixées à 1 par le calcul de la borne inférieure sont elles aussi fixées à 1, de

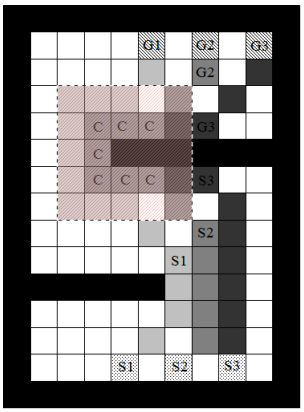


FIGURE 4.5 – Positionnement des points secondaires

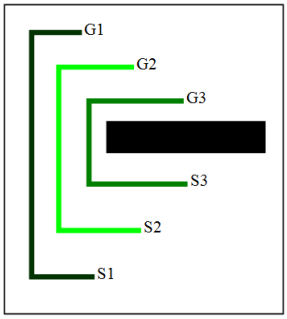


FIGURE 4.6 – Positionnement des points secondaires (schématiquement)

manière définitive. Le processus est répété jusqu'a ce que la borne inférieure obtenue corresponde à une solution entière ou qu'il n'existe pas de solution réalisable.

Fixer une variable à 1 signifie qu'on attribue une route $r_k \in \Omega^w$ au guide d'ondes $w \in W$. Soit $V_k \subset V$ l'ensemble des sommets $v \in V$ attribués à la route r_k et $D_k \subset D$ l'ensemble des points de croisement $d \in D$ attribués à la route r_k. Attribuer une route r_k au guide d'ondes w revient à supprimer de Ω_1 toutes les routes $r_{k'} \in \Omega_1$ vérifiant $\sum_{d \in D_k} \delta_{dk'} > 0$ ou $\sum_{v_i \in V_k} a_{ik'} > 0$. De plus, lors de la recherche de nouvelles colonnes, tout $d \in D_k$ et $v \in V_k$ ne sont plus accessible. Les routes r_k sont considérées comme des obstacles.

La sélection de la variable à fixer, lors de chaque itération est effectuée en deux temps. Tout d'abord, un guide d'ondes est sélectionné selon deux critères évalués de manière hiérarchique :

1. Plus courte distance estimée séparant la source et la destination du guide d'ondes. Ce critère permet de favoriser les guides d'ondes les plus courts.
2. Le moins de variables fractionnaires, afin de favoriser les guides d'ondes qui ont rencontré le moins de conflit.

Une fois le guide d'ondes sélectionné, la variable choisie est celle dont la valeur est maximale. En cas d'égalité, la variable de coût minimum est privilégiée.

Algorithme 2: Heuristique Génération de Colonnes

tantque La solution est fractionnaire **faire**
  GenerationDeColonnes(Ω_1) {Une solution est générée.}
  Fixer les variables de la solution dont la valeur est égale à 1.
  Sélectionner et fixer une variable. {Sélection selon les critères cités précédemment.}
fin tantque

4.4 Expérimentations

4.4.1 Description des instances de test

Les expérimentations sont effectuées sur les instances de test décrites dans la section 3.4 du chapitre 3.

Dans la suite du mémoire, nous notons, **H** la méthode heuristique gloutonne, **HGC** l'heuristique basée sur la génération de colonnes, **GC**, **A*** et **Eucl** respectivement les bornes inférieures donnée par la génération de colonnes, l'A* où l'on effectue le routage de chaque guide d'ondes sans considerer les routages précédents et la distance Euclidienne où l'on calcule la distance Euclidienne entre s_w et t_w pour chaque guide d'ondes.

4.4.2 Résultats

Les tableaux 4.1 et 4.3 regroupent les résultats des expérimentations des deux heuristiques. À chaque instance correspond un coût de la solution par heuristique, ainsi qu'un écart entre les deux heuristiques exprimé en pourcentage. Une instance non résolue (solution incomplete) a un coût associé égale à « - ». Parmi les 63 instances seulement 12 sont dans ce cas, avec au plus deux guides d'ondes non routés : deux pour les instance (30 30 6 1 9), (25 25 5 1 8) et (50 50 11 1 7), et un seul guide d'ondes pour les 9 restantes. De plus, la meilleure valeur obtenue est notée en gras. Les différents cas de figure cités précédemment sont représentés dans le tableau 4.1, alors que le tableau 4.3 regroupe des instances réelles. Notez que toutes les instances non résolues appartiennent au tableau 4.1.

Sur les 51 instances résolues par les deux heuristiques, on constate un écart moyen de 0,01% et au plus 4,73 % en faveur de la génération de colonnes. Plus particulièrement, on constate un écart moyen très faible, de 0,27% et au plus 1,98% pour les instances réelles. Au regard de ces premiers résultats, on peut affirmer que les deux heuristiques sont très proches et cela est confirmé par la proportion de meilleures valeurs obtenues. Les heuristiques gloutonne et génération de colonnes donnent la meilleure valeur respectivement pour 24 et 26 instances, sont à égalité pour 11 instances. Seulement deux instances n'ont pas été résolues par chaque heuristique. On peut néanmoins remarquer que l'heuristique génération de colonnes est meilleure pour 13 instances parmi les 20 instances réelles alors que l'heuristique gloutonne l'est pour 6, avec une seule égalité. Il faut tout de même relativiser ce dernier résultat au vu des écarts constatés entre les deux heuristique. Par contre, l'heuristique génération de colonnes est en grand difficulté pour traiter les instances de type croisement, une seule instance résolue sur 4 contre la totalité pour l'heuristique gloutonne.

La méthode de résolution basée sur la génération de colonnes, permet de calculer la relaxation linéaire du modèle de couverture et donc d'obtenir une borne inférieure de meilleur qualité que celles fournie par l'A* et la distance Euclidienne, mais dont il reste à évaluer la qualité face à la borne supérieure fournie par les deux heuristiques. Les tableaux 4.2 et 4.4 regroupent les écarts, en pourcentage de la meilleure borne supérieure trouvée (heuristiques gloutonne et génération de colonnes) et les trois bornes inférieures. On constate que pour la majorité des instances à un seul niveau, la borne supérieure est égale à la borne inférieure de la génération de colonnes, 25 instances parmi 33. On atteint donc la valeur optimale. Par contre, l'écart est en moyenne de 15,81% pour les instances de croisement. De plus, dès que le nombre de niveaux augmente, l'écart devient plus important, avec au plus 25,05% pour une instance à deux niveaux. On remarque aussi que l'écart est en moyenne de 7,06% et au plus 12,27% pour les instances réelles du tableau 4.2. Enfin, sur la totalité des instances on obtient une moyenne de 5,78% avec au plus un écart de 25,05%.

Les tableaux 4.5 et 4.6 illustrent les temps d'exécution (en seconde) de H et HGC. Sur l'ensemble des instances, H est nettement plus rapide que HGC. Néanmoins, cette rapidité ne lui confère pas un net avantage au vu des écarts en terme d'évaluation de longueur illustrés par les tableaux 4.3 et 4.2.

4.4.3 Conclusion

Dans ce chapitre nous avons présenté deux heuristiques, l'heuristique gloutonne associée à un mécanisme de réparation, notée **H** et une seconde heuristique gloutonne basée sur la Génération de colonnes, notée **HGC** . Au vu des résultats d'expérimentations, les deux heuristiques sont assez proches en terme de capacité d'évaluation des longueurs des guides d'ondes et de qualité des solutions proposées. Il faut noter néanmoins un net avantage en terme de temps d'execution pour l'heuristique **H** face à **HGC**, même si cet avantage ne lui permet pas de prendre l'ascendant sur l'heuristique **HGC**. Cependant, du fait de l'incapacité de **HGC** à résoudre les instances de croisement, il nous semble plus judicieux de privilégier l'heuristique gloutonne associée à un mécanisme de réparation.

Nous avons également proposé une méthode d'évaluation d'une borne inférieure, basée sur la generation de colonnes. Cette borne inférieure nous a permis d'évaluer la qualité des deux bornes supérieures fournies par les deux heuristiques.

Le modèle de routage traité dans ce chapitre est un modèle assez simple et non réaliste. Il ne prend pas en compte, par exemple l'épaisseur des guides d'ondes. Dans le chapitre 5 nous proposons une nouvelle heuristique de réparation exploitant un modèle de routage plus réaliste. De plus, nous proposons également un algorithme A* grand voisinage permettant d'affiner les trajectoires et d'obtenir ainsi une évaluation des longueurs plus précise. Il sera en tout point identique à l'algorithme A*, excepté pour la phase d'expansion où le grand voisinage lui permettra d'atteindre des cases non adjacentes à la case à l'origine de l'expansion. Un tel mécanisme est inspiré par les travaux de Watanabe et Sugiyama [163], qui ont proposé une adaptation de l'algorithme de Lee, avec comme paramètre supplémentaire la taille de la vague d'expansion.

Nous proposons également une méthode de réparation globale, en opposition à la méthode de réparation locale décrite dans ce chapitre.

(X,Y,Z)	Nwg	Occ	H	HGC	Écart
			valeur	valeur	(%)
Croisement					
(5,5,1)	2	0	**10,83**	**10,83**	0,00
(10,10,1)	2	0	**29,31**	-	-
(30,30,1)	2	0	**97,60**	-	-
(50,50,1)	2	0	**165,88**	-	-
Goulet d'étranglement					
(15,15,1)	4	4	**65,11**	**65,11**	0,00
(15,15,1)	4	9	**66,53**	**66,53**	0,00
(15,15,1)	5	22	**85,67**	**85,67**	0,00
(30,30,1)	5	8	169,41	**163,95**	3,22
(30,30,1)	6	9	-	-	-
(30,30,1)	7	5	-	**225,21**	-
(30,30,1)	7	7	218,58	**217,17**	0,65
(50,50,1)	11	3	596,78	**593,26**	0,59
(50,50,1)	11	7	**623,00**	-	-
(50,50,2)	16	16	**390,75**	408,75	-4,61
Virage en S					
(15,20,1)	2	6	53,77	**53,18**	1,09
(15,20,1)	3	6	**77,08**	**77,08**	0,0
(15,20,1)	4	6	99,40	**98,23**	1,18
(30,30,1)	2	3	**85,98**	**85,98**	0,0
(30,30,1)	3	3	**124,37**	-	-
(30,30,1)	4	3	156,65	**156,07**	0,37
(30,30,1)	5	3	-	**210,25**	-
(50,50,1)	2	3	**189,62**	193,62	-2,11
(50,50,1)	3	3	**267,78**	**279,78**	-4,48
(50,50,1)	4	3	**344,37**	-	-
(50,50,1)	5	3	-	-	-
Croisement / Goulet d'étranglement / Virage en S					
(10,10,1)	2	56	**27,90**	**27,90**	0,0
(10,10,1)	3	18	**27,31**	**27,31**	0,0
(10,10,1)	4	22	**36,97**	**36,97**	0,0
(10,10,2)	6	42	**46,83**	**46,83**	0,0
(25,25,1)	5	11	**141,81**	142,40	-0,42
(25,25,1)	6	14	-	**206,02**	-
(25,25,1)	5	8	-	**256,92**	-
(25,25,1)	6	17	**175,54**	177,88	-1,33
(30,30,2)	15	19	241,81	**240,98**	0,34
(30,30,2)	23	24	**494,94**	497,87	-0,59
(30,30,2)	28	22	**342,98**	343,81	-0,24
(50,50,2)	14	34	505,28	**502,84**	0,48
(50,50,2)	31	20	**677,43**	690,16	-1,88
(50,50,2)	38	24	1030,50	**981,77**	4,73
(200,120,1)	42	28	**1388,50**	1394,74	-0,45
(200,120,1)	42	30	**1422,07**	1426,56	-0,32
(100,100,1)	23	14	**836,74**	844,99	-0,99
(100,100,1)	27	18	**1028,85**	1032,13	-0,32

TABLE 4.1 – Résultats d'expérimentation

(Eucl,A*,GC) vs Best(H,HGC) : écart en %					
(X,Y,Z)	Nwg	Occ	Eucl	A*	GC
Croisement					
(5,5,1)	2	0	20,50	16,25	13,54
(10,10,1)	2	0	17,98	17,98	15,98
(30,30,1)	2	0	17,41	17,41	16,81
(50,50,1)	2	0	17,30	17,30	16,95
Goulet d'étranglement					
(15,15,1)	4	4	13,88	2,53	0,00
(15,15,1)	4	9	15,72	1,25	0,00
(15,15,1)	5	22	23,68	7,28	0,00
(30,30,1)	5	8	17,57	4,31	0,00
(30,30,1)	6	9	-	-	-
(30,30,1)	7	5	15,99	2,94	0,00
(30,30,1)	7	7	12,88	2,29	0,00
(50,50,1)	11	3	9,11	1,26	0,00
(50,50,1)	11	7	13,45	2,71	0,79
(50,50,2)	16	16	21,58	11,69	10,45
Virage en S					
(15,20,1)	2	6	25,60	3,11	0,00
(15,20,1)	3	6	26,05	6,58	0,00
(15,20,1)	4	6	22,63	3,38	0,00
(30,30,1)	2	3	26,99	2,61	0,00
(30,30,1)	3	3	28,04	5,03	0,00
(30,30,1)	4	3	23,54	5,16	0,00
(30,30,1)	5	3	27,79	8,93	0,00
(50,50,1)	2	3	49,38	5,17	2,07
(50,50,1)	3	3	45,84	6,07	0,00
(50,50,1)	4	3	43,66	8,92	0,00
(50,50,1)	5	3	-	-	-
Croisement / Goulet d'étranglement / Virage en S					
(10,10,1)	2	56	38,39	2,97	0,00
(10,10,1)	3	18	15,42	0,88	0,00
(10,10,1)	4	22	12,44	2,25	0,00
(10,10,2)	6	42	44,22	42,73	24,69
(25,25,1)	5	11	15,19	4,33	0,00
(25,25,1)	6	14	30,02	5,78	0,00
(25,25,1)	5	8	53,19	16,65	0,00
(25,25,1)	6	17	17,87	0,95	0,00
(30,30,2)	15	19	30,68	21,24	18,22
(30,30,2)	23	24	36,32	29,39	25,05
(30,30,2)	28	22	30,43	23,92	19,70
(50,50,2)	14	34	33,79	2,36	2,36
(50,50,2)	31	20	29,23	21,71	20,27
(50,50,2)	38	24	32,98	26,34	23,63
(200,120,1)	42	28	15,81	1,47	0,27
(200,120,1)	42	30	17,79	1,72	0,39
(100,100,1)	23	14	11,21	1,04	0,00
(100,100,1)	27	18	14,44	2,83	0,36

TABLE 4.2 – Écart entre les bornes inférieures et la borne supérieure

(X,Y,Z)	Nwg	Occ	H	HGC	Écart
			valeur	valeur	(%)
(215,140,3)	80	24	2151,44	**2125,10**	1,22
(215,140,3)	37	21	867,59	**865,83**	0,20
(215,140,3)	138	34	4169,94	**4118,97**	1,22
(215,140,3)	112	31	2510,94	**2490,41**	0,82
(215,140,3)	140	34	3704,94	**3667,47**	1,01
(215,140,3)	96	27	2677,40	**2662,34**	0,56
(250,230,2)	13	7	**1084,15**	**1084,15**	0,0
(250,230,2)	48	2	**3173,21**	3183,22	-0,32
(250,230,3)	91	23	6060,94	**5977,23**	1,38
(300,200,3)	37	18	1201,11	**1190,08**	0,92
(300,200,3)	74	22	**1903,91**	1911,87	-0,42
(300,200,3)	74	23	2032,78	**2031,41**	0,07
(300,200,3)	80	21	**2939,40**	2953,79	-0,49
(300,200,3)	96	25	**3814,88**	3845,19	-0,79
(300,200,3)	112	28	**3390,69**	3390,94	-0,01
(300,200,3)	123	27	**5329,88**	5435,38	-1,98
(300,200,3)	138	32	5754,57	**5744,47**	0,18
(300,200,3)	140	32	5038,29	**4993,47**	0,89
(300,200,3)	148	30	6477,95	**6469,64**	0,13
(300,200,3)	168	35	7452,66	**7386,49**	0,89

TABLE 4.3 – Résultats d'expérimentation : instances réelles

(Eucl,A*,GC) vs Best(H,HGC) : écart en %					
(X,Y,Z)	Nwg	Occ	Eucl	A*	GC
(215,140,3)	80	24	29,58	12,42	9,98
(215,140,3)	37	21	28,79	4,98	4,06
(215,140,3)	138	34	25,05	10,88	8,64
(215,140,3)	112	31	31,16	12,39	9,34
(215,140,3)	140	34	29,79	14,67	12,27
(215,140,3)	96	27	20,06	8,41	7,78
(250,230,2)	13	7	10,36	4,85	4,85
(250,230,2)	48	2	7,59	3,16	3,07
(250,230,3)	91	23	15,03	6,13	5,79
(300,200,3)	37	18	22,37	3,53	3,05
(300,200,3)	74	22	20,22	7,21	6,92
(300,200,3)	74	23	21,17	7,94	7,44
(300,200,3)	80	21	22,83	8,92	8,26
(300,200,3)	96	25	16,07	6,28	5,93
(300,200,3)	112	28	24,10	8,83	8,07
(300,200,3)	123	27	19,84	7,48	2,27
(300,200,3)	138	32	19,19	7,50	6,99
(300,200,3)	140	32	22,40	9,81	9,20
(300,200,3)	148	30	21,73	9,80	9,43
(300,200,3)	168	35	21,69	8,35	7,89

TABLE 4.4 – Écart entre les bornes inférieures et la borne supérieure : instances réelles

Temps d'exécution (en seconde) : H vs HGC				
(X,Y,Z)	Nwg	Occ	H	HGC
Croisement				
(5,5,1)	2	0	< 1	< 1
(10,10,1)	2	0	< 1	< 1
(30,30,1)	2	0	< 1	< 1
(50,50,1)	2	0	< 1	< 1
Goulet d'étranglement				
(15,15,1)	4	4	< 1	< 1
(15,15,1)	4	9	< 1	< 1
(15,15,1)	5	22	< 1	< 1
(30,30,1)	5	8	< 1	< 1
(30,30,1)	6	9	< 1	3,70
(30,30,1)	7	5	< 1	< 1
(30,30,1)	7	7	< 1	< 1
(50,50,1)	11	3	< 1	< 1
(50,50,1)	11	7	< 1	1,20
(50,50,2)	16	16	< 1	< 1
Virage en S				
(15,20,1)	2	6	< 1	< 1
(15,20,1)	3	6	< 1	< 1
(15,20,1)	4	6	< 1	< 1
(30,30,1)	2	3	< 1	< 1
(30,30,1)	3	3	< 1	< 1
(30,30,1)	4	3	< 1	< 1
(30,30,1)	5	3	< 1	2,20
(50,50,1)	2	3	< 1	0,69
(50,50,1)	3	3	< 1	3,30
(50,50,1)	4	3	< 1	9,90
(50,50,1)	5	3	< 1	35,60
Croisement / Goulet d'étranglement / Virage en S				
(10,10,1)	2	56	< 1	< 1
(10,10,1)	3	18	< 1	< 1
(10,10,1)	4	22	< 1	< 1
(10,10,2)	6	42	< 1	< 1
(25,25,1)	5	11	< 1	< 1
(25,25,1)	6	14	< 1	< 1
(25,25,1)	5	8	< 1	< 1
(25,25,1)	6	17	< 1	< 1
(30,30,2)	15	19	< 1	< 1
(30,30,2)	23	24	< 1	2,42
(30,30,2)	28	22	< 1	< 1
(50,50,2)	14	34	< 1	< 1
(50,50,2)	31	20	< 1	1,94
(50,50,2)	38	24	< 1	13,50
(200,120,1)	42	28	< 1	< 1
(200,120,1)	42	30	< 1	< 1
(100,100,1)	23	14	< 1	< 1
(100,100,1)	27	18	< 1	5,00

TABLE 4.5 – Temps d'exécution de H et HGC

Temps d'exécution (en seconde) : H vs HGC				
(X,Y,Z)	Nwg	Occ	H	HGC
(215,140,3)	80	24	**3,00**	13,15
(215,140,3)	37	21	**1,00**	1,17
(215,140,3)	138	34	**46,00**	49,15
(215,140,3)	112	31	**2,00**	7,41
(215,140,3)	140	34	**30,00**	55,70
(215,140,3)	96	27	**2,00**	16,53
(250,230,2)	13	7	**1,00**	1,12
(250,230,2)	48	2	**3,00**	3,72
(250,230,3)	91	23	**7,00**	288,30
(300,200,3)	37	18	**2,00**	4,90
(300,200,3)	74	22	**3,00**	3,05
(300,200,3)	74	23	**5,00**	3,60
(300,200,3)	80	21	**7,00**	23,10
(300,200,3)	96	25	**3,00**	22,39
(300,200,3)	112	28	**3,00**	20,30
(300,200,3)	123	27	**5,00**	167,30
(300,200,3)	138	32	**4,00**	62,10
(300,200,3)	140	32	**7,00**	93,81
(300,200,3)	148	30	**7,00**	460,48
(300,200,3)	168	35	**8,00**	422,06

TABLE 4.6 – Temps d'exécution de H et HGC : Instances réelles

CHAPITRE 5

MODÈLE RÉALISTE

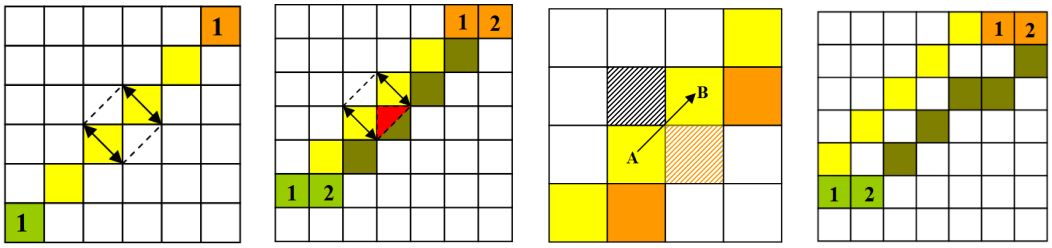
Jusqu'à présent, une version simplifiée de la problématique a été considérée, cela afin de mieux l'appréhender. Seules les contraintes interdisant le chevauchement et le croisement entre guides d'ondes étaient prises en compte. L'inconvénient d'une telle simplification réside dans le fait que le routage obtenu n'est pas dimensionnant. Dans ce chapitre, nous proposons un modèle réaliste avec l'ajout de nouvelles contraintes. Le modèle réaliste sera développé exclusivement pour une méthode de résolution approchée de type de glouton, couplée à un mécanisme de réparation. Ce choix a été motivé par les résultats obtenus précédemment et par la difficulté que représente une modélisation des nouvelles contraintes sous forme de PLNE.

5.1 Constraints supplémentaires

Un routage réaliste nécessite la prise en compte de contraintes supplémentaires. Un guide d'ondes possède une épaisseur propre qui depend de l'orientation des connecteurs terminaux (voir section 1.2), et une limitation angulaire lors des changement de direction (coude). De plus une distance minimale doit être respectée entre un guide d'ondes et un composant, entre deux guides d'ondes pour cause de contrainte liée à la dissipation calorifique, et avant tout changement de direction, notamment pour les portions de départ et d'arrivée (voir section 1.2). les portions de départ et d'arrivée permettent de conserver un libre accès aux vis de montage/démontage d'un guide d'ondes. Les contraintes citées ci-dessus garantissent un niveau de réalisme suffisant durant le routage. Ansi, les estimations de longueur obtenues se rapproche des longueurs réelles.

Comme indiqué dans la section 3.1, l'espace de routage est discrétisé sous forme de grille de deux dimensions sur plusieurs niveaux, le passage d'un niveau à l'autre ne pouvant se faire qu'entre cases de mêmes coordonnées. En dehors des limites de cette modélisation liée à la discrétisation de l'espace, une difficulté provient de l'impossibilité physique pour deux guides d'ondes de se chevaucher durant le routage. Un cas de chevauchement peut être de deux types : chevauchement explicite et chevauchement implicite. Un chevauchement explicite est défini par l'occupation d'une case par plusieurs guides d'ondes. En revanche, dans le cas implicite, le chevauchement ne peut être défini par l'occupation d'une case. Un chevauchement implicite n'est rencontré que dans le cas d'un déplacement en diagonal. Soit l'exemple illustré par la figure 5.1(a) où l'on observe le guide d'ondes n°1 qui suit une route diagonale. L'épaisseur du guide d'ondes n°1 est représentée par la zone délimitée par les deux segments en pointillé. La figure 5.1(b) représente deux guides d'ondes routés côte-à-côte et suivant tous deux une route diagonale. Un chevauchement implicite est alors illustré par le triangle rouge qui indique que les épaisseurs des deux guides d'ondes ne sont pas respectées du fait de l'imbriquement des deux guides d'ondes. La prise en compte de cette contrainte est alors effectuée en interdisant l'utilisation des deux cases hachurées de la figure 5.1(c), ceci afin de respecter l'épaisseur des guides d'ondes lors d'un déplacement en diagonal. L'impact

de la prise en compte de cette contrainte est illustré par la figure 5.1(d).



(a) (b) (c) (d)

FIGURE 5.1 – Respect de l'épaisseur des guide d'ondes (1)

Le modèle réaliste impose la prise en compte de l'épaisseur d'un guide d'ondes lors du routage. Sur le plan, l'épaisseur d'un guide d'ondes dépend de l'orientation des ports associés. La figure 5.2 illustre une coupe d'un guide d'ondes. Cette coupe permet de visualiser les deux dimensions qui nous interessent, les dimensions a et b, avec $a > b$. La dimension b est la valeur de l'épaisseur d'un guide d'ondes si les deux connecteurs associés ont une orientation de 0°, avec une orientation de 90° l'épaisseur est égale à a. Ainsi, avec une orientation de 0° un guide d'ondes a un encombrement plus important du fait d'une épaisseur plus grande qu'avec une orientation de 90°.

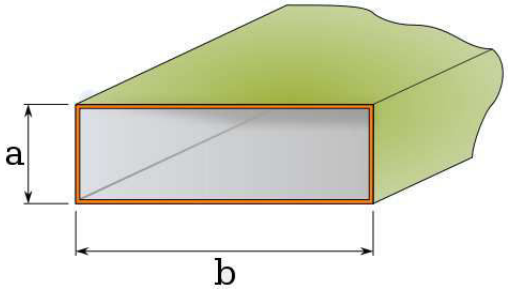
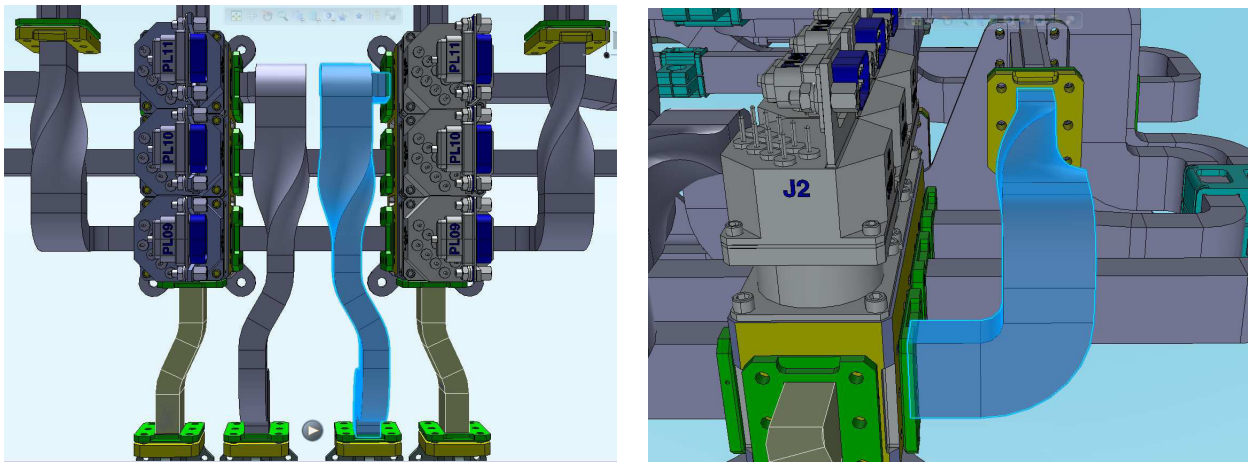


FIGURE 5.2 – Épaisseur d'un guide d'ondes

De plus, indépendamment de l'orientation de ses connecteurs, un guide d'ondes peut avoir des sections d'une épaisseur égale à a et des section d'une épaisseur égale à b. Ce changement d'épaisseur est dû soit à un changement de direction qui implique un changement d'orientation ou à l'utilisation d'un élément nommé twist qui permet de modifier l'orientation d'un guide d'ondes soit afin de réduire l'encombrement de ce dernier, soit en vue d'un changement de direction nécessitant un changement d'orientation ou dans le but d'avoir une orientation compatible avec celle de son connecteur. La figure 5.3 illustre l'utilisation d'un twist, section du guide d'ondes présentant une torsion à 90°. La figure 5.4(a) représente un changement de direction qui a impliqué un changement d'orientation du guide d'ondes. Ce changement d'orientation impact l'épaisseur du guide d'ondes sur la section concernée. La figure 5.4(b) illustre le fait qu'un guide d'ondes peut être composé de sections de différentes épaisseur.

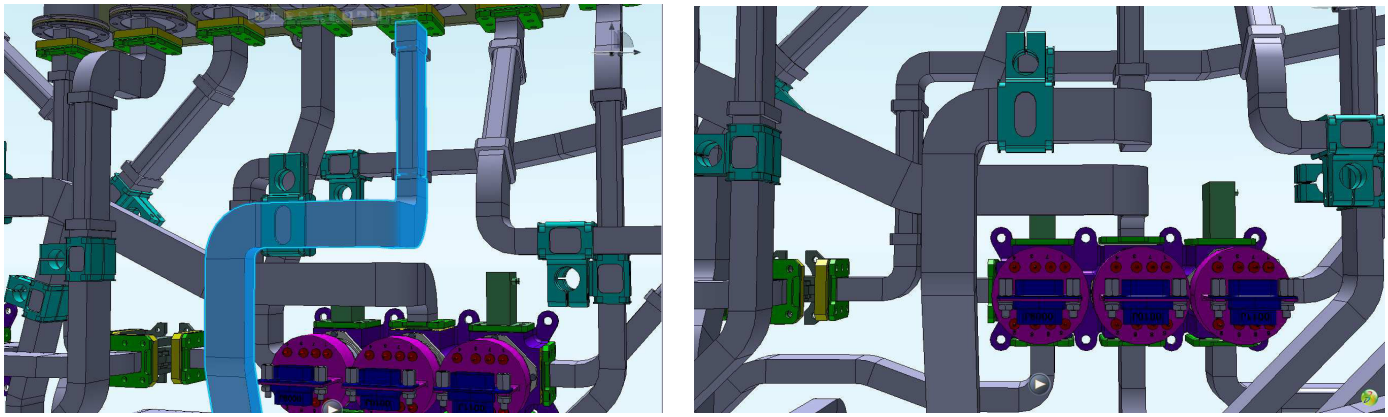
Notre modèle réaliste prend bien en compte l'épaisseur d'un guide d'ondes mais ne permet à un guide d'ondes de n'avoir qu'une unique épaisseur. Les changements d'épaisseur duent aux changements de direction ou à l'utilisation d'un twist ne sont pas considérés. Lorsque les connecteurs associés à un même guide d'ondes ont la même orientation, l'épaisseur est sélectionnée selon cette orientation. Dans le cas d'orientations différentes, l'épaisseur sélectionnée est la plus dimensionnante, c'est-à-dire la plus grande, ceci pour être conservatif.

L'épaisseur d'un guide d'ondes est prise en compte lors d'une extension en diagonal, horizontal, vertical ou lors d'un changement de niveau. Soit la figure 5.5(a) qui illustre les 8 directions de déplacement lors du routage. Afin de modéliser l'épaisseur d'un guide d'ondes, une position n'est plus assimilée à une case mais à un ensemble de



(a) (b)

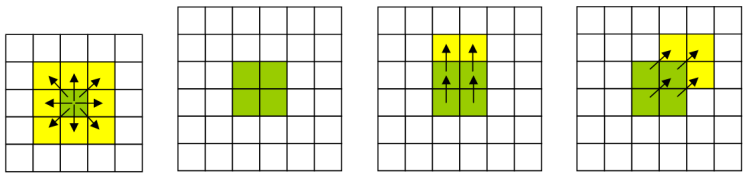
FIGURE 5.3 – Twist



(a) (b)

FIGURE 5.4 – Changement de direction impliquant un changement d'épaisseur

cases comme illustré par la figure 5.5(b), où la position représentée appartient à un guide d'ondes d'épaisseur équivalente à deux cases. Cet ensemble de cases forme un carré dont la longueur d'un coté est égale à l'épaisseur du guide d'ondes. Les figures 5.5(c) et 5.5(d) illustrent respectivement un déplacement vertical et un déplacement diagonal d'un tel ensemble.



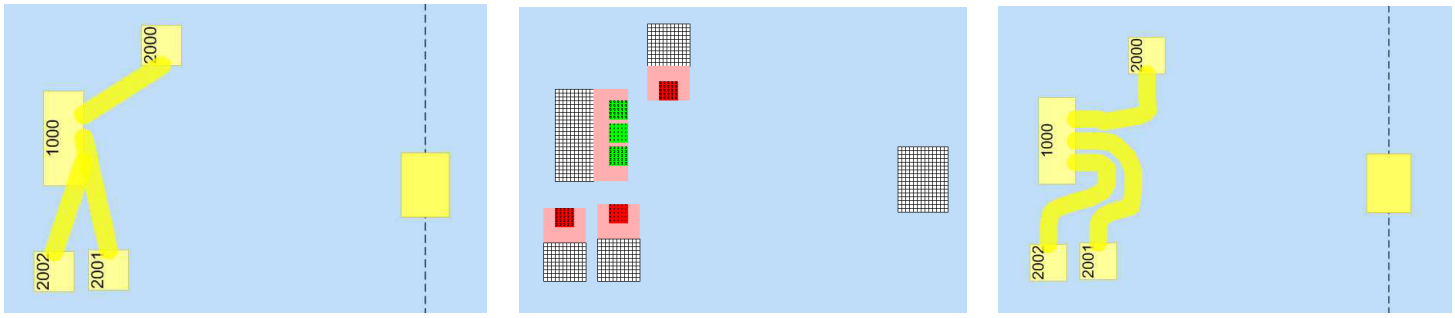
(a) (b) (c) (d)

FIGURE 5.5 – Respect de l'épaisseur des guide d'ondes (2)

Lors du routage, il est nécessaire de conserver une distance minimale entre les guides d'ondes. Cette distance est égale à $5mm$ et est directement comptabilisée dans l'épaisseur des guides d'ondes à router. Afin de respecter une contrainte de démontabilité garantissant un libre accès au vis de montage/démontage des guides d'ondes, des portions de depart et d'arrivée doivent être routées pour chaque guide d'ondes (voir section 1.2). La portion de départ (respectivement d'arrivée) est un segment qui a pour origine le connecteur source (respectivement

destination) du guide d'ondes et qui est perpendiculaire au composant contenant ce connecteur. La longueur du segment est de $16mm$. Cette longueur est une valeur préconisée par les concepteurs, mais néanmoins elle peu être réduite si nécessaire.

Les figures 5.6(a-c) illustrent ces contraintes. La figure 5.6(a) représente une zone de routage contenant 4 composants et 3 guides d'ondes représentés, pour l'instant par un segment reliant chaque couple (origine-destination). Cette zone de routage est dans un domaine continu. La figure 5.6(b) illustre la discrétisation de cette zone en une grille de largeur de maille égale à $4mm$. Les guides d'ondes ont une épaisseur équivalente à 5 cases. Les couples (origine-destination) sont représenté par, respectivement, un ensemble de cases vertes et un ensemble de cases rouges. La distance qui sépare chaque connecteur du composant associé représente une portion de depart pour un connecteur origine et une portion d'arrivée pour un connecteur destination. Afin de protéger la sortie et l'arrivée d'un guide d'ondes, une zone de surcoût est positionnée sur le côté des composants contenant le connecteur d'origine et le connecteur de destination. Cette zone est illustrée par la figure 5.6(b) par une zone de couleur rose. La figure 5.6(c) représente le résultat du routage, sur un seul niveau.



(a) (b) (c)

FIGURE 5.6 – Respect des portions de départ et d'arrivée

La figure 5.7 illustre un routage plus dense et sur 4 niveaux. La couleur jaune indique le premier niveau, les couleurs orange, rouge et marron indiquent respectivement le second, le troisième et le quatrième niveau. Cet extrait de routage est un routage sur un mur de satellite réel. Il nécessite 4 niveaux au lieu de 3 niveaux habituellement. On y observe les portions de départ et d'arrivée et la prise en compte de l'épaisseur des guides.

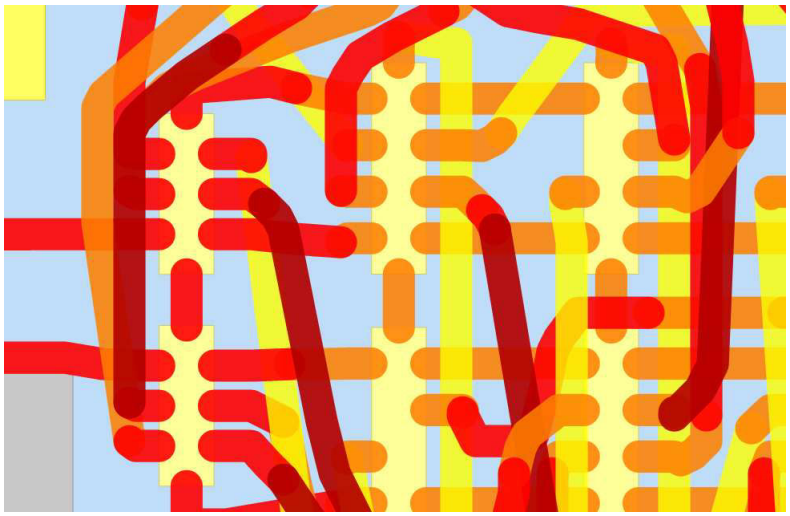
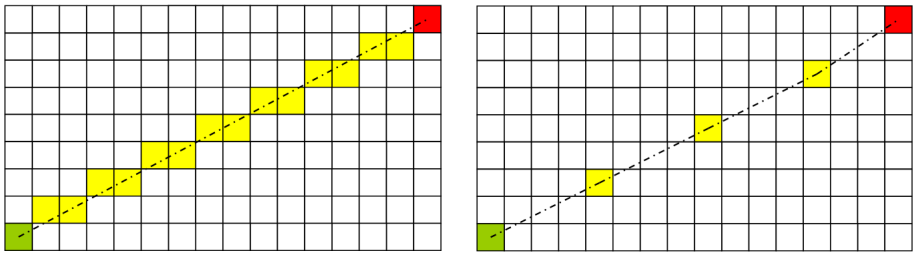


FIGURE 5.7 – Portion de départ et d'arrivée

L'heuristique gloutonne est basée sur l'algorithme de recherche de plus court chemin **A*** dont le principal inconvénient est la surestimation, dans certain cas, de la longueur des guides d'ondes lors du passage d'un

espace continu à un espace discret et inversement. Cette surestimation provient de la limitation à 8 déplacements possibles, sur un même plan, lors de la phase d'expansion d'un chemin. Soit l'exemple illustré par la figure 5.8(a) qui représente le plus court chemin d'un guide d'ondes obtenue après une succession de phases d'expansion limitées à 8 directions possibles par phase, sur un même plan. Il en résulte un plus court chemin comprenant 8 déplacements diagonaux et 7 déplacements horizontaux pour une longueur totale de 18.94 unités de longueur (une unité de longueur correspond à la longueur d'une case). Or, la longueur optimale du guide d'ondes dans un espace continue correspond à la distance Euclidienne séparant le centre des deux extrémités, ce qui est égal à $16,94$ unités de longueur, soit une surestimation d'environs 8%. Cette surestimation est accentuée par le fait que notre exemple est un déplacement en diagonal. Dans la suite du mémoire, une telle expansion, limitée à 8 directions, est appelé **simple voisinage**.



(a) (b)

FIGURE 5.8 – Simple voisinge

Afin de limiter cette surestimation nous introduisons une recherche de plus court chemin basée sur une phase d'expansion **grand voisinage**.

5.2 Grand voisinage

Pour une case donnée, le simple voisinage limite le nombre de direction d'expansion à 8 directions, ce qui correspond aux 8 cases adjacentes, comme illustré par la figure 5.5(a). Lors d'une phase d'expansion grand voisinage, une case peut atteindre 8 directions pour un voisinage de taille 1, 16 directions supplémentaires pour un voisinage de taille 2 (i.e, 24 directions), et $m = \sum_{i=1}^{n} 8 * i$ directions pour un voisinage de taille n. Ainsi, une expansion grand voisinage permet d'atteindre des cases non adjacentes à la case de départ, dont la plus lointaine se trouve à $n * \sqrt{2}$ pour un voisinage de taille n.

La figure 5.9 illustre l'évolution du nombre de directions possibles en fonction de la taille n du voisinage. Pour un voisinage de taille $n = 1$ les cases accessibles sont celles labélisées d'un 1. Pour un voisinage de taille 2, les cases atteignables sont celles marquées d'un 1 ou d'un 2. Ainsi, pour un voisinage de taille n, les cases atteignables sont celles labélisées de 1 jusqu'à n.

La figure 5.10 illustre l'évolution de la longueur totale pour un ensemble de 100 guides d'ondes pour une taille de voisinage allant de 1 à 9. Une premiere remarque concerne la surestimation de la longueur totale en voisinage de taille 1 par rapport à un voisinage de taille 2. Cette surestimation est de 2.31%. Du voisinage de taille 2 au voisinage de taille 4 on observe une légère baisse de la longueur totale. À partir du voisinage de taille 5 l'écart devient négligeable.

Afin d'exploiter le grand voisinage, il est nécessaire d'identifier, pour chaque direction, les cases survolées par celle-ci, c'est-à-dire les cases qu'occupent ces directions. Pour ce faire, nous introduisons la notion de **masque**. Un masque regroupe l'ensemble des cases survolées par une direction, ainsi que les cases permettant de respecter l'épaisseur d'un guide d'ondes lors d'un déplacement en diagonal. La figure 5.11 illustre le masque correspondant à une direction verticale et une direction diagonale pour des guides d'ondes d'épaisseur une et deux cases. Les

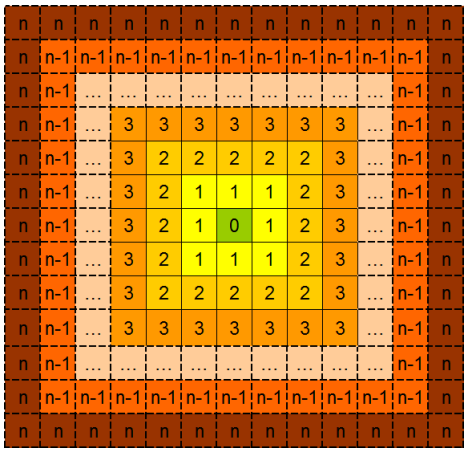


FIGURE 5.9 – Grand voisinage

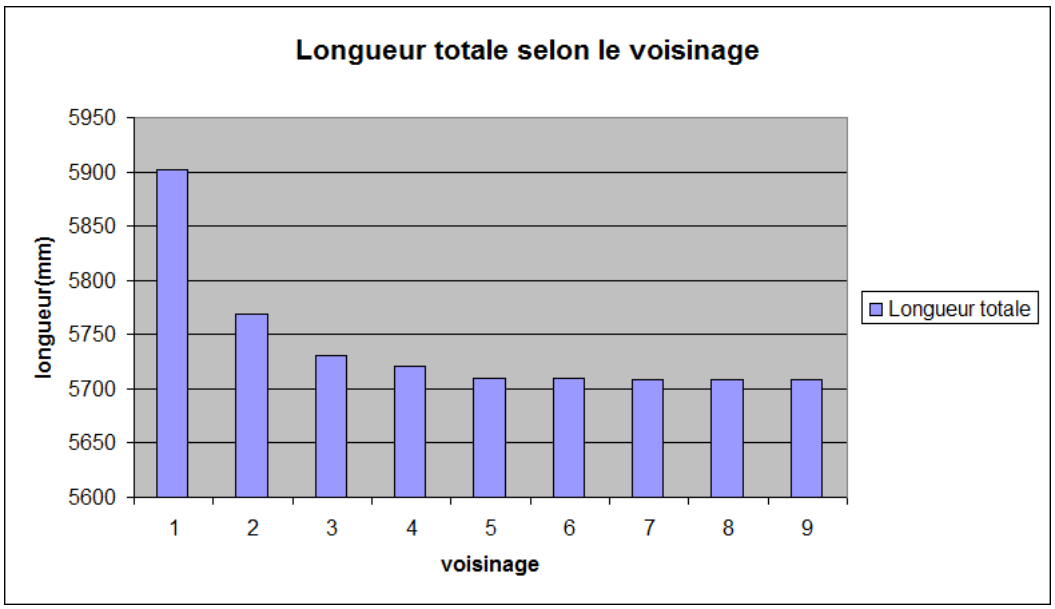
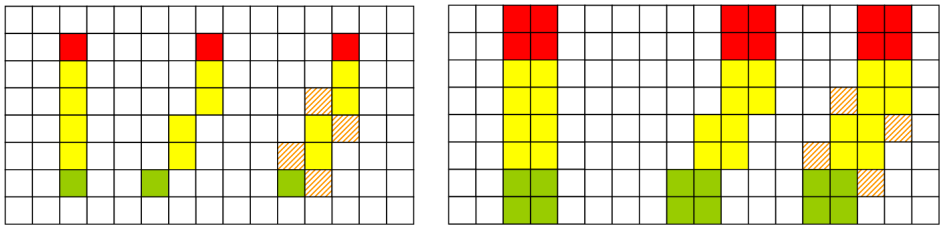


FIGURE 5.10 – Longueur totale selon le voisinage

directions relient leurs extrémités respectives par un ensemble de cases colorées en jaune. Cet ensemble représente l'ensemble des cases survolées par une direction. De plus, un masque, associé à une direction diagonale, contient également les cases qui doivent être interdites afin d'éviter un chevauchement implicite entre guides d'ondes. Ces cases sont représentées par les cases hachurées.



(a) (b)

FIGURE 5.11 – Masque

Ainsi, un masque est associé à chaque direction du grand voisinage. Donc, pour un voisinage de taille n, m

masques sont nécessaires. De plus, à chaque épaisseur de guide d'ondes m masques doivent lui être associés. Tout ceci soulève la question de la façon de générer ces masques. Un masque étant associé à une direction, il est défini par deux extrémités qui représentent les points d'origine et destination d'une direction. L'ensemble des cases survolées est alors déterminé en calculant le plus court chemin reliant ces deux extrémités. Ce calcul est effectué grâce à l'algorithme **A*** avec une nouvelle fonction de coût qui intègre un coût supplémentaire représentant l'écart angulaire entre deux directions : la direction définie par la droite reliant les points de départ et d'arrivée (respectivement case verte et rouge sur la figure 5.12), et une seconde direction définie par la droite reliant le point à étendre et le point d'arrivée (respectivement case jaune et rouge sur la figure 5.12).

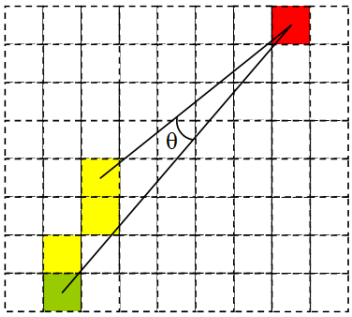


FIGURE 5.12 – Écart angulaire

Le grand voisinage permet de limiter la surestimation des longueurs des guides d'ondes en proposant un routage plus fin que le simple voisinage du fait d'un voisinage beaucoup plus étendu, et il facilite la prise en compte de certaines contraintes, notamment la contrainte limitant l'écart angulaire entre deux changements de direction et imposant une distance minimale avant tout changement de direction. La prise en compte de cette contrainte nécessite de connaître l'angle entre deux directions successives et la distance depuis le dernier changement de direction. Il suffit alors de ne prendre en compte que les directions dont la longueur est au moins égale à la distance minimale et dont la valeur absolue de l'angle formé avec la précédente direction est inférieur à l'angle limite. En pratique, la distance minimale avant tout changement de direction est un multiple de l'épaisseur du guide d'ondes et l'angle limite est égal à 90°.

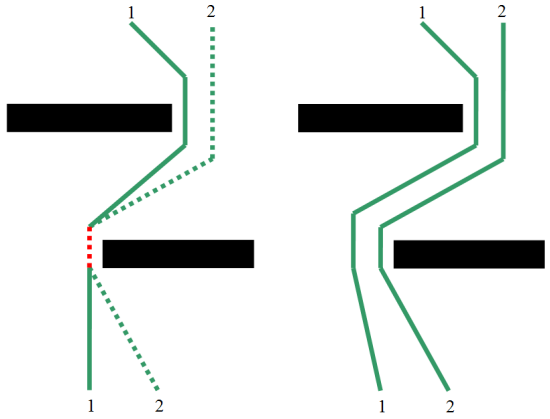
Comme dit précédemment, la méthode de résolution choisie pour résoudre le modèle réaliste de routage est inspirée de la méthode de résolution approchée de type gloutonne avec réparation décrite dans le chapitre 4. Néanmoins, les mécanismes de réparation décrits ne peuvent être appliqués au grand voisinage. De nouveaux mécanismes de réparation sont définis.

5.3 Réparation Grand voisinage

La nouvelle méthode de résolution grand voisinage adopte le même principe que la méthode de résolution simple voisinage. Comme décrit dans le chapitre 4, un guide d'ondes peut être sujet à plusieurs conflits. La fonction de réparation simple voisinage traite ces zones de conflit successivement et de manière indépendante. La nouvelle fonction de réparation traite ces conflits en une seule passe. Cette nouvelle fonction est basée sur un principe simple, le guide qui a provoqué le conflit (le dernier guide d'ondes routé) est prioritaire sur les zones concernées par un conflit. Cela signifie que le ou les guides d'ondes occupant ces zones doivent être déroutés afin de céder ces zones au dernier guide d'ondes routé.

Dans un premier temps nous allons illustrer ce nouveau mécanisme de réparation par l'exemple d'un routage de deux guides d'ondes. Soit la figure 5.13(a) qui illustre le routage de deux guides d'ondes, le n°1 routé en premier, et le n°2 routé en second. Le guide d'ondes n°2 est en conflit avec le n°1. La situation de conflit est illustrée par le segment rouge (en pointillés). Le mécanisme de réparation consiste alors à donner la priorité

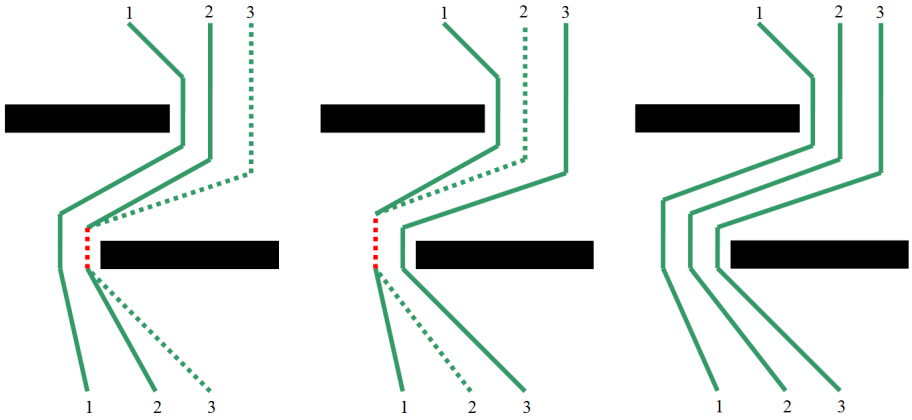
au dernier guide d'ondes routé, c'est-à-dire le n°2. Pour ce faire, le guide d'ondes n°2 conserve sa route initiale, illustré par le passage au trait plein dans la figure 5.13(b), et le guide d'ondes n°1 est dérouté. On obtient alors le résultat illustré par la figure 5.13(b).



(a) (b)

FIGURE 5.13 – Réparation grand voisinage : deux guides d'ondes

Contrairement au mécanisme de réparation simple voisinage, il n'est pas nécessaire pour le mécanisme de réparation grand voisinage d'utiliser une boite englobante (section 4.2) permettant de détecter les guides d'ondes concernés par une situation de conflit lorsque le nombre de guides d'ondes est supérieur à deux. La figure 5.14(a) illustre le routage de trois guides d'ondes. Le routage des guides d'ondes n°1 et n°2 est le résultat d'une premiere phase de réparation. Le guide d'ondes n°3, le dernier routé, est en conflit avec le guide d'ondes n°2. Comme précédemment, le guide d'ondes n°3 conserve sa route initiale, illustré par le passage au trait plein dans la figure 5.14(b), et le guide d'ondes n°2 est dérouté, mais sans avoir le droit d'entrer en conflit avec le guide d'ondes n°3. On obtient alors le résultat illustré par la figure 5.14(b) où le guide d'ondes n°2 est en conflit avec le n°1. De la même manière, le guide d'ondes n°2 conserve sa route initiale, illustré par le passage au trait plein dans la figure 5.14(c), et le guide d'ondes n°1 est dérouté, mais sans avoir le droit d'entrer en conflit avec les guides d'ondes n°2 et n°3. On obtient le résultat illustré par la figure 5.14(c).



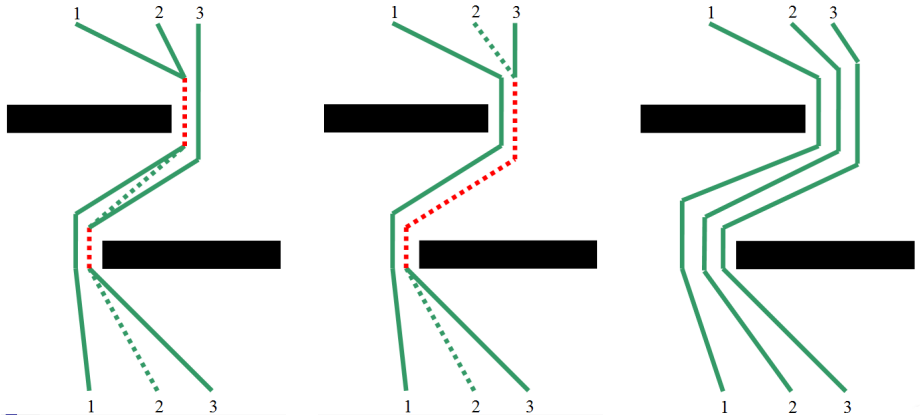
(a) (b) (d)

FIGURE 5.14 – Réparation grand voisinage : trois guides d'ondes

Finalement, la fonction de réparation décale la zone de conflit d'un guide d'ondes à l'autre jusqu'à sa disparition. Cela permet de se soustraire à la nécessité d'une boite englobante, et par la même occasion de la nécessité du paramètre **radius**. L'exemple illustré par la figure 5.14 permet d'introduire un nouveau mécanisme. Supposons que durant le mécanisme de réparation, le routage du guide d'ondes n°1 n'ai pas abouti. Dans ce cas, il est

possible d'effectuer une seconde passe de réparation mais dans le sens contraire de la première : après l'échec du routage du guide d'ondes n°1, ce dernier est rerouté mais cette fois-ci en autorisant l'entrée en conflit avec le guide d'ondes n°2. Une fois la situation de conflit constatée entre les deux guides d'ondes, il suffit d'appliquer la même règle de priorité, c'est-à-dire, attribuer la route au guide d'ondes n°1 et dérouter le guide d'ondes n°2 en interdisant son entrée en conflit avec le n°3.

L'ordre de routage a une importance cruciale dans la réussite d'un routage. Les exemples precedents ont été imaginés de telle sorte que l'ordre de routage soit favorable au mécanisme de réparation. Or, si l'on modifie cet ordre, la réparation peut être mise en échec. Reprenons l'exemple de la figure 5.14, mais avec un ordre de routage différent et quelques modifications mineures. Ce nouvel exemple est illustré par la figure 5.15. Dans cet exemple, les guides n°1 et n°3 sont routés en premier. Ils sont le résultat du mécanisme de reparation décrit précédemment. Lors du routage du guide d'ondes n°2, ce denier entre en conflit à la fois avec le guide d'ondes n°1 et le guide d'ondes n°3. Il en résulte une situation de conflit inextricable, malgré les deux passes de réparation. Cette situation est illustrée par la figure 5.15(a). Cet échec a pour cause l'ordre de routage. Un ordre favorable au mécanisme de réparation serait, pour cet exemple, de router les guides d'ondes dans l'ordre croissant ou décroissant des indices. Or, il est difficile de s'en rendre compte pendant le routage. Une solution permettant de rétablir le bon ordre de routage est de mettre en place un historique de réparation. Cet historique mémorise le fait que le routage des guides d'ondes n°1 et n°2 résulte d'une réparation, cela en liant les deux guides d'ondes. Ainsi, lors du routage du guide d'ondes n°3, il lui sera interdit d'entrer en conflit simultanément avec plusieurs guides d'ondes liées. Ansi, le guide n°3 est autorisé à entrer en conflit, soit avec le guide n°2, soit avec le guide n°3. Avec cette restriction, il suffit alors d'appliquer deux passes de réparation. Ainsi, l'historique de réparation permet une déduction de l'ordre de routage idéal pour la réussite d'une phase de réparation.

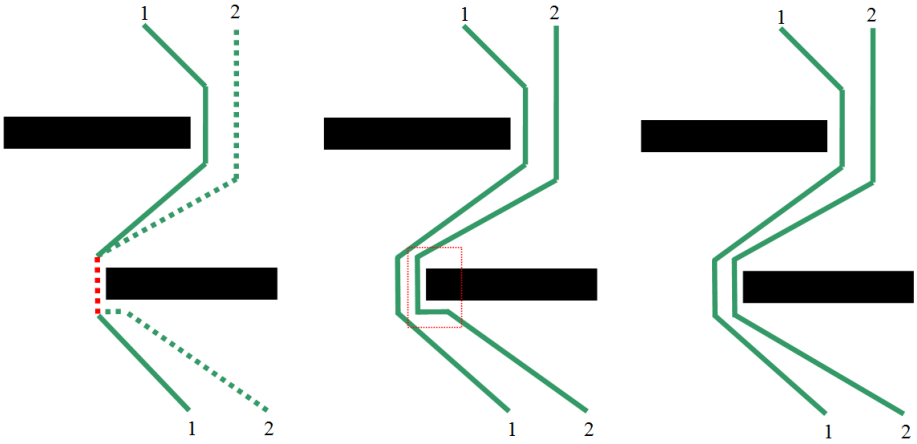


(a) (b) (c)

FIGURE 5.15 – Réparation grand voisinage : historique de réparation

Un effet de bord de la phase de réparation est la creation d'artefacts géométriques pour les guides d'ondes. La figure 5.16(a) illustre un exemple de routage de deux guides d'ondes nécessitant une phase de réparation. Le guide d'ondes n°2 est en conflit avec le guide d'ondes n°1 précédemment routé. Du fait du mécanisme de réparation qui repose sur la minimisation de la zone de conflit et l'attribution de la priorité au guide d'ondes créant la situation de conflit, un artefact géométrique peut apparaître lors du routage. La figure 5.16(b) illustre ce genre d'artefact pour le guide d'ondes n°2 (entouré d'un rectangle rouge). La solution choisie consiste à figer tous les guides d'ondes précédemment routés sauf le dernier guide d'ondes et de recalculer sa route. Pour notre exemple, cela consiste à figer le guide d'ondes n°1 et de rerouter le guide d'ondes n°2. La figure 5.16(c) illustre le résultat, avec la suppression de l'artefact.

La figure 5.17 illustre le routage de 200 guides d'ondes sur le mur d'un satellite. Le routage est effectué sur 3 niveaux.



(a) (b) (c)

FIGURE 5.16 – Artefact dû à au processus de réparation

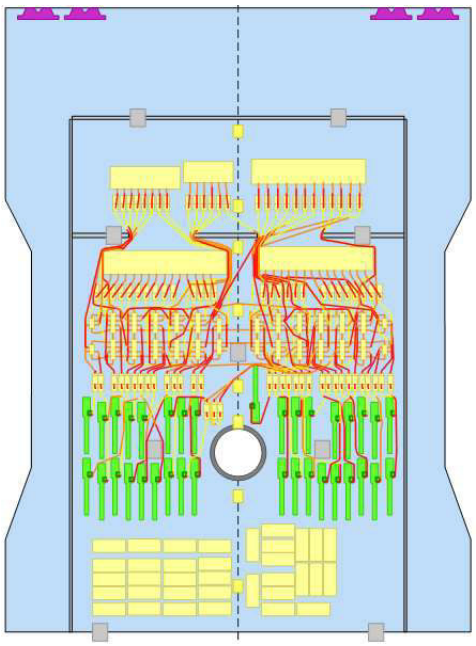
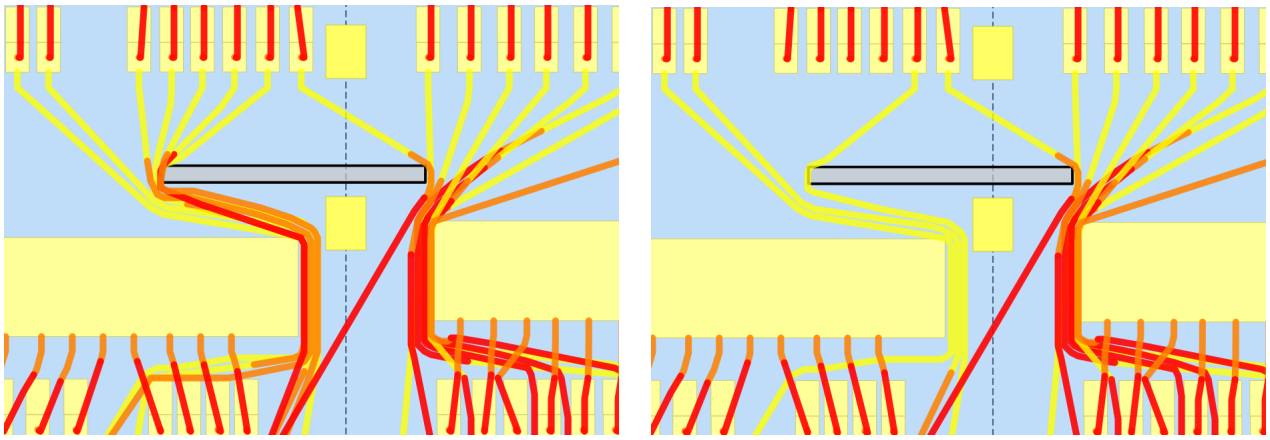


FIGURE 5.17 – Routage sur le mur d'un satellite

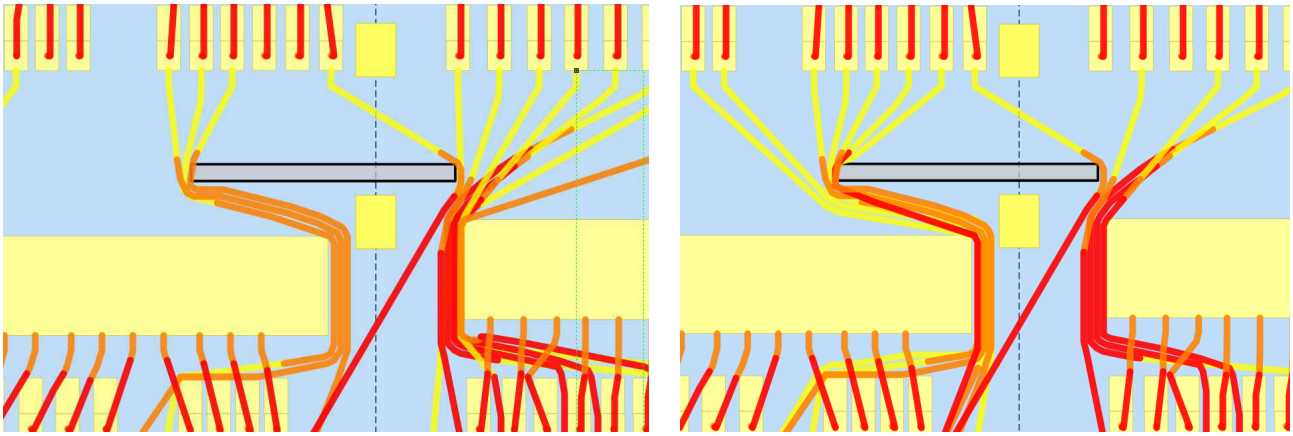
Cet exemple nous permet de mettre en évidence le processus de réparation. Pour ce faire nous allons nous focaliser sur une zone de ce mur. La figure 5.18(a) illustre le routage dans la zone qui nous interesse avec un routage en nappe de plusieurs guides d'ondes sur le premier, le deuxième et le troisième niveaux, respectivement de couleur jaune, orange et rouge. La figure 5.18(b) représente un effet nappe sur le premier niveau, sur la zone concernée ; les guides d'ondes du deuxième et troisième niveaux ont été supprimés afin de mieux distinguer ceux du premier niveau. L'effet nappe est le résultat du processus de réparation qui permet de router 3 guides d'ondes avec une configuration en S. Les figures 5.19(c) et 5.19(d) illustrent, respectivement, la même configuration mais sur le deuxième et troisième niveaux.

La réparation grand voisinage met en oeuvre quatre mécanismes principaux. Tout d'abord, elle applique une règle de priorité favorisant le dernier guide d'ondes routé, ce qui lui permet de transférer les zones de conflit aux guides d'ondes proches jusqu'à disparition des conflits. Ensuite, en cas d'échec de la premiere passe de réparation, une seconde passe est appliquée, mais dans le sens contraire de la première passe. Elle exploite



(a) (b)

FIGURE 5.18 – Résultat du processus de réparation (1)



(c) (d)

FIGURE 5.19 – Résultat du Processus de réparation (2)

également un historique de réparation lui permettant de déduire un ordre de routage idéal durant le processus de réparation. Enfin, le dernier mécanisme lui permet de supprimer les artefacts géométriques dus au processus de réparation.

5.4 Expérimentations

Dans un premier temps, les expérimentations sont effectuées sur les instances de test décrites dans la section 3.4 du chapitre 3. Afin d'être dans les mêmes conditions que la méthode simple voisinage et l'heuristique basée sur la génération de colonnes, la méthode grand voisinage se restreindra aux contraintes interdisant le chevauchement et le croisement entre guides d'ondes. Les guides d'ondes auront tous une épaisseur equivalente à une case.

Dans un second temps et avec cette fois-ci un modèle réaliste pour la méthode grand voisinage, nous présenterons des résultats concernant la comparaison entre les longueurs réelles de guides d'ondes extraites de véritables satellites et les longueurs estimées par la méthode grand voisinage.

Dans la suite du mémoire, la méthode grand voisinage sera notée *HGV* pour Heuristique Grand Voisinage. Pour rappel, nous notons **H** la méthode heuristique gloutonne simple voisinage et **HGC** l'heuristique basée sur

la génération de colonnes. La méthode HGV utilise un voisinage de 4, ce qui est un bon compromis entre finesse de l'estimation et temps d'exécution.

5.4.1 Instances de test

Les tableaux 5.1 et 5.2 illustrent les écarts en terme de longueur totale entre HGV et la plus petite valeur entre HGC et H (la meilleure estimation). On observe que sur la totalité des instances la méthode HGV permet d'obtenir une meilleure estimation que H et HGC. Il est intéressant de constater que HGV a résolu toutes les instances de routage, contrairement à H et HGC. Les améliorations obtenues varient en terme de pourcentage entre $-0,84\%$ et $-10,10\%$ en faveur de HGV, et en terme de longueur elles varient entre $-0.23mm$ et $-186.49mm$ en faveur de HGV. Ces écarts s'expliquent par le fait que la recherche de plus court chemin en grand voisinage limite les impacts de la discrétisation en grille, ce qui permet d'obtenir des estimations de longueur plus fines. Il est aussi intéressant de constater qu'en terme de temps de calcul, l'augmentation du nombre de directions à analyser lors d'une recherche de plus court chemin grand voisinage a un impact très limité. Les tableaux 5.3 et 5.4 illustrent ce constat. Cet impact limité s'explique par le fait que le surcoût en terme de temps de calcul est compensé par le gain apporté par les performances du mécanisme de réparation en terme de capacité de réparation des conflits et d'optimisation de l'espace du routage avec l'effet nappe et à la correction des artifacts géométriques, ce qui à pour effet de faciliter le routage, notamment pour les instances réelles dont les résultats sont exposés par le tableau 5.4.

5.4.2 Longueurs réelles

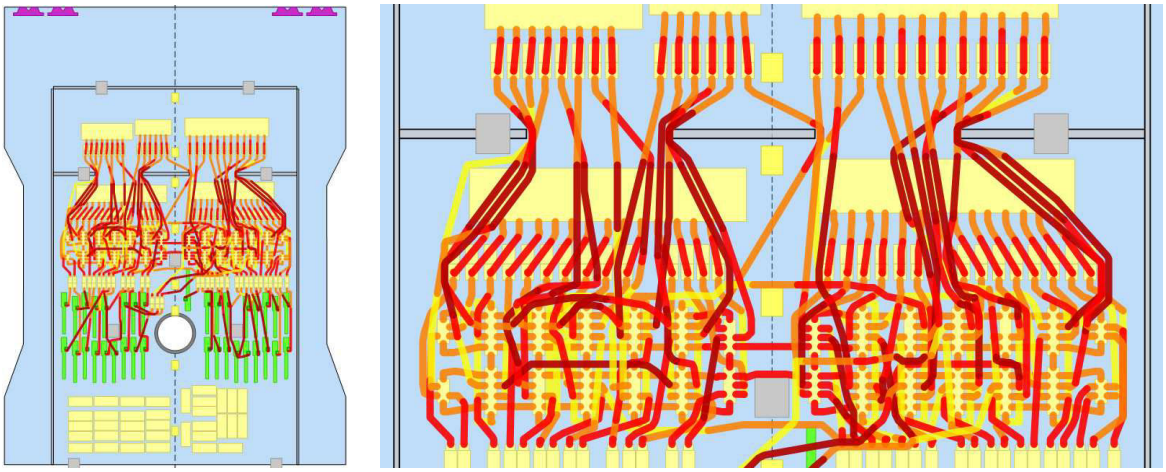
Afin de mieux évaluer les capacités de routage de la méthode grand voisinage nous avons effectué une comparaison entre des longueurs réelles de guides d'ondes et les longueurs estimées par notre méthode, ceci sur quatre satellites différents, mais partageant la même plateforme. Le positionnement des composants et le nombre de niveaux sont spécifiés pour chaque satellite. Pour ces expérimentations, la méthode grand voisinage adopte le modèle réaliste précédemment décrit, contrairement aux expérimentations ci-dessus. Les longueurs sont issues des véritables guides d'ondes connectant les composants de la charge utile.

Dans la section 5.1, nous avons vu que notre modèle associait une unique épaisseur à un guide d'ondes. Or, dans la réalité, un guide d'ondes peut avoir des sections de différentes épaisseurs, ceci du fait de l'orientation de ces mêmes sections. Afin d'être conservatif, les expérimentations effectuées sur les véritables satellites le sont en prenant comme épaisseur des guides d'ondes l'épaisseur la plus dimensionnante, donc la plus grande. Cette épaisseur correspond à une orientation des connecteurs de 0°. Généralement, plus de 95% des guides d'ondes sont routés avec une orientation de 90°, ainsi, sur le plan, la plus petite épaisseur est prise en compte, ce qui implique un encombrement moindre.

Les quatres satellites sélectionnés sont :
- Satellite A avec 72 guides d'ondes sur trois niveaux,
- Satellite B avec 162 guides d'ondes sur trois niveaux,
- Satellite C avec 254 guides d'ondes sur quatre niveaux,
- Satellite D avec 158 guides d'ondes sur quatre niveaux.

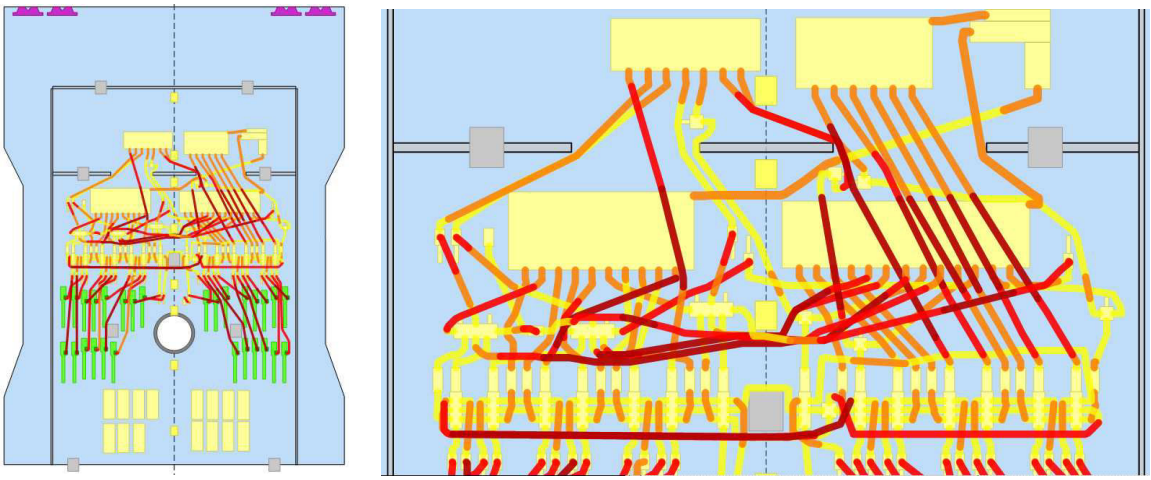
Le tableau 5.5 illustre les résultats concernant la longueur totale (somme des longueurs des guides d'ondes) pour chaque satellite. Les satellites A, B et D voient une légère sous-estimation de la longueur totale, contrairement au satellite C où nous avons une légère surestimation. La surestimation s'explique d'une part par le nombre très important de guides d'ondes et l'utilisation de l'épaisseur la plus dimensionnante, qui combinée au nombre de guide d'ondes engendre un encombrement important, et d'autre part par une complexité de la zone de routage associée au satellite C. La figure 5.20 illustre le résultats de routage sur 4 niveau. Il aisé de constater que l'espace

de routage est fortement encombrée, d'où la difficulté de routage. En revanche la sous estimation est quant à elle due à un nombre de guides d'ondes moins important, un espace de routage qui offre plus de lattitude, comme illustré par la figure 5.21 et au fait que le modèle réaliste ne prend pas en compte toutes les contraintes liées au routage, notamment les contraintes métiers. Il est intéressant de remarquer que les temps de calcul restent très raisonnables, 217 secondes pour le plus lent (254 guides d'ondes routés).



(a) (b)

FIGURE 5.20 – Satellite C : zone de routage



(a) (b)

FIGURE 5.21 – Satellite D : zone de routage

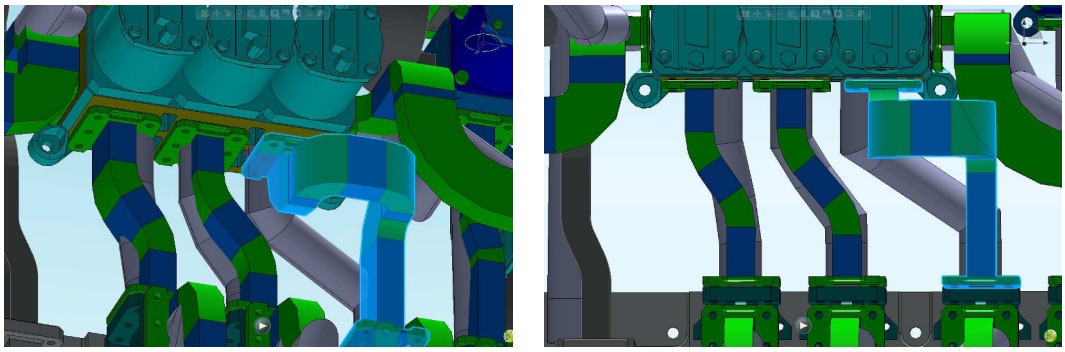
Nous allons à present nous intéresser aux écarts de longueur par guide d'ondes. Pour cela, nous nous limitons au satellite B. Néanmoins, les observations concernant le satellite B sont également valables pour les trois autres satellites.

Les tableaux 5.6 à 5.10 illustrent les longueurs réelles, les estimations de longueur (résultat du routage) et les écarts en valeur absolue du pourcentage pour 162 guides d'ondes, ceci dans l'ordre décroissant des écarts. Il est intéressant de remarquer que les écart les plus importants, avec un maximum de 41.05% concernent des guides d'ondes courts. Les guides longs ont un écart maximum de 24,65% pour un guide d'ondes d'une longueur réelle de $1768mm$ et d'une longueur estimée de $1332mm$. Globalement, les écarts de longueur s'expliquent par des choix différents en terme de route et de priorités données à certain guides d'ondes. En ce qui concerne les écarts les plus importants, ceux supérieur à 30%, l'explication tient au fait que certaines contraintes métiers liées au

routage ne sont pas pris en compte.

Nous allons à présent expliquer plus précisément les écarts importants pour les guides d'ondes 1 et 3 du tableau 5.6, respectivement 41.05% et 40.48%. Ces explications sont également applicables aux autres guides d'ondes.

Les figures 5.22(a)-(b) illustrent une superposition entre un routage réel, avec des guides d'ondes à section rectangulaire et de couleur bleu et verte, et le routage issu de notre modèle réaliste avec des guides d'ondes à section cylindrique (pour cet exemple) et de couleur grise. Ces figures représentent trois guides d'ondes courts. En partant de la gauche, pour les deux premiers guides d'ondes nous avons une correspondance presque parfaite entre le routage réel et notre routage. Cependant, le troisième guide d'ondes présente une forte sous-estimation. Notre routage est effectué sur le premier niveau et d'une manière plus directe, alors que le routage réel effectue un changement de niveau, ce qui a pour effet un écart de 41.05% entre la longueur réelle et la longueur estimée. Ce changement de niveau est dû à une contrainte métier qui n'est pas prise en compte.



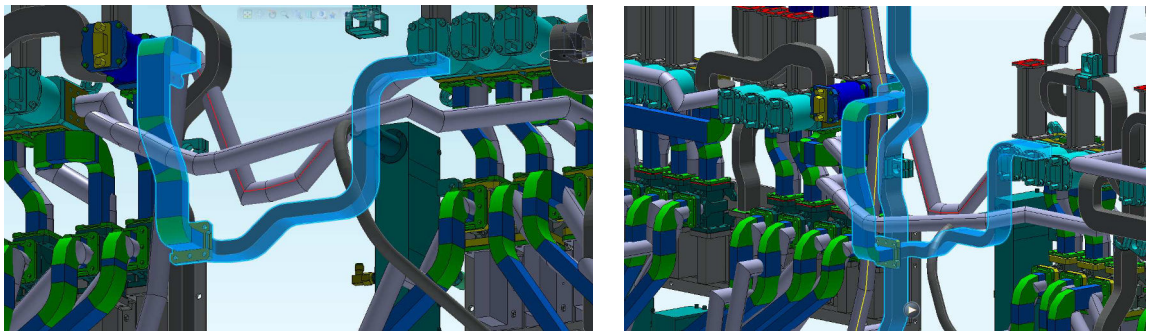
(a) (b)

FIGURE 5.22 – Satellite B : Contraintes métiers

Sur le même principe de superposition des routages, Les figures 5.23(a)-(b) illustrent une difference de choix de priorité entre le routage réel et le modèle réaliste. La figure 5.23(a) montre un guide d'ondes issu du routage réel, nommé $R1$, ce dernier est souligné d'une couleur bleuté (sélectionné). Elle montre également le guide d'ondes équivalent mais issu de notre routage, nommé $E1$, ce dernier est marqué d'un trait rouge soulignant sa géométrie. Le modèle réaliste a choisi une route plus directe alors que le routage réel effectue un changement de niveau et adopte une géométrie plus complexe, ce qui à pour effet d'allonger sa longueur. Il en résulte un écart de 40.48%. L'explication de cette difference est donnée par la figure 5.23(b). On y observe un guide d'ondes réel supplémentaire sélectionné, nommé $R2$, dont l'équivalent est souligné d'un trait jaune, nommé $E2$. Ce second guide d'ondes $R2$ réel a sa route qui passe juste devant le connecteur de notre premier guide d'ondes $R1$, ce qui a obligé ce dernier à changer de niveau. En revanche, avec notre modèle réaliste, ce même guide d'ondes $E2$ est routé au second niveau car la priorité a été donnée au premier guide d'ondes $E1$.

Il est intéressant de remarquer, d'après les tableaux 5.6 à 5.10, que 132 guides d'ondes parmi 162 ont un écart inférieur à 20%, 99 ont un écart inférieur à 10% et 60 ont un écart inférieur à 5%, ce qui fait respectivement 81.48%, 72.22%, 61.11%, et 37.04% des 162 guides d'ondes. Ainsi, plus de la moitié des guides d'ondes ont un écart inférieur ou égal à 10%. Il est intéressant de constater que 81.48% ont un écart inférieur ou égal à 20%, ce qui représente un écart de $15cm$ pour les plus grands guides d'ondes.

La figure 5.24 illustre un graphique avec en abscisse les identifiants des guides d'ondes et en ordonnée la longueur en millimètre. Le graphique contient deux courbes, une courbe pour les longueurs réelles et une courbe pour les longueurs estimées, respectivement courbe bleu et courbe rose. Les longueurs réelles et les longueurs estimées sont triées dans l'ordre croissant, d'où l'allure des courbes. Ce graphique valide les précédentes observation, à savoir qu'une grand partie des guides d'ondes ont un écart assez faible. Il apporte néanmoins



(a) (b)

FIGURE 5.23 – Satellite B : Priorités

une information supplémentaire sur la répartition entre surestimation et sous-estimation des longueurs, une répartition équivalente.

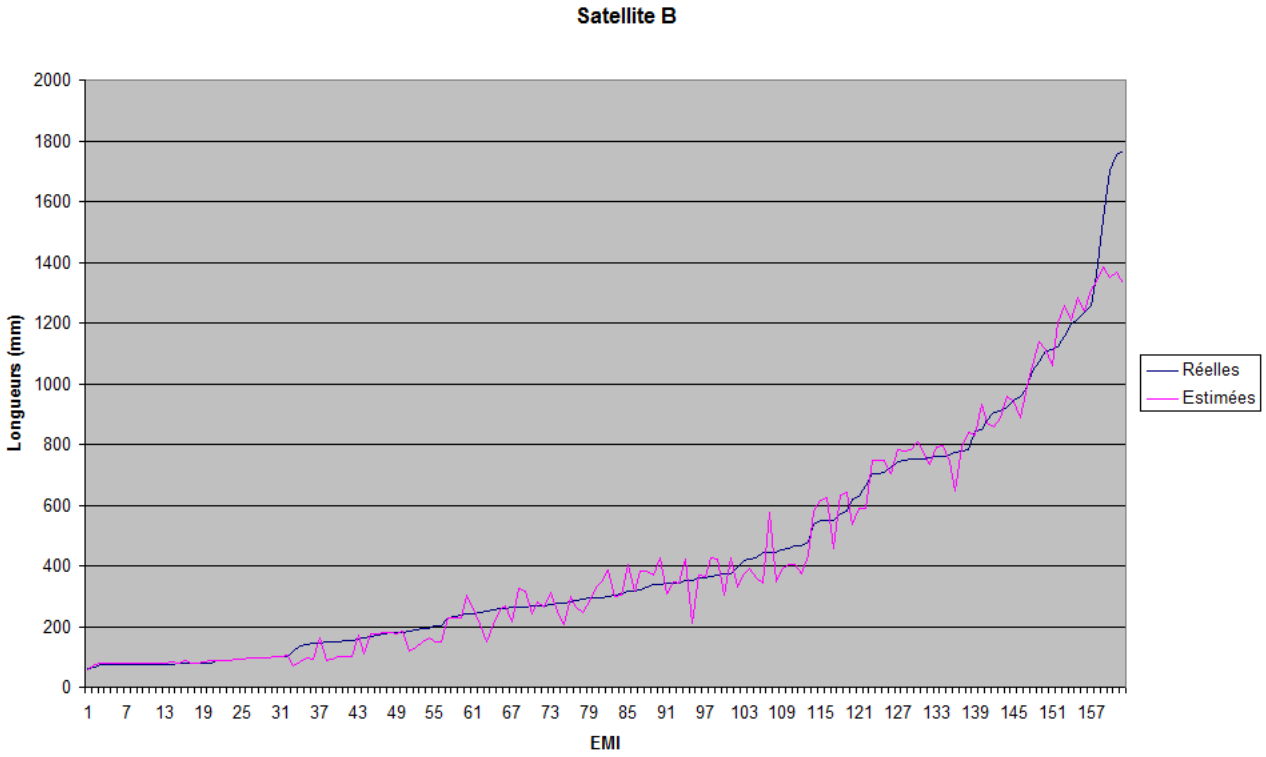


FIGURE 5.24 – Satellite B : Longueurs réelles / Estimées

Les figures 5.25, 5.26 et 5.27 illustrent le même constat pour, respectivement les satellites A, C et D. Les tableaux 5.11 et 5.12 présentent la proportion de guides d'ondes dont l'écart est inférieur à 5%, 10%, 15% et 20%, ceci en nombre de guides d'ondes et en pourcentage du nombre total de guides d'ondes pour les quatre satellites. Comme pour le satellite B, les satellites A, C et D ont plus de la moitié des guides d'ondes qui ont un écart inférieur ou égal à 10% et 80% et plus ont un écart inférieur ou égal à 20%, ce qui représente un écart de 20cm pour les plus grands guides d'ondes.

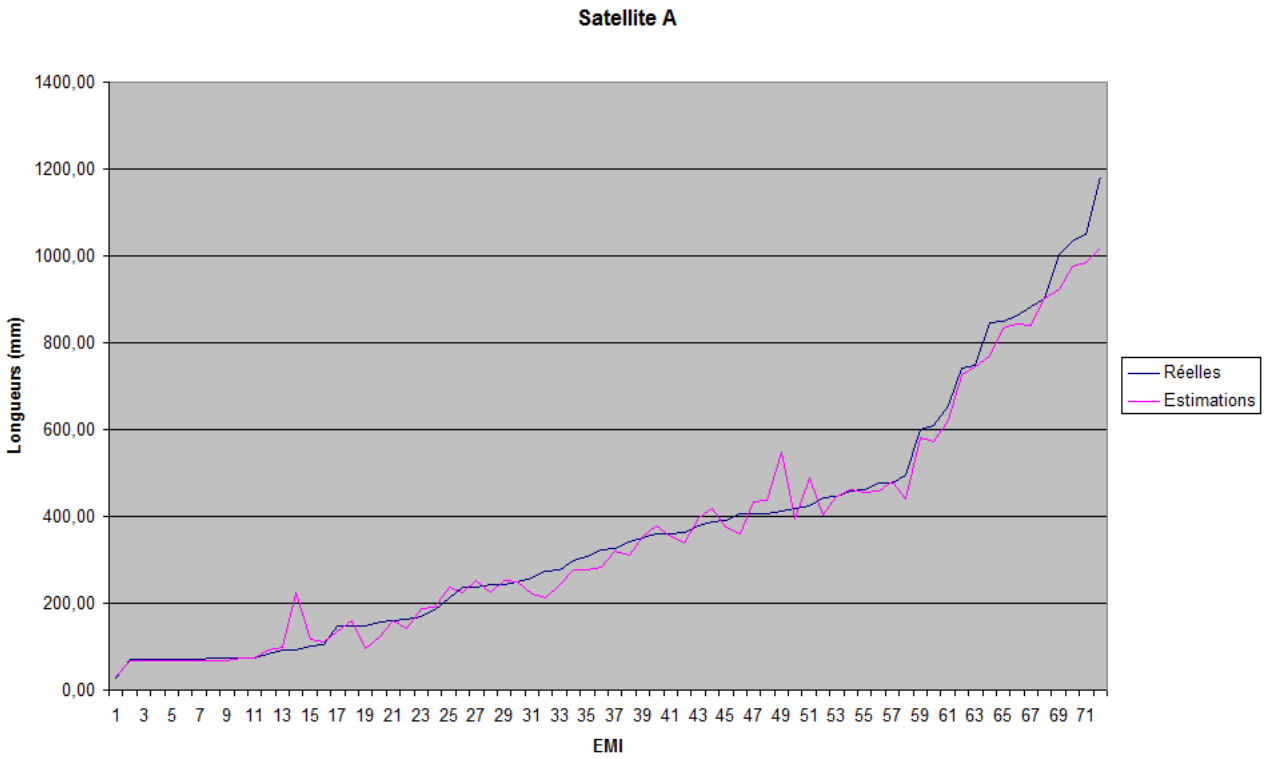


FIGURE 5.25 – Satellite A : Longueurs réelles / Estimées

5.5 Conclusion

Dans ce chapitre nous avons proposé un modèle réaliste pour la problématique de routage de guides d'ondes. Ce modèle réaliste est associé à une heuristique de routage basée sur une méthode gloutonne associée à mécanisme de réparation grand voisinage et à un algorithme de recherche de plus court chemin grand voisinage dont l'optimalité est garantie par celle de l'algorithme A* sur lequel elle est basée. Le méthode de réparation que nous proposons met en oeuvre quatre mécanismes principaux. Tout d'abord, elle applique une règle de priorité favorisant le dernier guide d'ondes routé, ce qui lui permet de transférer les zones de conflit aux guides d'ondes proches jusqu'à disparition des conflits. Ensuite, en cas d'échec de la premiere passe de réparation, une seconde passe est appliquée, mais dans le sens contraire de la première passe. Elle exploite également un historique de réparation lui permettant de déduire un ordre de routage idéal durant le processus de réparation. Enfin, le dernier mécanisme lui permet de supprimer les artefacts géométriques dus au processus de réparation.

Pour un contexte donné, cette méthode permet d'obtenir pour un guide d'ondes une solution localement optimale ou très proche de l'optimum selon que la méthode de réparation ait été un succès ou pas.

Sur les instances de test, la méthode grand voisinage a fourni des estimations de longueur plus fines que les deux heuristiques H et HGC, à savoir l'heuristique de réparation simple voisinage et l'heuristique basée sur la génération de colonnes, pour un faible impact en temps de calcul. De plus, les expérimentations menées sur de véritables satellites ont montré la précision des estimations fournies par notre méthode, dont 80% ont un écart inférieur ou égal de 20%. Les estimations fournies par notre méthode ont été comparées aux longueurs réelles et non pas aux estimations effectuées par les concepteurs en phase de réponse à appel d'offre.

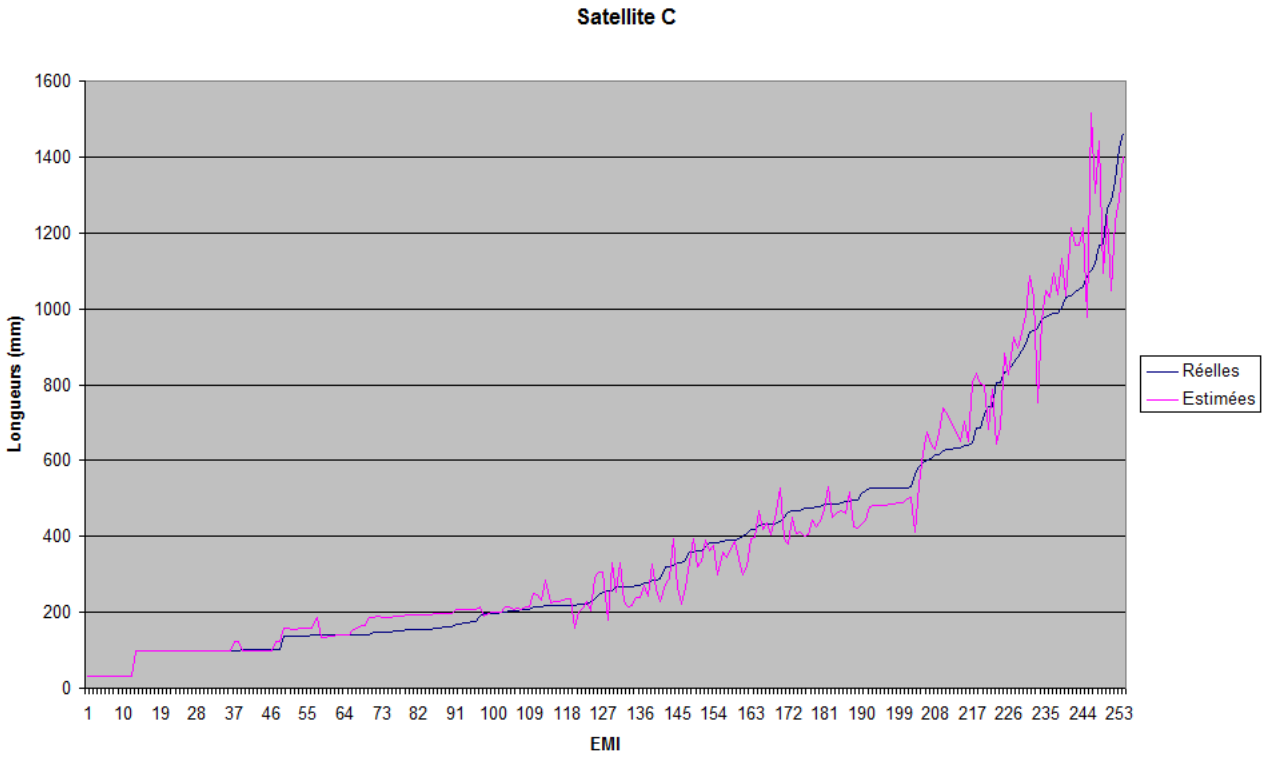


FIGURE 5.26 – Satellite C : Longueurs réelles / Estimées

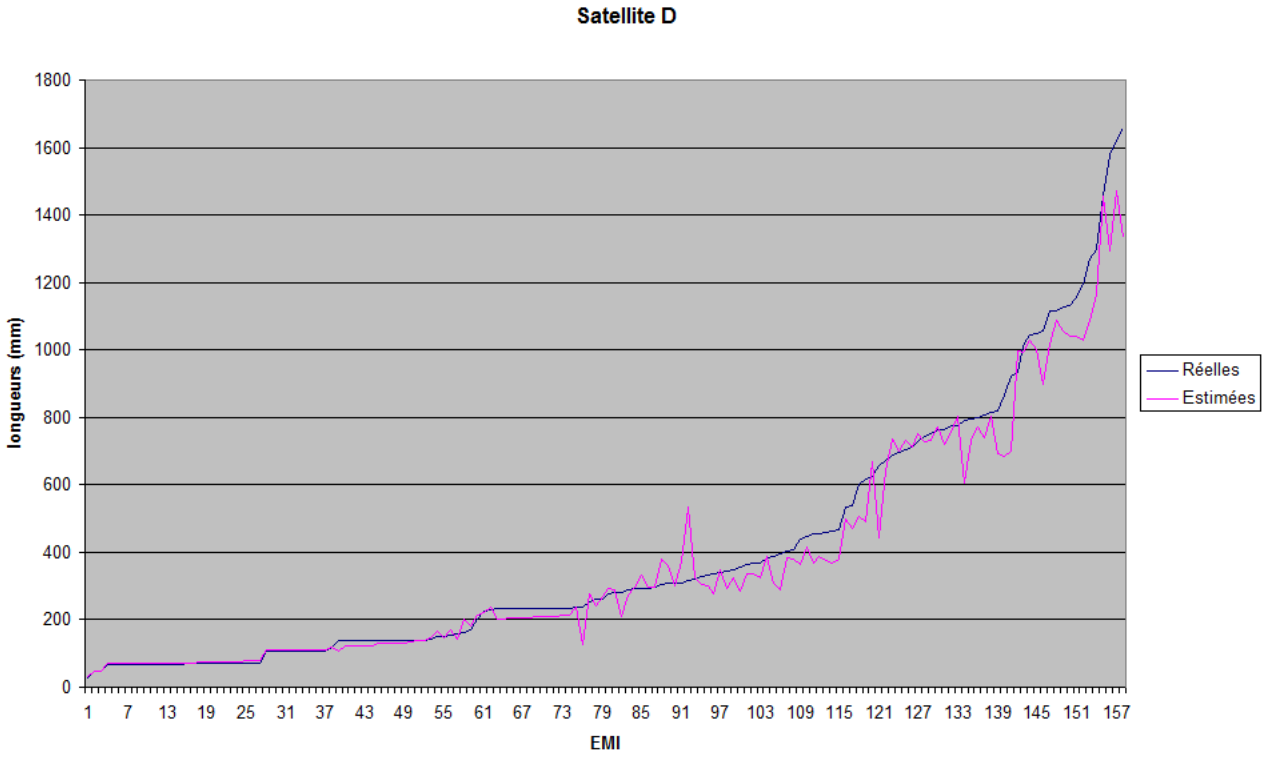


FIGURE 5.27 – Satellite D : Longueurs réelles / Estimées

(X,Y,Z)	Nwg	Occ	HGC	H	HGV	Écart
			valeur	valeur	valeur	(%)
Croisement						
(5,5,1)	2	0	10,83	10,83	**10,25**	-5,36%
(10,10,1)	2	0	-	29,31	**27,70**	-5,49%
(30,30,1)	2	0	-	97,60	**91,90**	-5,84%
(50,50,1)	2	0	-	165,88	**153,33**	-7,57%
Goulet d'étranglement						
(15,15,1)	4	4	65,11	65,11	**62,96**	-3,30%
(15,15,1)	4	9	66,53	66,53	**63,97**	-3,85%
(15,15,1)	5	22	85,67	85,67	**81,00**	-5,45%
(30,30,1)	5	8	163,95	169,41	**159,89**	-2,48%
(30,30,1)	6	9	-	-	**198,43**	-
(30,30,1)	7	5	225,21	-	**218,25**	-3,09%
(30,30,1)	7	7	217,17	218,58	**210,13**	-3,24%
(50,50,1)	11	3	593,26	596,78	**572,52**	-3,50%
(50,50,1)	11	7	-	623,00	**599,18**	-3,82%
(50,50,2)	16	16	408,75	390,75	**365,45**	-6,47%
Virage en S						
(15,20,1)	2	6	53,18	53,77	**52,38**	-1,50%
(15,20,1)	3	6	77,08	77,08	**75,76**	-1,71%
(15,20,1)	4	6	98,23	99,40	**96,17**	-2,10%
(30,30,1)	2	3	85,98	85,98	**83,87**	-2,45%
(30,30,1)	3	3	-	124,37	**121,51**	-2,30%
(30,30,1)	4	3	156,07	156,65	**152,48**	-2,30%
(30,30,1)	5	3	210,25	-	**205,31**	-2,35%
(50,50,1)	2	3	193,62	189,62	**185,45**	-2,20%
(50,50,1)	3	3	279,78	267,78	**261,49**	-2,35%
(50,50,1)	4	3	-	344,37	**336,28**	-2,35%
(50,50,1)	5	3	-	-	**439,52**	-
Croisement / Goulet d'étranglement / Virage en S						
(10,10,1)	2	56	27,90	27,90	**27,65**	-0,90%
(10,10,1)	3	18	27,31	27,31	**27,08**	-0,84%
(10,10,1)	4	22	36,97	36,97	**36,65**	-0,87%
(10,10,2)	6	42	46,83	46,83	**46,38**	-0,96%
(25,25,1)	5	11	142,40	141,81	**139,60**	-1,56%
(25,25,1)	6	14	206,02	-	**202,99**	-1,47%
(25,25,1)	5	8	256,92	-	**253,73**	-1,24%
(25,25,1)	6	17	177,88	175,54	**172,84**	-1,54%
(30,30,2)	15	19	240,98	241,81	**234,96**	-2,50%
(30,30,2)	23	24	497,87	494,94	**479,60**	-3,10%
(30,30,2)	28	22	343,81	342,98	**332,42**	-3,08%
(50,50,2)	14	34	502,84	505,28	**493,29**	-1,90%
(50,50,2)	31	20	690,16	677,43	**654,40**	-3,40%
(50,50,2)	38	24	981,77	1030,50	**952,32**	-3,00%
(200,120,1)	42	28	1394,74	1388,50	**1335,02**	-3,85%
(200,120,1)	42	30	1426,56	1422,07	**1365,72**	-3,96%
(100,100,1)	23	14	844,99	836,74	**809,15**	-3,30%
(100,100,1)	27	18	1032,13	1028,85	**988,72**	-3,90%

TABLE 5.1 – Résultats d'expérimentation

(X,Y,Z)	Nwg	Occ	HGC	H	HGV	Écart
			valeur	valeur	valeur	(%)
(215,140,3)	80	24	2125,10	2151,44	**2082,39**	-2,01%
(215,140,3)	37	21	865,83	867,59	**844,18**	-2,50%
(215,140,3)	138	34	4118,97	4169,94	**3956,27**	-3,95%
(215,140,3)	112	31	2490,41	2510,94	**2387,56**	-4,13%
(215,140,3)	140	34	3667,47	3704,94	**3524,07**	-3,91%
(215,140,3)	96	27	2662,34	2677,40	**2606,43**	-2,10%
(250,230,2)	13	7	1084,15	1084,15	**1071,14**	-1,20%
(250,230,2)	48	2	3183,22	3173,21	**3106,57**	-2,10%
(250,230,3)	91	23	5977,23	6060,94	**5792,53**	-3,09%
(300,200,3)	37	18	1190,08	1201,11	**1160,33**	-2,50%
(300,200,3)	74	22	1911,87	1903,91	**1843,56**	-3,17%
(300,200,3)	74	23	2031,41	2032,78	**1969,45**	-3,05%
(300,200,3)	80	21	2953,79	2939,40	**2865,92**	-2,50%
(300,200,3)	96	25	3845,19	3814,88	**3645,50**	-4,44%
(300,200,3)	112	28	3390,94	3390,69	**3204,20**	-5,50%
(300,200,3)	123	27	5435,38	5329,88	**5223,12**	-2,00%
(300,200,3)	138	32	5744,47	5754,57	**5565,87**	-3,11%
(300,200,3)	140	32	4993,47	5038,29	**4835,18**	-3,17%
(300,200,3)	148	30	6469,64	6477,95	**6339,75**	-2,01%
(300,200,3)	168	35	7386,49	7452,66	**7213,21**	-2,35%

TABLE 5.2 – Résultats d'expérimentation : instances réelles

Temps d'exécution (en seconde) : HGC, H, HGV					
(X,Y,Z)	Nwg	Occ	HGC	H	HGV
Croisement					
(5,5,1)	2	0	< 1	< 1	< 1
(10,10,1)	2	0	< 1	< 1	< 1
(30,30,1)	2	0	< 1	< 1	< 1
(50,50,1)	2	0	< 1	< 1	< 1
Goulet d'étranglement					
(15,15,1)	4	4	< 1	< 1	< 1
(15,15,1)	4	9	< 1	< 1	< 1
(15,15,1)	5	22	< 1	< 1	< 1
(30,30,1)	5	8	< 1	< 1	< 1
(30,30,1)	6	9	3,70	< 1	< 1
(30,30,1)	7	5	< 1	< 1	< 1
(30,30,1)	7	7	< 1	< 1	< 1
(50,50,1)	11	3	< 1	< 1	< 1
(50,50,1)	11	7	1,20	< 1	< 1
(50,50,2)	16	16	< 1	< 1	< 1
Virage en S					
(15,20,1)	2	6	< 1	< 1	< 1
(15,20,1)	3	6	< 1	< 1	< 1
(15,20,1)	4	6	< 1	< 1	< 1
(30,30,1)	2	3	< 1	< 1	< 1
(30,30,1)	3	3	< 1	< 1	< 1
(30,30,1)	4	3	< 1	< 1	< 1
(30,30,1)	5	3	2,20	< 1	< 1
(50,50,1)	2	3	0,69	< 1	< 1
(50,50,1)	3	3	3,30	< 1	< 1
(50,50,1)	4	3	9,90	< 1	< 1
(50,50,1)	5	3	35,60	< 1	< 1
Croisement / Goulet d'étranglement / Virage en S					
(10,10,1)	2	56	< 1	< 1	< 1
(10,10,1)	3	18	< 1	< 1	< 1
(10,10,1)	4	22	< 1	< 1	< 1
(10,10,2)	6	42	< 1	< 1	< 1
(25,25,1)	5	11	< 1	< 1	< 1
(25,25,1)	6	14	< 1	< 1	< 1
(25,25,1)	5	8	< 1	< 1	< 1
(25,25,1)	6	17	< 1	< 1	< 1
(30,30,2)	15	19	< 1	< 1	< 1
(30,30,2)	23	24	2,42	< 1	< 1
(30,30,2)	28	22	< 1	< 1	< 1
(50,50,2)	14	34	< 1	< 1	< 1
(50,50,2)	31	20	1,94	< 1	< 1
(50,50,2)	38	24	13,50	< 1	< 1
(200,120,1)	42	28	< 1	< 1	< 1
(200,120,1)	42	30	< 1	< 1	< 1
(100,100,1)	23	14	< 1	< 1	< 1
(100,100,1)	27	18	5,00	< 1	< 1

TABLE 5.3 – Temps d'exécution de HGC, HGC et HGV

Temps d'exécution (en seconde) : HGV vs best(H,HGC) : écart en seconde						
(X,Y,Z)	Nwg	Occ	HGC	H	HGV	écart
(215,140,3)	80	24	13	**3**	5	2
(215,140,3)	37	21	1	**1**	2	1
(215,140,3)	138	34	49	46	**40**	-6
(215,140,3)	112	31	7	**2**	5	3
(215,140,3)	140	34	56	**30**	35	5
(215,140,3)	96	27	17	**2**	4	2
(250,230,2)	13	7	1	**1**	**1**	0
(250,230,2)	48	2	4	**3**	4	1
(250,230,3)	91	23	288	7	**5**	-2
(300,200,3)	37	18	5	**2**	**2**	0
(300,200,3)	74	22	3	**3**	4	1
(300,200,3)	74	23	4	**5**	**5**	1
(300,200,3)	80	21	23	7	9	2
(300,200,3)	96	25	22	**3**	4	1
(300,200,3)	112	28	20	**3**	4	1
(300,200,3)	123	27	167	**5**	6	1
(300,200,3)	138	32	62	**4**	6	2
(300,200,3)	140	32	94	7	**6**	-1
(300,200,3)	148	30	460	**7**	**7**	0
(300,200,3)	168	35	422	**8**	9	1

TABLE 5.4 – Temps d'exécution de HGC, H et HGV : Instances réelles

Satellite	Longueur totale		Écart	temps CPU
	Réelle	Estimée	(%)	(s)
A	25 895,36	26 619,00	-2,72%	55
B	70 381,36	68 626,73	-2,49%	123
C	94 587,70	96 108,21	1,61%	217
D	63 295,73	61 458,20	-2,91%	113

TABLE 5.5 – Longueurs totales

EMI	Longueurs (mm)		Écart
	Réelle	Estimée	(%)
1	148	87,25	41,05%
2	252	149,19	40,80%
3	352	209,51	40,48%
4	138	83,61	39,41%
5	117	71,57	38,83%
6	184	116,79	36,53%
7	148	94,12	36,41%
8	143	93,69	34,48%
9	153	100,88	34,06%
10	151	100,92	33,17%
11	153	102,95	32,72%
12	140	94,87	32,24%
13	163	110,56	32,17%
14	189	130,05	31,19%
15	443	576,39	30,11%
16	298	386,10	29,56%
17	316	403,32	27,63%
18	202	147,56	26,95%
19	241	303,54	25,95%
20	204	151,10	25,93%
21	279	207,00	25,81%
22	338	424,69	25,65%
23	1768	1 332,11	24,65%
24	263	325,09	23,61%
25	192	147,37	23,24%
26	442	341,74	22,68%
27	1756	1 366,26	22,19%
28	444	345,97	22,08%

TABLE 5.6 – Satellite B : Detail des longueurs - EMI 1 à 28

EMI	Longueurs (mm)		Écart
	Réelle	Estimée	(%)
29	1701	1 351,50	20,55%
30	351	422,57	20,39%
31	264	317,54	20,28%
32	468	375,70	19,72%
33	295	352,65	19,54%
34	373	301,82	19,08%
35	254	208,39	17,96%
36	262	215,22	17,86%
37	323	380,33	17,75%
38	363	427,39	17,74%
39	397	327,48	17,51%
40	551	455,66	17,30%
41	775	648,31	16,35%
42	248	208,34	15,99%
43	428	360,68	15,73%
44	330	381,81	15,70%
45	66	76,21	15,47%
46	368	423,44	15,06%
47	290	246,64	14,95%
48	272	312,39	14,85%
49	192	163,77	14,70%
50	621	534,19	13,98%
51	451	388,59	13,84%
52	550	622,97	13,27%
53	375	424,26	13,14%
54	293	331,21	13,04%
55	466	405,84	12,91%
56	77	86,78	12,71%
57	548	613,60	11,97%
58	456	404,16	11,37%

TABLE 5.7 – Satellite B : Detail des longueurs - EMI 29 à 58

EMI	Longueurs (mm)		Écart
	Réelle	Estimée	(%)
59	662	589,95	10,88%
60	571	631,69	10,63%
61	581	642,03	10,50%
62	147	162,44	10,50%
63	418	374,67	10,37%
64	1542	1 383,82	10,26%
65	341	307,06	9,95%
66	266	239,75	9,87%
67	850	932,61	9,72%
68	76	83,35	9,67%
69	481	435,97	9,36%
70	276	250,61	9,20%
71	338	368,43	9,00%
72	74	80,52	8,81%
73	1158	1 255,58	8,43%
74	75	81,18	8,24%
75	536	579,57	8,13%
76	750	809,98	8,00%
77	74	79,84	7,89%
78	959	886,62	7,55%
79	783	841,39	7,46%
80	60	55,55	7,42%
81	73	78,24	7,17%
82	1122	1 200,80	7,02%
83	73	78,12	7,01%
84	285	265,34	6,90%
85	81	86,57	6,88%
86	73	78,00	6,84%
87	166	177,14	6,71%
88	420	392,13	6,63%
89	76	81,02	6,60%
90	630	590,77	6,23%
91	702	745,51	6,20%
92	703	746,20	6,14%
93	243	257,68	6,04%
94	1072	1 136,46	6,01%
95	707	748,40	5,86%
96	160	169,35	5,84%
97	1214	1 283,47	5,72%
98	281	296,97	5,68%
99	74	78,18	5,65%
100	742	783,35	5,57%
101	267	281,83	5,55%
102	907	858,01	5,40%

TABLE 5.8 – Satellite B : Detail des longueurs - EMI 59 à 102

EMI	Longueurs (mm)		Écart
	Réelle	Estimée	(%)
103	74	77,63	4,90%
104	75	78,56	4,75%
105	293	279,27	4,69%
106	1113	1 060,94	4,68%
107	762	797,31	4,63%
108	74	77,34	4,52%
109	80	83,45	4,31%
110	1257	1 309,17	4,15%
111	750	780,46	4,06%
112	747	776,65	3,97%
113	761	791,13	3,96%
114	102	105,66	3,59%
115	925	958,07	3,58%
116	359	371,29	3,42%
117	727	702,15	3,42%
118	753	777,06	3,20%
119	175	180,40	3,08%
120	182	176,42	3,06%
121	261	268,65	2,93%
122	173	178,01	2,90%
123	757	735,95	2,78%
124	236	229,59	2,72%
125	763	742,72	2,66%
126	99	101,61	2,63%
127	912	889,40	2,48%
128	182	186,49	2,47%
129	259	252,81	2,39%
130	232	226,45	2,39%
131	304	297,38	2,18%
132	268	262,68	1,98%
133	846	829,55	1,94%
134	96	97,84	1,92%
135	1042	1 061,28	1,85%
136	779	792,96	1,79%
137	1358	1 336,09	1,61%
138	360	365,77	1,60%
139	93	91,65	1,45%
140	306	301,62	1,43%
141	341	345,48	1,31%
142	1194	1 209,16	1,27%
143	90	88,87	1,25%
144	101	99,91	1,08%
145	980	969,68	1,05%
146	879	869,79	1,05%

TABLE 5.9 – Satellite B : Detail des longueurs - EMI 103 à 146

EMI	Longueurs (mm)		Écart
	Réelle	Estimée	(%)
147	90	89,21	0,88%
148	77	77,66	0,86%
149	98	97,24	0,78%
150	95	95,53	0,56%
151	947	942,22	0,50%
152	78	78,38	0,49%
153	1236	1 241,62	0,46%
154	1107	1 111,71	0,43%
155	78	78,30	0,38%
156	229	228,38	0,27%
157	87	87,23	0,27%
158	96	95,75	0,26%
159	317	317,70	0,22%
160	342	341,40	0,18%
161	179	179,24	0,14%
162	91	91,04	0,05%

TABLE 5.10 – Satellite B : Detail des longueurs - EMI 147 à 162

Satellite	Nb Total de wg	Nb de wg			
		$\leq 5\%$	$\leq 10\%$	$\leq 15\%$	$\leq 20\%$
A	72	31	54	66	68
B	162	60	99	117	132
C	254	82	143	179	203
D	158	69	109	132	145

TABLE 5.11 – Répartition des guides d'ondes selon l'écart en % : en nombre de guides d'ondes

Satellite	Nb Total de wg	pourcentage de wg			
		$\leq 5\%$	$\leq 10\%$	$\leq 15\%$	$\leq 20\%$
A	72	43,06%	75,00%	91,67%	94,44%
B	162	37,04%	61,11%	72,22%	81,48%
C	254	32,28%	56,30%	70,47%	79,92%
D	158	43,67%	68,99%	83,54%	91,77%

TABLE 5.12 – Répartition des guides d'ondes selon l'écart en % : en pourcentage de guides d'ondes

CHAPITRE 6

CONCLUSION

6.1 contributions

Afin de traiter la problématique de routage, dans cette thèse nous avons proposé quatre méthodes de résolution, une méthode exacte qui s'est révélée inefficace et trois méthodes heuristiques. L'heuristique génération de colonnes et l'heuristique de réparation simple voisinage ont donné de bons résultats, avec un léger avantage pour l'heuristique de réparation en terme de temps de calcul. Cependant, l'heuristique de réparation grand voisinage s'est révélée plus efficace en terme de capacité à réparer les situations de conflit et en terme d'estimation des longueurs qui, grâce au grand voisinage sont plus fines qu'avec un simple voisinage. Le grand voisinage permet de limiter l'impact sur les longueurs de la discrétisation de la zone de routage en grille. Elle permet également de mieux contrôler la courbure des guides d'ondes afin de rendre leur géométrie plus réaliste.

Le modèle réaliste associé à l'heuristique grand voisinage a fourni des estimations de longueurs très proches des longueurs réelles pour quatres satellites, pour des temps de calcul très raisonnables, inférieur à 5 min pour le plus lent. L'exercice pour les quatres satellites a été effectué en 10 jours. Cette durée regroupe le positionnement des composants, la creation de la connectivité pour chaque satellite, le routage et l'analyse des résultats. En phase de réponse à appel d'offre, le même exercice pour un seul satellite prendrait plusieurs mois aux concepteurs pour avoir des estimations de longueur correctes.

La contribution de cette thèse réside dans la proposition d'une méthode de multi-routage efficace basée sur des mécanismes novateurs, notamment l'algorithme de recherche de plus court chemin grand voisinage basé sur les masques de direction, et le méthode de réparation globale avec tous les mécanismes qu'elle met en oeuvre pour proposer un routage réaliste.

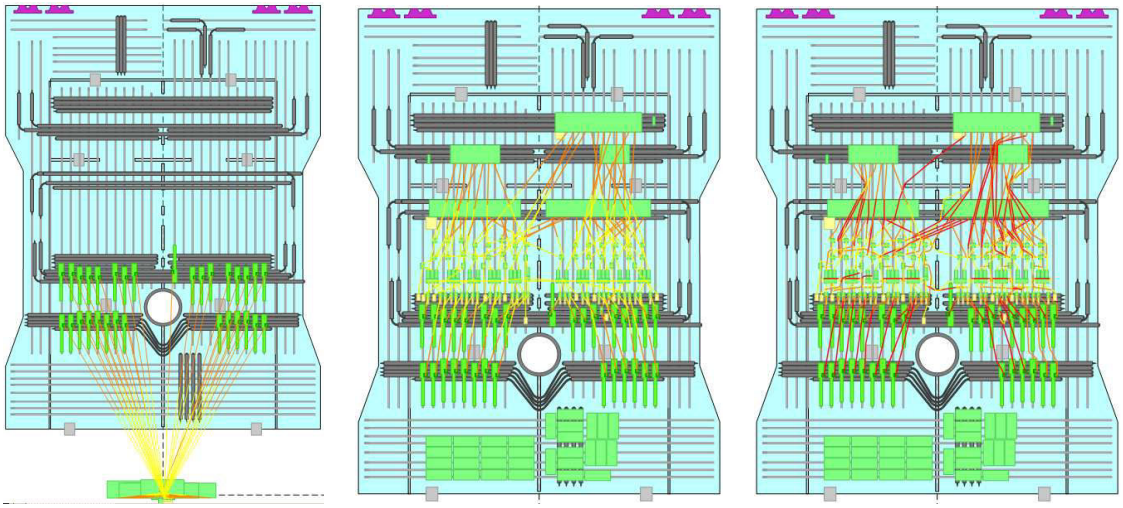
La contribution de cette thèse réside également dans le fait d'avoir offert un outil d'estimation des longueurs aux équipes d'accommodation, leur permettant de gagner en efficacité.

Lors d'une phase de réponse à appel d'offre, les équipes d'accommodation doivent déterminer un positionnement des composants et le routage associé respectant le cahier des charges en terme de performance. En effet, la solution de positionnement et de routage choisie impacte les performances finales de la charge utile et les spécifications requises pour les équipements. Une solution optimale permettrait d'augmenter les performances et d'avoir des spécifications d'équipements plus basses et donc une solution moins coûteuse. Les figures 6.1(a), (b) et (c) illustrent respectivement, la creation des composants et de la connectivité, le positionnement des composants et enfin le routage des guides d'ondes. Ces trois étapes ont nécessité un investissement de 5 jours.

Lors de cette phase, les équipes d'accommodation demandent au bureau d'études de travailler sur plusieurs options d'aménagement des murs. Cette étape se faisait « à la main » et nécessitait quelques mois. En général,

par manque de temps, la première solution réalisable est retenue.

Associé à un outil permettant de positionner manuellement les composants de la charge utile sur un mur et de créer la connectivité, le module de routage permet de calculer les longueurs des guides d'ondes et de détecter les éventuelles incohérences. Le résultat permet aux équipes d'accommodation de chercher le meilleur compromis en positionnant les équipements différemment. Ainsi, cette association permet de tester très rapidement différentes configurations jusqu'à trouver le meilleur compromis design, performance et coût. La rapidité d'exécution permet à ces équipes d'inclure dès la phase de proposition la solution trouvée et d'éviter ainsi de prendre des marges pénalisantes pour la compétitivité de l'offre de design. Cette association d'outils (positionnement manuel et module de routage) est désormais utilisée pour l'aménagement de tous les satellites de télécommunication, a été un facteur clé dans le gain d'un satellite où des performances optimales étaient requises. Elle a également contribuée à une réduction des coûts de 12%.



(a) (b) (c)

FIGURE 6.1 – Création d'une instance de routage, un mur

6.2 perspectives

Différentes perspectives s'offrent à nous sur la base de la méthode de réparation grand voisinage et de l'algorithme de recherche de plus court chemin grand voisinage. On peut citer trois perspectives à court terme qui permettraient d'avoir un routage plus réaliste. La première concerne la prise en compte de pieces standards tel que les coudes à 90° ou 45°. Comme décrit précédemment, l'algorithme de recherche de plus court chemin grand voisinage repose sur l'utilisation de masques permettant de déterminer les cellules survolées par une direction lors du routage. La prise en compte des pieces standards est alors possible en remplaçant les directions par des coudes par exemple. Les masques contiendraient alors les cellules survolées par ces coudes.

Une seconde amélioration serait la prise en compte du changement d'épaisseur d'un guide d'ondes par l'utilisation d'un twist ou par un simple changement de direction. Une troisième perspective serait de discrétiser les niveaux de routage. Actuellement un niveau est déterminé par un z_{min} et un z_{max}. Un guide d'ondes utilise la totalité de l'espace d'un niveau quelque soit sa hauteur, ceci tout au long de sa route. Il n'est pas possible d'avoir une superposition de guide d'ondes dans un même niveau. L'idée serait alors d'avoir une discrétisation des niveaux plus fine, équivalente à celle utilisée pour le plan XY.

Une perspective à long terme serait de s'affranchir partiellement ou totalement de la discrétisation de la zone de routage, ainsi le routage serait effectué dans un espace continu. On peut imaginer adopter le même principe que

pour les circuits imprimés, c'est-à-dire un routage global et un rouage détaillé. Le routage global serait effectué avec une discrétisation grossière. Il permettrait de détecter les conflit potentiels. Le routage détaillé quant à lui serait effectué dans un espace continu et exploiterait les informations récoltées par le routage global.

Une dernière perspective pour notre module de routage serait son utilisation pour le routage de cable coax, qui est équivalente au routage de guide d'ondes mais avec moins de contraintes.

Deuxième partie

Validation de la charge utile d'un satellite de télécommunication

CHAPITRE 1

INTRODUCTION

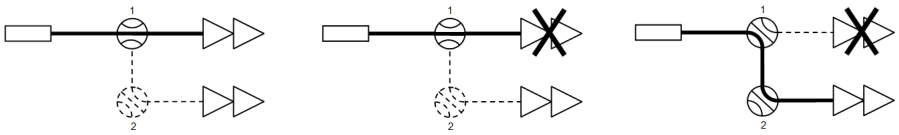
Une fois un satellite de télécommunication en orbite géostationnaire autour de la terre, aucune réparation ne peut être entreprise en cas de panne. Afin de pallier cette éventualité, un mécanisme de redondance a été mis en place, principalement pour les tubes amplificateurs qui sont les équipements les plus sollicités et les plus coûteux de la charge utile. Le mécanisme de redondance est basé sur une matrice de switches (redondance et sélectivité), en section d'entrée et en section de sortie comme indiqué dans le paragraphe 1.2. Ce mécanisme est caractérisé par sa capacité à absorber un nombre de pannes de tube amplificateur NPT prédéfini. Cette capacité, appelée aussi robustesse, doit être validée en phase de design, d'où la problématique de validation de matrice de switches.

1.1 Problématique de validation de matrice de switches

La robustesse d'une charge utile est préalablement définie par le cahier des charges et s'exprime en un nombre maximum NPT de cas de pannes de tubes amplificateurs que le satellite doit compenser afin de mener à bien sa mission. Le nombre NPT signifie que pour un nombre de tubes amplificateurs NT, il y a NPT tubes amplificateurs redondants. Généralement, 4 à 6 tubes sont redondants pour un ensemble d'une vingtaine, voire d'une trentaine de tubes amplificateurs.

La matrice de redondance garantit à la fois un accès aux tubes amplificateurs nominaux lorsqu'il n'y à pas de pannes, et un accès aux tubes amplificateurs redondants en cas de pannes grâce à la capacité de reconfiguration de la matrice. Cette capacité permet de rediriger les signaux impactés par ces pannes vers les tubes amplificateurs redondants. Une reconfiguration s'effectue par la rotation d'un certain nombre de switches. Une description plus détaillée de la charge utile est disponible dans la section 1.2 du chapitre 1.

La figure 1.1 illustre un cas de reconfiguration d'une matrice de redondance après une panne d'un tube amplificateur. La figure ne représente que la section d'entrée de la matrice de redondance. Cette matrice est composée d'un canal (rectangle), de deux switches et de deux tubes amplificateurs (un tube nominal et un tube redondant). Dans un premier temps, la matrice est dans un état de fonctionnement nominal. Un signal est acheminé du canal au tube amplificateur n°1, en passant par le switche n°1 (voir figure 1.1(a)). Supposons que le tube n°1 tombe en panne, cas illustré par la figure 1.1(b). Cette panne nécessite la reconfiguration de la matrice de redondance afin d'acheminer le signal vers le tube redondant. Pour ce faire, les switches sont positionnés de telle sorte que le signal puisse atteindre le tube redondant. Pour notre exemple cela se traduit par la rotation du switche n°1 (le switche n°2 est dans la bonne position). La reconfiguration de la de la matrice de redondance est illustrée par la figure 1.1(c).



(a) (b) (c)

FIGURE 1.1 – Exemple de reconfiguration de la charge utile en cas de panne

Pour plus de flexibilité, au cours de son existence, un satellite de télécommunication peut être amené à effectuer plusieurs missions, une mission étant définie par une sélection d'un nombre *NCA* de canaux de communication actifs parmi un ensemble de *NC* canaux. La propriété de sélectivité de mission est établie, comme pour la propriété de redondance, par une matrice de switches dite de sélectivité. Comme la matrice de redondance, la matrice de sélectivité est présente en en section d'entrée et en section de sortie (voir figure 1.12 de la section 1.2 du chapitre Introduction Générale). Dans la suite du mémoire, l'association matrice de sélectivité et de redondance (ou de redondance et de sélectivité) est appelé simplement : matrice de switches.

Durant une phase de réponse à un appel d'offre, les concepteurs doivent proposer un design d'une matrice de switches répondant aux spécifications du cahier des charges délivré par le client, ceci dans un délai limité. La spécification majeure est la propriété de robustesse du design aux cas de pannes de tubes amplificateurs. Du fait de la multiplicité des missions pour un satellite de télécommunication, la propriété de robustesse doit non seulement être validée pour tous les cas de panne, mais aussi quelque soit la mission, c'est-à-dire pour toutes les sélections de *NCA* canaux actifs parmi *NC* canaux disponibles. Soit une combinaison définie par un ensemble de *NPT* tubes en pannes et par un ensemble de *NCA* canaux actifs parmi *NC*. La validation consiste alors à examiner toutes les combinaisons possibles, qui correspondent à autant de cas dégradés de fonctionnement. L'examen d'une combinaison doit déterminer l'existence ou non d'une reconfiguration de la matrice de switches permettant d'acheminer le signal.

Cette validation exhaustive permet d'identifier, s'ils existent, les cas dégradés pour lesquels il n'existe pas de reconfiguration possible de la matrice de switches, dans ce cas une combinaison est dite non valide. Cette identification permet alors aux concepteurs, soit de modifier le design afin que la matrice soit reconfigurable pour tous les cas dégradés, soit de supprimer les cas d'utilisation correspondant à ces cas dégradés dans le cas où une modification de design n'est pas suffisante. Plusieurs itérations de validations/modifications sont nécessaires afin d'aboutir à un design répondant aux attentes du client. On peut donc aisément conclure que le temps de validation est crucial durant cette phase de design. Par exemple, une validation dont une combinaison est définie par 18 canaux actifs parmi 27 canaux de communication et 4 cas de pannes de tubes parmi 24 tubes amplificateurs nécessite l'examen de 49 milliards de combinaisons. Cet exemple de validation est un cas réel. Supposons que la validation d'un cas dégradé est effectuée en 0.1 seconde, alors les 49 milliards de combinaisons seraient validées au bout d'environ 155 années (sur une seule machine).

Depuis 2001, la société EADS Astrium développe une application, **SWITCHWORKS**, permettant de valider le design d'une matrice de switches. Elle utilise un processus de validation basé sur l'exploration systématique et exhaustive de la combinatoire. Comme indiqué dans le paragraphe 1.2, un satellite de télécommunication comprend deux matrices de switches symétriques, du fait de cette symétrie il n'est pas nécessaire de valider les matrices des deux sections. Seule la matrice de la section de sortie est validée, ce choix étant purement pratique. Cette application repose sur une modélisation qui associe un problème de satisfaction de contrainte binaire à chaque combinaison. Avec cette modélisation, le processus de validation consiste alors à énumérer et à résoudre plusieurs millions, voire plusieurs milliards de CSPs. La résolution du CSP i associé à la combinaison i est effectuée indépendamment de la résolution des CSPs précédents. La méthode de résolution est une méthode faisant partie de la famille des méthodes dites complètes ou exactes, basée sur une recherche systématique alliant un schéma prospectif « look-ahead » et un schéma rétrospectif « look-back ». Elle permet de prouver la

réalisabilité d'un CSP, et donc la validité ou la non-validité d'une combinaison.

Avec l'accroissement de la complexité des satellites de télécommunication, l'application **SWITCHWORKS** atteint ses limites en terme de temps de validation. Une limitation inhérente au principe de validation adopté, énumération exhaustive, et à la méthode de recherche utilisée pour résoudre la succession de CSPs. Dans cette thèse nous nous proposons d'apporter des améliorations algorithmiques (et d'implémentation) permettant de réduire le temps de traitement d'une combinaison, et d'étudier des mécanismes permettant soit, de réduire la combinatoire, soit de capitaliser l'effort de résolution des CSPs successifs. Nous nous proposons également d'étudier des pistes permettant de s'affranchir de la combinatoire.

1.2 Méthode de Validation

1.2.1 Modélisation

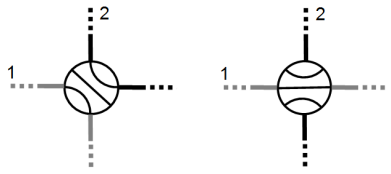
Comme un très grand nombre de problèmes combinatoires réels (ou théoriques) [162], telles que les problématiques de configuration, de planification, d'ordonnancement, d'affectation de ressources..., la problématique de validation du design d'une matrice de switches appartient à la famille des Problèmes de Satisfaction de Contraintes (*Constraint Satisfaction Problem ou CSP*). Ces problématiques partagent une description commune basée sur un formalisme très simple, celui des CSPs qui autorise en général une modélisation claire et intuitive. Un CSP [156, 5] est défini par un ensemble $\mathcal{V}$ de variables, un ensemble $\mathcal{D}$ de domaines de définition qui encadrent les valeurs que peuvent prendre ces variables, et un ensemble $\mathcal{C}$ de contraintes qui conditionnent les valeurs que pourront prendre ces variables.

Pour la problématique de validation, à chaque cas dégradé est associé un CSP binaire $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ défini par une séquence $\mathcal{V} = (V_1, \cdots, V_{NCA})$ de variables, chacune prenant ses valeurs dans un domaine discret figurant dans $D = (dom[V_1], \cdots, dom[V_{NCA}])$ et par une séquence de C contraintes binaires. Une contrainte binaire C_{ij} est un sous-ensemble du produit cartésien $dom[V_i] \times dom[V_j]$ qui stipule les couples de valeurs non permis pour les variables V_i et V_j. Une variable $V_i \in V$ est associée au canal de communication actif $i, i = 1 \cdots NCA$. Elle prend ses valeurs dans le domaine $dom[V_i] \in D$ constitué de tous les chemins possibles reliant le canal i aux tubes amplificateurs auxquels il a accès, avec un nombre maximal de switches traversés afin de limiter les pertes radio-fréquentielles. Un chemin est défini par une origine et une destination, respectivement un canal et un tube, et par un ensemble de switches traversés. De ce fait, un chemin n'appartient qu'à un unique domaine. Les domaines sont donc disjoints.

Comme dit précédemment, l'ensemble $\mathcal{C}$ regroupe les contraintes binaires. Pour la problématique de validation, ces contraintes portent sur les valeurs et traduisent la compatibilité entre les chemins. Cette compatibilité s'exprime en termes de compatibilité de position de switches entre deux chemins. C'est-à-dire que dans le cas où deux chemin, appartenant à deux domaines distincts, utiliseraient le même switche, ce dernier doit avoir la même position pour les deux chemins. Dans le cas contraire, les deux chemins sont dit incompatibles. La figure 1.2 illustre la compatibilité et l'incompatibilité entre deux chemins en terme de position de switches. La figure 1.2(a) illustre deux chemins utilisant le même switche, dans une même position. Ces deux chemins sont dit compatibles. En revanche, la figure 1.2(b) illustre deux chemins utilisant le même switche mais dans deux positions différentes. Ces deux chemins sont dit incompatibles. Il est admis que deux chemins ayant le même canal pour origine ou le même tube amplificateur pour destination sont dit incompatibles.

Ainsi le modèle CSP binaire $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ est définit de la manière suivante :

- $\mathcal{V} = (V_1, \cdots, V_{NCA})$, où la variable V_i est associée au canal i.
- $\mathcal{D} = (dom[V_1], \cdots, dom[V_{NCA}])$, avec $dom[V_i]$ l'ensemble des chemins possibles reliant le canal i aux tubes amplificateurs



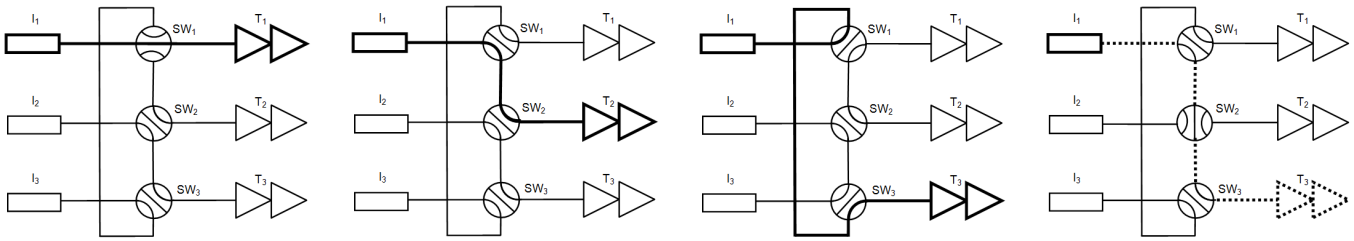
(a) (b)

FIGURE 1.2 – Compatibilité entre chemins

– $\mathcal{C} = \{C_1, \cdots, C_m\}$ est un ensemble fini de m contraintes dont l'arité est égale à 2. Une contrainte $C_j \in \mathcal{C}$ est associée à une relation $rel(C_j)$ représentant l'ensemble des tuples autorisés pour les deux variables $vars(C_j)$ liées par la contrainte C_j. Les tuples autorisés sont constitués de chemins compatibles.

Pour un tel CPS, une solution réalisable est une instantiation complète où chaque variable se voit attribuer une valeur, un chemin compatible avec les chemins attribués aux autres variables. La figure 1.2 illustre la compatibilité et l'incompatibilité entre deux chemins en terme de position de switches.

Prenons l'exemple de la matrice de switches illustrée par la figure 1.3(a). Cette matrice comprend 3 canaux, 3 switches et 3 tubes amplificateurs TWTA. Elle doit être validée pour 2 canaux actifs parmi 3 (NCA = 2, NA = 3), pour 1 panne de tube parmi 3 (NPT = 1, NT = 3) et avec une longueur maximale de chemin de 2 switches traversés. Soit V_1 la variable associée au canal I_1. Le domaine de V_1 doit alors contenir tous les chemins permettant d'atteindre tous les tubes dont le canal I_1 peut disposer. La figure 1.3(a) illustre un premier chemin qui relie le canal I_1 au tube T_1 en empruntant le switche SW_1. Les figures 1.3(b) et 1.3(c) illustrent deux autres chemins menant respectivement aux tubes T_2 et T_3, en passant par les switches SW_1 et SW_2 pour le premier et les switches SW_1 et SW_3 pour le second. Jusqu'à présent les chemins décrits ont au maximum une longueur de chemin égale à deux switches traversés. La figure 1.3(d) présente un chemin reliant le canal I_1 au tube T_3 avec une longueur de chemin égale à trois switches traversés (SW_1, SW_2 et SW_3). Or, le nombre de switches traversés est limité à deux switches, donc ce chemin ne peut appartenir au domaine de la variable V_1. Finalement, le domaine de la variable V_1 comprend trois chemin, donc trois valeurs.



(a) (b) (c) (d)

FIGURE 1.3 – Exemples de chemins

Lors de la validation, le processus d'énumération de la combinatoire suit un schéma bien précis. Ce schéma est exposé ci-après.

1.2.2 Schéma d'énumération

Le processus d'énumération permet de générer les combinaisons les unes après les autres de manière exhaustive, selon un schéma d'énumération. Le schéma d'énumération consiste à sélectionner un premier ensemble de canaux actifs, pour lequel seront ensuite énuméré tous les cas de pannes de tubes amplificateurs. Une fois la combinatoire des pannes de tubes explorée, un nouvel ensemble de canaux actifs est sélectionné et la combinatoire des tubes

est une nouvelle fois explorée. Le processus d'énumération se termine une fois la combinatoire des canaux actifs examinée en entier. Ainsi, pour un ensemble de canaux actifs, C_{NT}^{NPT} cas de pannes de tubes amplificateurs sont générés, avec C_{NT}^{NPT} le nombre de NPT pannes de tubes possibles parmi NT tubes. Durant la validation, C_{NC}^{NCA} ensembles de canaux actifs sont explorés, avec C_{NC}^{NCA} le nombre de NCA canaux actifs parmi NC canaux.

Soit une matrice de switches avec 2 canaux actifs parmi 3 canaux et une panne de tube amplificateur parmi 3 tubes amplificateurs. Nous avons donc $NC = 3$, $NCA = 2$, $NT = 3$ et $NPT = 1$. Cet exemple présente 3 ensembles de deux canaux actifs et 3 ensembles d'une panne de tubes amplificateurs, soit $C_3^2 \times C_3^1 = 9$ combinaisons.

Soit I_i et I_j les deux canaux i et j actifs, et T_k le tube k en panne. Une combinaison est notée (I_i, I_j) (T_k), avec $i \neq j$. Sur la base de cette notation, la figure 1.4 illustre l'exploration de la combinatoire selon le schéma défini.

- 1 (I_1, I_2) (T_1)
- 2 (I_1, I_2) (T_2)
- 3 (I_1, I_2) (T_3)
- 4 (I_1, I_3) (T_1)
- 5 (I_1, I_3) (T_2)
- 6 (I_1, I_3) (T_3)
- 7 (I_2, I_3) (T_1)
- 8 (I_2, I_3) (T_2)
- 9 (I_2, I_3) (T_3)

FIGURE 1.4 – Exemple d'énumération de combinatoire

Dans notre exemple d'énumération, pour un même ensemble de canaux actifs, par exemple (I_1, I_2), deux combinaisons successives ne diffèrent que d'un seul tube en panne, comme par exemple les combinaisons (I_1, I_2) (T_1) et (I_1, I_2) (T_2). De même, pour un même ensemble de tubes en panne, par exemple (T_1), les combinaisons successives, par exemple (I_1, I_2) (T_1) et (I_1, I_3) (T_1), ne diffèrent que d'un seul canal actifs.

D'un point de vue de la modélisation CSP, le schéma d'énumération adopté met en évidence le caractère dynamique de la problématique de validation. En effet, deux combinaisons, donc deux CSPs binaires successifs diffèrent, soit de quelques valeurs ou d'une variable et de quelques valeurs. Ainsi, le passage d'un CSP binaire à un autre (successif) se traduit par l'ajout et/ou la suppression de quelques valeurs, ou par le remplacement d'une variable par une autre et par l'ajout et/ou la suppression de quelques valeurs. Ceci correspond à la définition d'un CSP Dynamique (DCSP) binaire [52, 114]. Un CSP Dynamique (DCSP) binaire est une suite de CSP binaires statiques $P_{(0)}, P_{(1)}, \cdots, P_{(N)}$, chacun résultant du précédent par une opération sur les valeurs (ajout et/ou suppression) et sur les variables (activation, désactivation). La singularité de ce DCSP binaire est l'existence d'une période T qui rythme les modifications apportées aux CSPs. Dans la suite du mémoire, nous appellerons un DCSP binaire de période T, un DCSP binaire tel que tous les T CSP un changement de variable est opéré (activation, désactivation), et entre chaque couple de CSP successifs ayant les mêmes variables actives, seulement des modifications de valeurs dans les domaines sont effectuées (ajout et/ou suppression). Il en résulte une forte similarité entre deux CSPs binaires statiques successifs. La période T correspond au nombre de cas de pannes de tubes amplificateurs. Pour l'exemple illustré par la figure 1.4, la période T est égale à 3.

Le but premier de la validation exhaustive est d'identifier, si elles existent, les combinaisons pour lesquelles il n'existe pas de reconfiguration possible. Cette identification est effectuée en résolvant le CSP associé à chaque combinaison. Lors de la résolution de cette succession de CSPs, l'aspect dynamique de la problématique n'est pas exploité par la méthode couramment utilisée par EADS Astrium. L'une de nos contributions consiste justement à prendre en compte les similarités entre la succession de CSPs.

1.2.3 Processus de validation

Le context de validation est constitué :
- D'une matrice de switches, avec le nombre de canaux de communication, noté NC et le nombre de tubes amplificateurs, noté NT
- Du nombre de canaux de communication actifs, noté NCA

– Du nombre de pannes de tubes amplificateurs, noté NPT
– Du nombre maximum de switches traversés, noté NSW

Le processus de résolution débute par la création de l'ensemble des variables avec les domaines associés. Pour une matrice donnée, NC variables sont crées, correspondant à autant de canaux de communication. À chaque variable V_i est associé un domaine. Une fois tous les domaines constitués, une matrice de compatibilité entre les chemins est générée. Cette matrice constitue l'ensemble des contraintes binaires reliant tous les couples de chemins.

La suite de la validation consiste à énumérer toutes les combinaisons de manière exhaustive et en suivant le schéma décrit précédemment. Chaque combinaison est alors associée à un CSP binaire, avec NCA variables, où chaque variable est associée à un canal actif. La prise en compte des pannes de tubes amplificateurs est effectuée en supprimant les valeurs pour lesquelles le chemin associé a pour destination l'un des tubes en pannes, ceci pour chaque variable et chaque tube en panne.

La validation proprement dite consiste alors à résoudre itérativement l'ensemble des CSPs binaires. La résolution permet soit de trouver une solution, la première, et d'identifier la combinaison associée comme étant valide, soit à prouver sa non satisfiabilité et de l'identifier comme étant non valide. Une fois les combinaisons non valides identifiées, il est alors possible de relancer une validation pour ces seules combinaisons, ceci en incrémentant le nombre maximum de switches traversés. Le fait d'incrémenter ce nombre permet de créer de nouveaux chemins, plus longs et ainsi de fournir de nouvelles valeurs qui pourront potentiellement diminuer le nombre de combinaisons non valides. Ce type de validation est dit validation incrémentale, à l'opposé de la validation dite simple qui n'est effectuée que pour un unique nombre maximum de switches traversés.

La résolution fait appel à une méthode complète basée sur une recherche arborescente de type depth-first, avec à la fois un schéma prospectif « look-ahead » et un schéma rétrospectif « look-back », respectivement phase de progression et phase de régression. La phase de progression permet de déterminer l'affectation à réaliser, des modifications à apporter au problème, ce qui lui permet de vérifier la présence ou non d'une impasse. La phase de régression permet quant à elle de déterminer le traitement à réaliser lorsque une impasse est rencontrée. Le traitement consiste à décider du saut en arrière à effectuer, cela grâce à l'exploitation des informations implicites contenues dans les situations d'échec (impasse).

Sur la base de cet existant, dans la suite du mémoire, nous présentons les différentes améliorations algorithmiques et d'implémentation qui permettent de réduire le temps de traitement d'une combinaison. Ces améliorations portent principalement sur des mécanismes prospectifs et rétrospectifs, ainsi que sur un mécanisme permettant de capitaliser l'effort de résolution entre les CSPs successifs. Mais tout d'abord nous proposons un état de l'art des méthodes de résolution d'un Problème de Satisfaction de Contraintes.

CHAPITRE 2

ÉTAT DE L'ART

Dans ce chapitre, nous proposons une introduction aux CSPs ainsi qu'un tour d'horizon des techniques de résolution d'un Problème de Satisfaction de Contraintes basées sur la Programmation Par Contraintes (PPC).

2.1 Problème de Satisfaction de Contraintes

Le formalisme CSP a été introduit dans les années soixante dix pour modéliser un grand nombre de problèmes définis par des contraintes de ressources matérielles, temporelles, spatiales, etc. tels que des problèmes de planification, d'ordonnancement, d'affectation de ressources...[162]. Les problèmes CSPs ont la particularité commune d'être fortement combinatoires, du point de vue calculatoire. Les problèmes considérés induisent bien souvent des complexités algorithmiques élevées et bon nombre d'entre eux relèvent de la classe des problèmes NP-complets [61, 129]. La problématique de validation nous place dans le cadre du formalisme CSP Binaire Discret, de part la modélisation choisie. Dans cette section nous allons définir, dans un cadre plus général le Problème de Satisfaction de Contraintes. Tout d'abord, nous donnerons une définition formelle du CSP. Ensuite, nous décrirons les différents mécanismes mis en oeuvre pour le résoudre.

2.1.1 Définitions

Un CSP est défini par un ensemble de variables prenant leurs valeurs dans un domaine et un ensemble de contraintes qui conditionnent les valeurs que pourront prendre les variables. Il est généralement présenté sous la forme d'un réseau de contraintes $(\mathcal{V}, \mathcal{C})$.

Définition 2.1.1.1 *Un réseau de contrainte est un couple $(\mathcal{V}, \mathcal{C})$ où :*
- *$\mathcal{V} = \{V_1, \cdots, V_n\}$ est un ensemble fini de n variables. À chaque variable V correspond un domaine $dom(V)$ représentant l'ensemble des valeurs pouvant lui être affectée.*
- *$\mathcal{C} = \{C_1, \cdots, C_m\}$ est un ensemble fini de m contraintes. Une contrainte $C \in \mathcal{C}$ est associée à une relation $rel(C)$ représentant l'ensemble des tuples autorisés pour les variables $vars(C)$ liées par la contrainte C.*

Une contrainte C lie une variable V si et seulement si $V \in vars(C)$. Le nombre $|vars(C)|$ de variables liées par une contrainte C défini l'arité de celle-ci. L'arité permet de définir trois familles de contraintes :

- Unaire : une contrainte unaire lie une seule variable,
- Binaire : une contrainte binaire lie deux variables,

– N-aire : une contrainte n-aire lie $n > 2$ variables.

En revanche, le nombre de contraintes liant une variable V représente le degré de cette dernière. Deux variable V_i et V_j sont dites voisines si et seulement il existe une contrainte C liant les deux variables, c'est-à-dire $\exists C \in \mathcal{C}, \{V_i, V_j\} \in Vars(C)$. Soit une variables V, l'ensemble des variables voisines à la variable V est noté $\Gamma(V)$.

À tout CSP peut être associé un hypergraphe $(\mathcal{V}, \mathcal{C})$ de contraintes. Cet hypergraphe est obtenu en associant à chaque variable un sommet et à chaque contrainte C une hyper-arete reliant les sommets dont les variables correspondantes sont liées par C.Dans le cas d'un CSP binaire, $(\mathcal{V}, \mathcal{C})$ est alors dit graphe de contraintes ou réseau de contraintes.

L'affectation d'une valeur à une variable est appelée instanciation (ou assignation). Soit $s = (v_1, v_2, \cdots, v_p)$ une instanciation partielle où p variables sont affectées, avec $p = |s| \leq |V|$ et v_i la valeur affectée à la variable V_i. Si $p = |V|$, s est appelée instanciation totale. Une solution réalisable (ou simplement solution) est une instanciation totale de sorte que chaque contrainte soit satisfaite. L'ensemble des solutions réalisables d'un réseau de contraintes P est noté $sol(P)$. Ces solutions appartiennent à l'espace de recherche E qui est l'ensemble des assignations (partielles ou complètes) possibles. Cet espace est égal au produit cartésien de l'ensemble des domaines des variables : $E = dom(V_1) \times dom(V_2) \times \cdots \times dom(V_n)$

Un réseau de contraintes est aussi appelé instance CSP. Par abus de langage, CSP et instance CSP sont parfois confondus. Sans perte de généralité, dans ce mémoire, l'ensemble $vars(C_j)$ des variables liées par la contrainte C_j est considéré comme étant ordonné. Il est alors possible d'obtenir la position $pos(V_i, C_j)$ d'une variable V_i dans $vars(C_j)$. Un tuple t est dit autorisé par une contrainte C_j si et seulement si $t \in rel(C)$. Un tuple t est dit support de l'instanciation (V_i, v) dans C_j si t est autorisé par C_j et tel que $t[pos(V_i, C_j)] = v$. (V_i, v) est dit consistant par rapport à C_j si et seulement si il existe un support de (V_i, v) dans C_j. L'opération consistant à déterminer si un tuple est support d'une contrainte est appelé test de consistance.

Définition 2.1.1.2 *Soient P une instance CSP, C une contrainte de P, V_i une variable de $vars(C)$ et v une valeur de $dom(V_i)$. (V_i, v) est dit consistant par rapport à C si et seulement si il existe un support de (V_i, v) dans C. C est dit arc-consistant si et seulement si pour toute variable V_i de $vars(C)$ et pour toute valeur v de $dom(V_i)$, (V_i, v) est consistant par rapport à C. P est arc-consistant si et seulement si toute contrainte de P est arc-consistante.*

Une instanciation partielle consistante est globalement consistante si elle peut être étendue à une solution. De plus, une instanciation partielle est dite arc-consistante si pour chaque contrainte C entre les variables $(V_1, \cdots, V_p)$, il existe un support. Une instanciation s partielle consistante est appelée *impasse* ou *deadend* si et seulement si il existe une variable non instantiée dont toutes les extensions de s sont inconsistantes. Lors de l'exécution d'un algorithme de filtrage, un *deadend* est une instanciation partielle telle que le domaine d'une variable non instantiée devient vide.

Une assignation partielle qui ne peut être étendue à aucune solution réalisable de P est un *nogood*. Ainsi, toute instanciation partielle n'apparaissant dans aucune solution de P est un *nogood*. De plus, tout *deadend* est un *nogood*. Cependant, un *nogood* n'est pas forcément un *deadend*. Un *nogood* minimal de P est un *nogood* n'incluant pas lui-même un autre *nogood* de P.

Exemple 1 (CSP simple) Soient :
- $\mathcal{V} = \{x, y, z, w\}$
- $\mathcal{D}_x = \mathcal{D}_y = \mathcal{D}_z = \mathcal{D}_w = \{0, 1\}$
- $\mathcal{C} = \{x = y, x + z = y, x + y = 2, x + w = 2\}$

Il existe une unique solution pour ce CSP à 4 variables : $\{x = 1, y = 1, z = 0, w = 1\}$ La configuration partielle $s = \{x = 0, y = 1\}$ est un *deadend*, donc un *nogood*. Cependant, s n'est pas une *nogood minimal* car le sous

ensemble $\{x = 0\}$ de s est aussi un *nogood*. L'instanciation partielle $\{x = 0\}$ ne peut être étendue à une solution, donc $\{x = 0\}$ est bien un *nogood minimal*.

2.1.2 L'optimisation sous contrainte

Dans ce mémoire, résoudre un CSP consiste à trouver la premiere solution réalisable. Dans un cadre plus général, il peut également s'agir de trouver un nombre donné de solutions réalisables ou toutes les solutions. Un problème d'optimisation sous contraintes nécessite quant à lui de trouver une des meilleures solutions réalisables selon une fonction objective. Cette fonction a donc pour rôle d'évaluer la qualité d'une solution réalisable. Les contraintes permettent de déterminer les configurations qui ne sont pas des solutions.

Le problème d'optimisation sous contraintes est alors double : trouver l'ensemble des solutions réalisables ; identifier dans cet ensemble la meilleure solution qui minimise ou maximise la fonction objective.

2.1.3 Méthodes de résolution de CSP

À l'heure actuelle, il n'existe pas de méthode universelle pour résoudre tous les CSPs de manière efficace. Différentes techniques et méthodes ont été développées pour résoudre les CSPs, avec plus ou moins de succès selon la nature du problème traité. Ces techniques sont basées principalement sur les notions de consistance avec les algorithmes de filtrage et de propagation. Les méthodes quant à elles font partie de trois grandes familles, à savoir, les méthodes complètes, incomplètes et les méthodes hybrides :

Les méthodes complètes garantissent l'optimalité de la solution trouvée, mais au prix d'une complexité temporelle souvent rédhibitoire pour les instances de grande taille. Elles vont de la méthode la plus basique qui énumère toutes les combinaisons possibles, jusqu'à des méthodes plus élaborées qui utilisent les notions de consistance et de filtrage. Les méthodes complètes sont basées, en règle générale sur un algorithme de recherche arborescente en profondeur d'abord, avec gestion des retour-arrières, un algorithme de type *backtracking* chronologique [73] qui constitue la pierre angulaire de ces méthodes. Cet algorithme est basé sur l'extension d'une solution partielle en affectant des valeurs à des variables.

Les méthodes incomplètes (ou approchées) mettent en œuvre un processus d'amélioration itérative afin de trouver une affectation satisfaisant le plus grand nombre de contraintes. Elles sont souvent capables de donner une bonne solution en un temps acceptable. Le processus d'amélioration itérative d'une méthode incomplète est basé sur l'étude du voisinage d'une solution courante. L'éventail d'algorithmes incomplets s'étend des algorithmes gloutons et d'améliorations itératives les plus simples, aux métaheuristiques les plus sophistiquées telles la recherche tabou, l'algorithme génétique, le recuit simulé,... . Une méthode incomplète ne peut conclure sur la non réalisabilité d'un problème, mais elle peut être assez rapide pour trouver une bonne solution. Toutefois, elle ne peut garantir la qualité de la solution trouvée ni le temps nécessaire à son obtention. Du fait des garanties qu'elles proposent, les méthodes complètes sont préférables lorsqu'elles sont utilisables.

Les méthodes hybrides combinent des techniques issues de la programmation par contraintes (PPC) avec des méthodes incomplètes comme la recherche locale et les algorithmes génétiques.

Dans la section suivante, nous donnerons un aperçu de ces différentes techniques et méthodes de résolution.

2.1.4 Méthodes de propagation de contraintes

2.1.4.1 Notions de consistance

L'idée principale de la notion de consistance est la réduction de l'espace de recherche des CSPs afin d'en simplifier la resolution de ces derniers. Cette simplification se traduit par la suppression des valeurs inconsistantes des domaines des variables correspondantes. Lors de la résolution, une contrainte C impose l'utilisation de certaines valeurs. La propriété de consistance intervient lorsque une valeur ne peut en aucun cas satisfaire la contrainte C. La propriété de consistance est vérifiée lorsqu'aucune valeur ne peut être supprimée.

Comme dit précédemment, il existe trois type de contraintes : unaire, binaire et n-aire. À chaque type est associé une notion de consistance. La consistance de nœud pour les contraintes unaires, la consistance d'arc (ou l'arc consistance) pour les contraintes binaires et la consistance de chemin ainsi que la k-consistance pour les contraintes n-aires.

2.1.4.2 Consistance de nœud

Soit un CSP $(\mathcal{V}, \mathcal{C})$, la consistance de nœud est atteinte lorsque, pour chaque contrainte unaire $c \in \mathcal{C}$ toutes les valeurs inconsistantes de $dom(V), V \in vars(\mathcal{C})$ ont été supprimées.

Définition 2.1.4.1 (Consistance de Nœud) *Un CSP $(\mathcal{V}, \mathcal{C})$ est dit nœud consistant si pour toute variable $V_i \in \mathcal{V}$ et pour toute valeur $v \in dom(V_i)$, l'instanciation (V_i, v) satisfait toutes les contraintes.*

2.1.4.3 Consistance d'arc

La consistance d'arc (ou l'arc-consistance) est la notion de consistance la plus largement utilisée. Un CSP binaire P est arc-consistant si et seulement si pour tout couple de variables voisines $\{V_i, V_j\}$, pour toute valeur $v_i \in dom(V_i)$ il existe au moins une valeur $v_j \in dom(V_j)$ respectant la contrainte liant V_i et V_j. Plus formellement, l'arc-consistance est définie de la façon suivante :

Définition 2.1.4.2 (Consistance d'Arc) *Soit P une instance CSP, C une contrainte binaire de $\mathcal{C}$, V une variable de $vars(C)$ et v une valeur de $dom(V)$. C est dite arc-consistance si et seulement si pour toute variable V de $vars(C)$ et pour toute valeur v de $dom(V)$, (V, v) est consistante par rapport à C. P est arc-consistant si et seulement si toute contrainte de P est arc-consistante.*

2.1.4.4 Consistance d'hyper-arc

La définition de consistance d'arc a été étendue au CSP n-aires sous l'appellation **consistance d'arc généralisée** (GAC) [118, 120] ou consistance d'hyper-arc. De la même maniére qu'un CSP binaire, un CSP n-aire est arc consistant si pour toute variable V impliquée dans une contrainte C et toutes les valeurs du domaine $dom(V)$, il existe au moins un n-uplet support pour cette contrainte C.

Définition 2.1.4.3 (Consistance d'hyper-arc) *Soit P une instance CSP, C une contrainte n-aire de $\mathcal{C}$, $(V_1, \cdots, V_n)$ l'ensemble des variables appartenant à $vars(C)$. C est dite hyper-arc consistante si $\forall v_j \in dom(V_i)$, il existe un n-uplet support pour cette contrainte C. P est hyper-arc consistant si toutes les contraintes le sont.*

2.1.4.5 La k-consistance

La k-consistance est une généralisation de toutes les notions de consistantes précédentes.

Définition 2.1.4.4 (k-consistance) *Soit un CSP* $(\mathcal{V}, \mathcal{C})$, $\mathcal{W} \subseteq \mathcal{V}$ *l'ensemble des variables telle que* $|\mathcal{W}| = k-1$. *Une configuration partielle* I *sur* $\mathcal{W}$ *est dite k-consistante si et seulement si pour toute* $k^{éme}$ *variable* $v_k \in \mathcal{V}/\mathcal{W}$, *il existe une valeur* $v_k \in dom(V_k)$ *telle que* $I \cup \{(V_k, v_k)\}$ *est consistante localement. Un CSP est k-consistant si et seulement si pour tout ensemble* $\mathcal{W}$ *de* $k-1$ *variables, toute configuration partielle sur* $\mathcal{W}$ *est k-consistante.*

À partir de la definition précédente, nous pouvons remarquer que la K-consistance est un cas general avec :
– $k = 1$: Consistance de nœud
– $k = 2$: Consistance d'arc

2.1.4.6 Algorithmes de filtrage et de propagation de contraintes

De nombreuses propriétés de consistance locale ont été exploitées dans la résolution des problèmes de satisfaction de contraintes (CSP). L'objectif est de réduire l'espace de recherche et par conséquent d'améliorer les méthodes de résolution. Les techniques de filtrage par consistance locale exploitent ces propriétés. Elles permettent de retirer des valeurs inconsistantes des domaines des variables, permettant ainsi de réduire l'espace de recherche. Elles sont utilisées comme pré-traitement lorsque qu'elles détectent et retirent des inconsistances locales avant que la recherche ne commence, ou pendant la recherche pour élaguer l'arbre de décision. La littérature CSP offre un nombre assez important d'algorithmes de consistance locale, néanmoins, le plus utilisé d'entre eux reste la consistance d'arc, qui est associé au contraintes binaires.

Comme dit précédemment, il existe trois type de contraintes : unaire ; binaire et n-aire. À chaque type est associé une notion de consistance. Il en est de même pour les algorithmes de filtrage qui reposent sur le type de contraintes utilisées. À chaque type est associé une famille d'algorithme de filtrage :

– Algorithme de filtrage par consistance de nœuds,
– Algorithmes de filtrage par arc-consistance,
– Algorithmes de filtrage par consistance d'arc généralisée,
– etc...

L'algorithme de filtrage associé aux contraintes unaires (voir Algorithme 3) reste une bonne entrée en matière pour la compréhension du mécanisme de filtrage. C'est un algorithme très simple à comprendre et à mettre en œuvre. L'idée principale est de supprimer pour une contrainte donnée C, qui porte sur une variable V_i l'ensemble des valeurs appartenant à $dom(V_i)$ non consistantes avec la contrainte C.

Algorithme 3: ConsistanceDeNœud$(\mathcal{V}, \mathcal{C})$

```
pour tout contrainte unaire C ∈ 𝒞 faire
  pour tout V ∈ vars(C) faire
    pour tout v ∈ dom(V) faire
      si (V, v) non consistante avec C alors
        supprimer v de dom(V)
      finsi
    fin pour
  fin pour
fin pour
```

Pour atteindre l'arc-consistance, il est nécessaire de vérifier les couples de valeurs possibles. La procédure Revise-arc (voir Algorithme 4), extraite de [108], permet d'atteindre pour un arc donné c_{ij} la propriété de consistance.

Algorithme 4: Revise-arc(C_{ij})

```
delete = faux
pour tout v ∈ dom(V_i) faire
  si ∀w ∈ dom(V_j), (v, w) ne vérifie pas C_ij alors
    Supprimer v de dom(V_i)
    delete = vrai
  finsi
fin pour
return delete
```

Une seule application de la fonction *Revise-arc* n'est pas suffisante pour garantir la consistance de chaque arc. En effet, la suppression d'une valeur peut entraîner la suppression d'autres valeurs dans d'autres domaines. Ainsi, il est nécessaire de réviser des arcs précédemment visitès afin de propager les suppressions de valeurs antérieures.

La consistance d'arc est la consistance locale la plus largement utilisée. Son efficacité a favorisée l'apparition de nombreux algorithmes, dont les algorithmes basiques AC1, AC2 et AC3 proposés par Mackworth [108, 109]. Le plus efficace d'entre eux, l'AC3 subit la concurrence d'algorithmes tel que AC4[119], AC6 [14] et AC7 [25]. Sa simplicité de mise en œuvre et son efficacité relative lui ont assuré une place de choix, d'autant plus que les récentes améliorations lui assurent une certaine pérennité. Ces amélioration correspondent aux algorithmes AC2000 [18] et AC2001/AC3.1 [18, 167], $AC3_d$ [158], AC3.2 [28] et AC3.3 [28].

L'AC-1 (voir Algorithme 5) est un algorithme très simple à appréhender. Sa simplicité se traduit par une révision systématique de toutes les contraintes en dépit du fait qu'elles ne soient pas affectées par une révision antérieure (suppression de valeurs). Cette simplicité se traduit par une complexité de $O(nmd^3)$, qui le rend inefficace. La complexité d'AC-1 est donnée par le nombre d'appels à la fonction *Revise-arc*.

Algorithme 5: AC-1($(\mathcal{V}, \mathcal{C})$)

```
Q = all C_ij ∈ C
répéter
  change = false
  pour tout C_ij ∈ Q faire
    change = Revise-arc(C_ij)
  fin pour
jusqu'à change == true
```

L'AC-3 (voir Algorithme 6) se distingue de l'AC-1 par sa capacité à n'appliquer la procédure *Revise-arc* qu'aux arcs susceptibles d'être affectés par une révision effective (suppression de valeurs). Pour ce faire, l'AC-3 utilise une pile Q contenant les arcs en attente de traitement. Cette pile Q ne contient qu'un exemplaire unique de chaque arc devant être traité.

Un certain nombre d'améliorations, visant à réduire la complexité théorique et le temps de calcul effectif ont été apportées à l'AC3.

AC-4 introduit la notion de support mais avec un comportement en moyenne moins bon que AC-3. AC-4 sauvegarde pour chaque valeur, ses supports dans les autres domaines. Une valeur est supprimée si elle

Algorithme 6: AC-1($(\mathcal{V}, \mathcal{C})$)

Q = all $C_{ij} \in \mathcal{C}$
tantque Q is not empty **faire**
  $C_{ij} = Q.pop()$
  si Revise-arc(C_{ij}) **alors**
    $Q = Q \cup C_{mi}$,C_{mi} l'ensemble des contraintes impliquant la variable V_i
  finsi
fin tantque

n'a plus de support. Une fois supprimée, cette dernière est ajoutée dans une file afin de propager sa suppression.

AC-5 améliorent la complexité spatiale d'AC-4 (i.e., en $O(md)$) pour des types de contraintes particulières mais se réduit à AC-3 ou AC-4 dans le cas général.

AC-6 combine des principes issus d'AC-3 et AC-4. AC-6 sauvegarde seulement un support pour chaque valeur, ce qui lui permet de réduire la complexité spatiale engendrée par AC-4. Les valeurs sont supposées ordonnées.

AC-Inference est une amélioration d'AC-6. Il utilise des métaconnaissances pour réduire le nombre de tests.

AC-7 est un cas particulier d'AC-Inference. Il exploite le fait que si (x, a) est supporté par (y, b) alors (y, b) est supporté par (x, a). Ainsi, AC-7 est capable d'économiser quelques verifications par rapport à AC-6 tout en gardant les mêmes complexités spatiales et temporelles. Une bonne implémentation de cet algorithme permet d'atteindre des complexités linéaires intéressantes (i.e., complexité spatiale dans le pire des cas en O(ed)).

AC-8 exploite les supports minimaux, comme pour AC-6 mais sans les mémoriser.

AC-2000 et AC-2001/3.1 sont basés sur l'AC-3. AC-2000 est une modification de l'AC-3. AC-2001 (ou AC-3.1) est aussi basé sur l'AC-3 mais exploite des concepts utilisés en AC-6 lui permettant de réduire la complexité temporelle.

Autres beaucoup d'autres algorithmes d'arc-consistance ont été proposés dans la littérature comme AC-3.3 qui est une amélioration d'AC-3, AC-3d et AC-3.2 qui améliore AC-3, etc.

La définition de consistance d'arc a été étendue aux CSPs n-aires sous l'appellation **consistance d'arc généralisée** (GAC) [118, 120]. Quelques travaux ont été réalisés sur des algorithmes de filtrage par consistance d'arc généralisée. [108] et [118, 120] ont proposé respectivement les algorithmes CN et GAC-4. L'algorithme CN est une généralisation de l'AC3 alors que GAC-4 est une généralisation de l'AC4. L'utilisation de ces algorithmes reste très limitée du fait de leur complexité élevée. Bessière et Régin ont proposé un algorithme général appelé **GAC-schema** [16], basé sur l'AC7. Il permet de traiter des contraintes d'arité quelconque exprimées de différentes manières : en extension, sous forme d'expressions arithmétiques ou encore sous forme de prédicats sans sémantique particulière. Cet algorithme est plus performant que GAC-4 du fait qu'il ne mémorise que les supports utiles. Par la suite, dans [17], Bessière et Régin ont proposé une amélioration de cet algorithme. Dans cette nouvelle version, les petites contraintes (de faible degré) sont regroupées, formant ainsi des contraintes plus larges qui sont par la suite considérées comme des sous problèmes. Pour ces sous problèmes il existe des moyens de filtrage par consistance d'arc plus rapides que sur les contraintes initiales.

Ces techniques sont généralement très lourdes. Très peu d'algorithmes de résolution utilisent la consistance d'arc généralisée. Cependant, des tests de consistance ont été développés spécifiquement pour certains types de contraintes n-aires. Ils sont plutôt rapides et aussi plus efficaces que les contraintes n-aires binarisées. La plus célèbre est la contrainte globale **all-different** (conjonction de contraintes de différences). La consistance de cette contrainte [138] est assurée par la recherche d'un couplage maximal dans un graphe.

Pour plus de détails sur les algorithmes de filtrage, nous recommandons la lecture de [15, 107, 38]. Dans la section suivante, nous proposons une synthèse très compacte sur les algorithmes de résolution des CSP.

2.1.5 Méthodes complètes

Les méthodes complètes (ou exactes) sont capables de fournir une solution réalisable ou toutes les solutions réalisables lorsqu'elles existent. Elles sont également capables de conclure à la non satisfiabilité d'un CSP (aucune solution n'est trouvée à la fin de la recherche). Ce type de méthode s'appuie sur une recherche systématique, en opposition aux méthodes incomplètes. L'algorithme de recherche arborescente *backtracking* est l'algorithme de base de toutes ces méthodes. Du fait de la complexité inhérente à une recherche arborescente exhaustive, de nombreuses améliorations ont été proposées. Ces améliorations permettent principalement de réduire l'espace de recherche par propriété de consistance locale avant la recherche d'une solution (en prétraitement) ou pendant la recherche avec du maintien de consistance, par des heuristiques de choix de variables/valeurs et par des améliorations de l'algorithme de base, le *backtracking* par la mise en place de *backtrack intelligent* (retour arrière intelligent). Ces améliorations ont donné naissance à un certain nombre d'algorithmes. On peut citer des méthodes de retour arrière non chronologique tel que le *Backjumping* (BJ) [64, 65, 63] et le *Conflict Directed Backjumping* (CBJ) [132, 66], des méthodes d'apprentissage permettant de filtrer l'espace de recherche tel que le *Backmarcking* (BM) [64], le *Dynamic Backtracking* [68], le *Learning* [43] et le *NoGood recording* [146], sans oublier les méthodes effectuant un filtrage des domaines des variables tout au long de la recherche, afin de rendre le problème réduit partiellement (ou localement) consistant. C'est le cas notamment des algorithmes comme le *Forward-Checking* [83] ou le *Maintien d'Arc Consistance* (MAC) [63, 144], qui maintiennent à chaque nœud de l'arbre, une forme partielle ou complète de constance d'arc. Ces méthodes peuvent appliquer différents filtrages, en fonction du type de consistance considérée.

2.1.5.1 Recherche systématique

La méthode *Generate and test*, notée GT est la pierre angulaire des méthodes de résolution exactes. Elle consiste en une énumération exhaustive de toutes les instanciations complètes possibles et la verification de leurs consistances avec toutes les contraintes. Il en découle une complexité temporelle exponentielle. Le *backtracking*, notée BT constitue une premiere amélioration de GT, ceci en ne développant que les instanciations consistantes. BT se déroule en deux phase :

- Marche avant : consiste dans la selection de la prochaine variable à instancier et en l'instanciation de cette dernière à une valeur consistante de son domaine, si une telle valeur existe. Dans le cas où une telle valeur n'existe pas, un retour arrière est effectué.
- Retour arrière : lors d'un dead-end, un retour arrière est effectué au point de choix précédent, c'est-à-dire à la dernière variable instantiée afin de modifier son affectation courante.

Le *backtracking* est très simple de mise en œuvre. Cette simplicité est obtenue au détriment d'une complexité temporelle exponentielle, qui est accentuée par deux principaux défauts : le phénomène de *trashing* qui se traduit par la redécouverte de manière répétitive des mêmes inconsistances dues à une affectation antérieure issue d'un *no-good* minimal ; la découverte redondante des mêmes configurations partielles inconsistantes. Ces défauts sont dues à un parcours aveugle de l'espace de recherche sans prise en compte d'informations évidentes sur la non consistance globale d'instanciations partielles et la découverte redondante des mêmes configurations partielles inconsistantes. La figure 2.1 illustre le schéma général du *backtracking*

Une voie d'amélioration évidente du *backtracking* est de modifier ce dernier de sorte qu'il puisse détecter, à moindre coût et le plus rapidement possible la non consistance globale de l'instanciation courante. On sait alors qu'il n'est pas possible d'aboutir à une solution dans la branche courante de l'arbre et on peut continuer l'exploration dans une autre branche. Les différentes stratégies adoptées peuvent être classées dans deux grandes

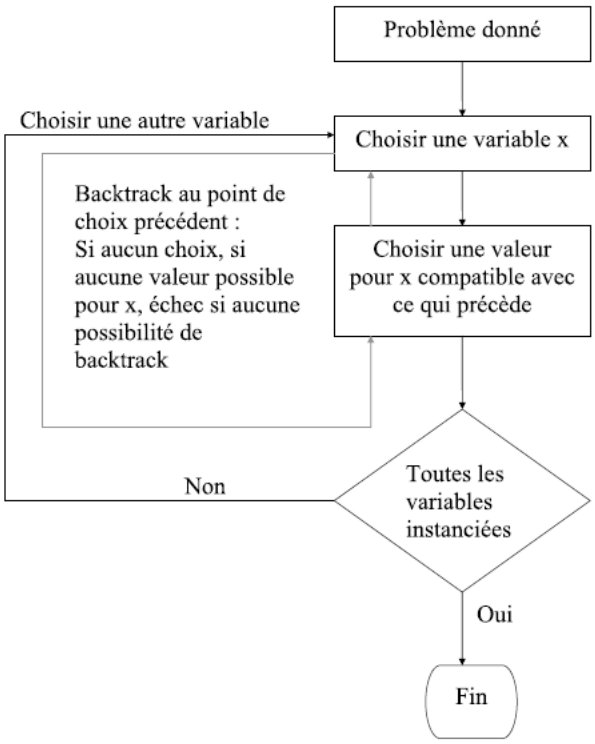


FIGURE 2.1 – Schéma général du Backtracking

catégories : les techniques dites prospectives *Look-Ahead* ou rétrospective *Look-Back*. Appliquées aux algorithmes de recherche énumérative avec retour arrière (de type *backtracking*), ces algorithmes suivent soit un schéma de parcours prospectif *Look-Ahead*, soit un schéma rétrospectif *Look-Back*, soit une combinaison des deux [51]. Le schéma prospectif est adopté lors de la phase d'extension de la solution partielle, alors que le schéma rétrospectif est quant à lui adopté lors du retour arrière suite à un phénomène de *dead-end*. Le principal intérêt de ces deux schémas est de réduire la redécouverte de manière répétitive des mêmes instanciations partielles inconsistantes, et le phénomène de *trashing* qui se traduit par la reproduction d'un même échec dû à une affectation antérieure issue d'un *nogood* minimal.

2.1.5.2 Schéma prospectif

Le Schéma prospectif est mis en œvre par le biais d'heuristiques de sélection de la prochaine variable à instancier et par la propagation des instanciations. Ceci lui permet de vérifier la présence ou non d'une impasse. Les principaux algorithmes adoptant un schéma prospectif sont le *Forward-Checking* [83] et le Maintien d'Arc Consistance (MAC)[144]. Le filtrage appliqué par le *Forward-Checking* est très semblable au filtrage par consistance d'arc. Quand le filtrage par consistance d'arc effectue la propagation à toutes les contraintes, jusqu'a ce qu'il n'y est plus de modification, le *Forward-Checking* limite la propagation aux contraintes qui lient la dernière variable instantiée. En revanche, le Maintien d'Arc Consistance, comme son nom l'indique restaure la propriété de consistance d'arc après chaque instanciation. La figure 2.2 illustre le schéma général d'un algorithme utilisant la propagation de contraintes, comme le **Forward-Checking**

Forward-Checking l'algorithme exploite le principe de la recherche énumérative avec retour arrière, avec un filtrage limité permettant de réduire la combinatoire de l'arbre de recherche. Ce filtrage consiste, après chaque nouvelle assignation de variable à restaurer la consistance d'arcs des contraintes concernées par

la variable nouvellement assignée. En d'autres termes, le filtrage par propagation permet de supprimer, des domaines concernés les valeurs qui sont incompatibles avec la dernière assignation. En cas de retour arrière, les modifications occasionnées par le filtrage sont annulées.

MAC cet algorithme est également basé sur le principe de l'exploration d'un arbre de recherche avec retour arrière, mais il effectue un filtrage plus poussé en restaurant la consistance d'arcs à chaque nœud de l'arbre de recherche, e.i. à la racine et après chaque assignation, pour toutes les contraintes et ce jusqu'a ce qu'il n'y est plus de modifications.

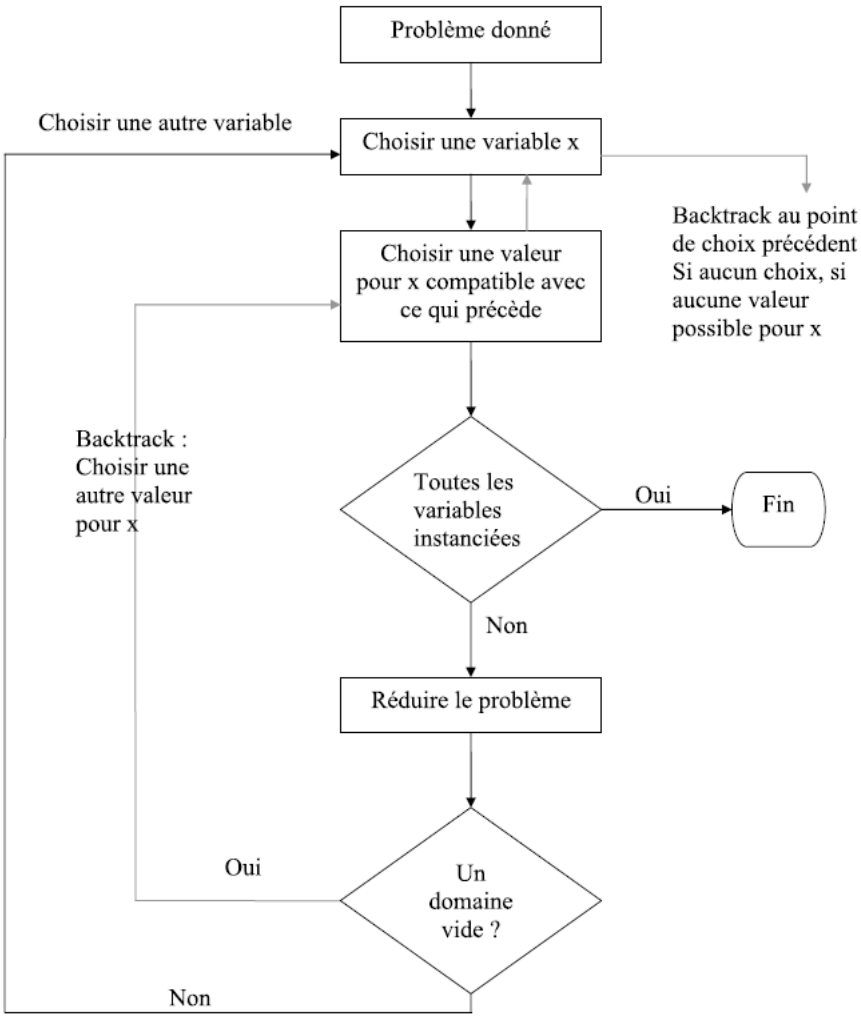


FIGURE 2.2 – Schéma général d'un algorithme utilisant la propagation de contraintes

Dans un algorithme de retour arrière, l'ordre d'instanciation des variables est determinant car l'application de différentes heuristiques de choix de variables pour résoudre une même instance peut conduire à des résultats extrêmement variés en terme d'efficacité [19]. Plus encore,[39] ont montré que l'introduction d'un processus aléatoire au niveau de l'heuristique de choix de variables engendre de grands écarts de performance. Une heuristique idéale consisterait à sélectionner en priorité les variables qui, une fois assignées rendent le problème facile à résoudre. Cet ensemble de variables se nomme un *backdoor* [136]. Les principales heuristiques peuvent être soit statiques (SVOs pour Static Variable Ordering), ou dynamique (DVOs pour Dynamic Variable Ordering).

Une heuristique statique conserve le même ordre de priorité tout au long de la recherche, ordre établi à partir de l'etat initial du problème. Les heuristiques les plus courantes sont :

lexico les variables sont ordonnées lexicographiquement,

deg les variable sont ordonnées de manière décroissante en fonction de leur degré initial [53]. Le *ddeg* est quant à lui fonction du degré courant (dit dynamique),

width les variables sont ordonnées de sorte que la largeur du graphe de contrainte soit minimale [59].

Une heuristique dynamique quant à elle prend en compte l'évolution du problème en considérant différentes informations de son état courant. L'heuristique la plus connue, *dom* [83], consiste à donner la priorité aux variables ayant la plus petite taille de domaine. L'utilisation de cette heuristique est justifiée par le principe fail-first : ≪ *to succeed, try first where you are most likely to fail* ≫ . [13] et [13, 150] proposent respectivement les heuristiques *dom/deg* et *dom/ddeg* qui combinent la taille du domaine courant et, le degré initial (*deg*) pour l'un et le degré courant (*ddeg*) pour l'autre. Ces combinaisons peuvent significativement améliorer l'efficacité de la recherche.

L'ordre établi par les heuristiques de choix de variables fait apparaître des sous-ensemble de variables équivalents (*tie*), c'est-à-dire des variables pour lesquelles il y a une égalité du critère de sélection (*dom, dom/deg, dom/ddeg...*). Une heuristique de choix peut être implicitement utilisée pour départager ces variables. Ainsi l'heuristique *dom* correspond en réalité à l'heuristique $dom \oplus lexico$, qui sélectionne, parmi les variables équivalentes en terme de taille de domaine, la premiere variable rencontrée dans l'ordre lexicographique. D'autre part, une heuristique de choix peut être implicitement indiquée pour départager les variables équivalentes, en revanche, l'ordre lexicographique sera toujours implicitement utilisé en dernier recours. Parmi les heuristiques de ce type, on peut citer $dom \oplus deg$ [60] et $dom \oplus ddeg$ [41, 149].

Un travail très intéressant à été proposé par [24], concernant une généralisation de l'ensemble des heuristiques de choix de variable existantes. Cette généralisation considère que chaque variable peut être évaluée en fonction de son voisinage. Le lecteur peut se référer à [19] pour plus de détails.

Afin d'évaluer l'impact des différentes heuristiques de choix de variable, de nombreuse expérimentations ont été effectuées. Il en résulte qu'aucune heuristique ne surclasse les autres. Néanmoins, il est généralement admis que les heuristiques dom, *dom/ddeg* et $dom \oplus deg$ sont les plus efficaces.

2.1.5.3 Schéma rétrospectif

Le schéma rétrospectif permet de tirer parti des informations implicites contenues dans les situations d'échec afin de limiter la redécouverte de manière répétitive des mêmes instanciations partielles inconsistantes, et le phénomène de *trashing*.

Deux catégories d'algorithmes, assez proches rentrent dans ce cadre général :

- Le Backtrack intelligent qui, lors d'un *dead-end* exploite le fait qu'une instanciation partielle s incluse dans l'instanciation partielle courante, est non globalement consistante en effectuant un retour-arrière non chronologique qui supprime de l'espace exploré les instanciations partielles étendue à partir de l'instanciation s
- La mémorisation de contraintes (aussi appelée apprentissage, learning en anglais) mémorise le fait que s est non globalement consistante sous la forme d'une contrainte interdisant s. Cette mémorisation s'ajoute au retour-arrière intelligent. Il faut alors faire un judicieux compromis entre la mémoire consommée et les effets bénéfiques de la mémorisation

Soit une instanciation partielle courante non consistante. Cette instanciation a connu un *dead-end* lors de son extension à la variable V_i. Le *backtracking* consiste à effectuer un retour arrière à la variable précédente V_{i-1}. Supposons que les deux variables ne sont liées par aucune contrainte, donc V_{i-1} ne peut être responsable de l'inconsistance rencontrée sur la variable V_i et cette inconsistance sera retrouvée quelque soit la valeur affectée à la variable V_{i-1}. Une amélioration consiste alors à identifier l'instanciation non globalement consistante et à

effectuer un saut arrière jusqu'à cette instanciation. Cette identification est basée sur la notion de *conflict-set* (ensemble de conflit).

Le *backjumping* notée BJ améliore l'algorithme du *backtracking*. Elle identifie la variable responsable de l'échec grâce au maintien d'un *conflict-set*, le plus petit possible. Pour l'instanciation partielle courante, il s'agit du plus petit sous-ensemble de variables (d'instanciations) interdisant à la variable courante d'être instantiée. Une fois la variable responsable du *dead-end* identifiée, le *backjumping* effectue une restauration de l'espace de recherche entre le point de retour et l'instanciation courante : toutes les instanciations entre le point de retour et l'instanciation courante sont annulées.

Lors d'un *dead-end*, au moment de l'instanciation de la variable V_i, un *conflict-set* possible serait l'ensemble des variables voisines de V_i précédemment instantiées. Le retour arrière est effectué à la variable la plus récente dans le *conflict-set*, garantissant ainsi l'exactitude et la convergence de la méthode.

Les différents algorithmes de *backjumping* se distingue par la manière de constituer le *conflict-set*. On peux citer *Backjumping de Gashnig* [45], *Graph-based backjumping* [44]ou encore le *Conflict Directed Backjumping* notée CBJ [132, 66], et bien d'autres.

Le *dynamic backtracking* notée DBT [69] constitue une amélioration intéressante du *backjumping*. Contrairement à ce dernier qui effectue une restauration de l'espace de recherche entre le point de retour et l'instanciation courante, le DBT sauvegarde ces instanciations.

Les algorithmes prospectifs et rétrospectifs ont pour principal objectif de réduire l'espace de recherche grâce à un mécanisme de propagation efficace et une technique de retours arrière intelligent. L'association des deux schémas a donc tout naturellement été envisagée et étudiée.

2.1.5.4 Look-ahead + Look-back

De nombreux travaux sur l'association des algorithmes look-back et look-ahead ont été réalisés. Il en a résulté de nouvelles méthodes combinant la propagation de contraintes avec des techniques de retours intelligents, telque :
– MAC et CBJ [133]
– FC et DBT [145]
– MAC et DBT [122]
L'hybridation ne garantit pas forcement un gain en terme d'efficacité et de temps de résolution. Le plus souvent, le choix de la méthode est fortement conditionné par les caractéristiques du problème traité (structure et propriétés). Ainsi, [13] a montré que MAC est souvent plus rapide que MAC et CBJ. Néanmoins, [122] a montré que MAC-DBT fournissait de très bonnes performances sur différents CSPs (grande et petite taille).

Les performances d'une hybridation sont fortement liées au compromis entre la réduction de l'espace de recherche (le nombre de nœuds traités) et le temps de traitement pour chaque nœud. Il est également primordial d'étudier la structure du problème afin de déterminer la meilleur combinaison possible entre heuristique d'ordonnancement, méthodes de propagation / de filtrage, méthodes de retour arrière et d'apprentissage.

2.1.5.5 Méthodes de résolution exploitant l'Apprentissage (Learning algorithms)

Les algorithmes de *backjumping* améliorent considérablement le *backtracking*, cependant ils n'empêchent pas la redondance de la recherche. Sauvegarder un certain nombre d'informations concernant les recherches précédentes permettrait d'éviter cette redondance. Cela consiste à enregistrer des informations déduites des branchements infructueux. Ainsi est née la notion d'apprentissage et les algorithmes qui l'exploitent : learning algorithms. Lors d'une recherche arborescente de type *backtracking*, chaque *deadend* rencontré est un *nogood*. La découverte

de ces *nogoods* est une partie intégrante de la recherche d'une solution, mais leur identification cause un effort de recherche inutile dans le cas ou une solution existe. Phénomène accentué par la redondance de la recherche, c'est-à-dire la redécouverte des mêmes *nogoods*. Stallman et Sussmann [153] ont apporté une réponse à cette difficulté en introduisant une technique permettant d'exploiter ces *nogoods* en les ajoutant au CSP sous forme de nouvelles contraintes. Les contraintes ajoutées permettent de réduire l'espace de recherche sans affecter l'espace des solutions. Cette technique est connue sous le nom de *Constraint Recording* ou *Nogood Recording*.

[153] proposent de sauvegarder un *nogood* à chaque *deadend* rencontré. Ce qui a pour effet de générer un nombre de *nogood* tel que la réduction de l'espace de recherche induite devient inintéressante face au coût d'exploitation de ces *nogood*.

Face à cette difficulté, les chercheurs ont proposé différentes méthodes permettant de réduire le nombre de *nogoods* stockés pendant la recherche. Par exemple, [44] propose de ne stocker que les *nogoods* contenant i affectations, de taille i. Le paramètre i doit être judicieusement sélectionné suivant les propriétés du problème. Contrairement à [44], [68] propose de stocker tous les *nogoods*, mais, toujours dans l'optique d'en réduire le nombre, il propose de supprimer ceux qui ne sont plus pertinents. [94] proposent quant à eux d'intégrer les deux idées précédentes : tous les *nogoods* sont enregistrés, seuls ceux qui ne sont plus pertinents et qui ont une taille plus grande qu'une certaine valeur i sont supprimés.

L'intégration des méthodes d'apprentissage de type *Nogood Recording* au processus de recherche d'une solution, notamment dans le mécanisme de propagation des contraintes est très coûteux. Toutes les expérimentations menées par les chercheurs sur l'enregistrement des *nogoods* tendent vers cette conclusion.

L'apprentissage lors de la résolution ne passe pas obligatoirement par l'exploitation des *nogoods*. Un bon exemple est la méthode proposé par Lecoutre, nommé Last Conflict reasoning (LC). La méthode LC [29] se base sur les conflits pour guider le développement de l'arborescence de recherche vers les variables à l'origine de l'inconsistance. À chaque étape de la résolution, l'heuristique de choix de variables sélectionne la dernière variable ayant conduit à un *deadend*, si une telle variable existe. Son efficacité a été attestée dans le cadre d'une méthode MAC sur des problèmes aléatoires ainsi que sur des problèmes réels. Cette méthode a été étendue par Grimes et Wallace [74] en introduisant des restarts à la méthode de base.

2.1.6 Méthodes incomplètes

Il arrive que l'utilisation de méthode exacte soit inenvisageable pour certaines classes de problèmes, les problèmes de grande taille. Des problèmes pour lesquels une méthode exacte est dans l'incapacité de fournir une solution réalisable en un temps raisonnable. Cette contrainte de temps concerne par exemple le monde industriel.

Une alternative aux méthodes exactes se trouve être les méthodes incomplètes, qui lorsque une limite de temps de calcul est imposée pour un problème de grande taille, peuvent fournir une solution de bonne qualité en un temps raisonnable. Voici un exemple, extrait de [54], illustrant ce cas de figure : une méthode exacte [105, 115], qui consiste à effectuer une énumération exhaustive en décomposant le problème a été appliquée à différents problèmes d'optimisation comme le problème de coloriage de graphe et testée sur les instances DIMACS [115]. Plusieurs instances ont été résolues en quelques heures de manière optimale. Une de ces instances a été résolue par la méthode exacte en deux heures alors que certaines méthodes approchées trouvent une solution de même qualité en moins d'une minute.

Deux défauts majeurs des méthodes incomplètes concerne leur incapacité à garantir l'optimalité d'une solution et de prouver la non faisabilité d'un CSP.

Le nombre de méthodes et leurs variantes étant extrêmement important, dans les sections suivantes nous ne présentons que quelques méthodes. Pour plus de détails, le lecteur peux se référer à [54, 57, 92, 93].

2.1.7 Recherche gloutonne

La recherche gloutonne [11, 130] est une méthode constructive très basique. Son principe de fonctionnement est l'instanciation successive et definitive des variables, c'est-à-dire sans remise en cause. Le parcours de l'espace de recherche, si l'on peut dire ainsi, est conditionné par les heuristiques de choix de variables et de valeurs. Cette méthode a l'avantage d'être de complexité linéaire en fonction du nombre de variables et de la taille des domaines. Cependant, elle s'avère peu efficace sur des problèmes de grande taille. Pour cette classe de problème, on lui préférera le glouton stochastique itéré [20], qui, en effectuant plusieurs recherches gloutonnes successives avec un bruitage des heuristiques de choix de variables et de valeurs, arrive à parcourir l'espace de recherche de manière plus efficace. Ainsi, le glouton stochastique itéré est plus efficace que la recherche gloutonne sur des instances de grande taille.

2.1.8 Métaheuristiques

Une métaheuristique est un algorithme d'optimisation visant à résoudre des problèmes difficiles. Des problèmes souvent issus de la recherche opérationnelle, de l'ingénierie ou de l'intelligence artificielle et pour lesquels on ne connaît pas de méthode classique plus efficace. Elles manipulent une ou plusieurs configurations complètes (solutions) à la recherche de l'optimum. Les itérations successives doivent permettre de passer d'une configuration complète de mauvaise qualité à une bonne solution. L'arrêt de l'algorithme est conditionné par un critère prédéfini : un nombre d'itérations, un temps d'exécution, la solution trouvée est suffisamment bonne, etc.

Il existe un grand nombre de métaheuristiques différentes, allant de la simple recherche locale à des algorithmes plus complexes comme les algorithmes génétiques. Ces méthodes utilisent cependant un haut niveau d'abstraction leur permettant d'être adaptées à une large gamme de problèmes différents. Les métaheuristique peuvent être classées dans deux catégories :

Parcours Ces algorithmes partent d'une solution initiale obtenue de façon exacte, ou par tirage aléatoire. Par la suite, elles s'en éloignent progressivement pour réaliser une trajectoire, un parcours progressif dans l'espace des solutions. On peux citer le Recuit Simulé, la méthode Tabou ou encore la Recherche par Voisinage Variable. Le terme de recherche locale est de plus en plus utilisé pour qualifier ces méthodes.

Population (ou évolutionnaire) Ce type d'approche manipule simultanément une population de configurations complètes. On peux citer les algorithmes génétiques, l'optimisation pas essaim particulaire, etc.

Contrairement aux méthodes constructives, la plupart des méthodes de recherche locale [1] manipulent des configurations complètes qu'elles modifient grâce à des mouvement locaux afin de rechercher une meilleure solution. La recherche locale se base donc sur l'idée d'amélioration d'une configuration complète initiale, généralement ni consistante, ni optimale en faisant des changements sur les affectations afin d'obtenir une configuration voisine consistante et optimale. La première étape est la construction d'une configuration de départ. Cette étape est simple, on utilise généralement une configuration aléatoire ou des algorithmes gloutons pour construire une configuration de départ. Ensuite, on entre dans un processus itératif qui consiste à remplacer la configuration courante par une configuration qui lui est voisine, ce qui permet de visiter l'espace de recherche. La notion de voisinage est alors primordiale.

Définition 2.1.8.1 *Soit s une configuration complète, le voisinage de s, noté $V(s)$, est un sous-ensemble de configurations complètes directement atteignable à partir d'une transformation de s. Soit s' une telle configuration, s' est alors dite voisine de s et $s' \in V(s)$.*

Les changements d'affectations permettant d'atteindre une meilleur solution conduisent généralement le processus de recherche vers un minimum local. Les métaheuristiques permettent de s'écarter de ces optimums locaux pour atteindre un optimal global.

Nous avons choisi de présenter quelques méthodes qui nous paraissent pertinentes pour exposer les principes et les mécanismes de la recherche locale

Le Recuit Simulé (en anglais Simulated Annealing) Proposé par trois chercheurs de la société IBM en 1983 [142], le recuit simulé est une méthode de recherche locale inspirée d'un processus physique en métallurgie, le recuit des métaux, basé sur le refroidissement lent des atomes permettant de maximiser la dureté du metal. Le recuit simulé est basé sur l'acceptation systématique d'un mouvement local améliorant la solution courante et l'acceptation, selon un critère d'un mouvement local dégradant la solution. Ce critère exprime le niveau de dégradation de la solution courante par rapport à la meilleure solution trouvée. Le niveau de dégradation dépend d'un paramètre température qui décroit au cours du temps. Le niveau de dégradation est important au début pour favoriser l'exploration et progressivement de plus en plus faible pour intensifier la recherche sur l'optimum local. Cet algorithme est toujours très utilisé notamment dans le domaine des réseaux.

La Recherche Tabou Proposée par Glover [70, 71] et Hansen [82], c'est une méthode de recherche locale qui construit un voisinage excluant un certain nombre de configurations récemment visitées ou de mouvements récemment utilisés. La méthode permet de gérer la redécouverte de configurations en les interdisant complètement ou pas selon les options choisies. L'idée initiale consiste à se déplacer d'une configuration à une autre tout en s'interdisant de revenir sur une configuration déjà rencontrée dans un nombre donné d'itérations antérieures (éventuellement depuis le début). Pour se faire, une liste Tabou a été introduite. Elle contient les configurations vers lesquelles il est interdit de se déplacer pendant un certain nombre d'itérations. Cependant, stocker et gérer des configurations entières est coûteux en espace et en temps de calcul. Pour éviter ces inconvénients, la méthode Tabou consiste aussi à interdire un ensemble de mouvements récemment utilisés. Ces mouvements peuvent ramener à une configuration déjà visitée ou non. Pour plus de details le lecteur peux se référer à [54].

La Recherche à Voisinage Variable est une métaheuristique basée l'utilisation de plusieurs voisinages et sur le changement du voisinage au sein d'une recherche locale dès lors que le voisinage courant n'a pas réussi à améliorer la solution courante, que ce soit en descente vers un optimum local ou pour s'échapper de ces optimums locaux.

Les Algorithmes Génétiques Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle : croisement, mutation, sélection, etc. Les algorithmes génétiques ont déjà une histoire relativement ancienne puisque les premiers travaux de John Holland sur les systèmes adaptatifs remontent à 1962 [86]. L'ouvrage de David Goldberg [72] a largement contribué à les vulgariser. Contrairement aux algorithmes précédent qui font évoluer itérativement une unique configuration complète, l'algorithme génétique fait évoluer un ensemble, une population de configurations complètes. La convergences vers une meilleur solution est obtenue grâce aux mécanismes de croisement, mutation, sélection, ... qui interviennent directement sur les affectations des variables des configurations.

2.2 Conclusion

La problématique de validation adopte une modélisation sous la forme d'une succession de Problème de Satisfaction de Contraintes Binaire, résolue grace à une méthode de la Programmation Par Contrainte. La méthode de résolution adoptée est une méthode de recherche arborescente de type depth-first, avec à la fois un schéma prospectif *look-ahead* incluant un mécanisme de propagation, une heuristique dynamique de choix de variables et une heuristique statique de choix de valeurs, et un schéma rétrospectif *look-back*. Le schéma prospectif est mis en œuvre par un algorithme de propagation, le **Forward-Checking** [83] et une heuristique de choix de

variable dynamique $dom/wdeg_{cbj}$. La modélisation adoptée se justifie par le besoin de statistiques précises lors de la validation.

CHAPITRE 3

AMÉLIORATIONS

Dans ce chapitre, nous proposons un ensemble d'améliorations permettant une réduction significative du temps de validation.

3.1 Base algorithmique

Comme dit précédemment, la base algorithmique est une méthode complète sous la forme d'une recherche arborescente de type depth-first, avec à la fois un schéma prospectif *look-ahead* incluant un mécanisme de propagation, une heuristique dynamique de choix de variables et une heuristique statique de choix de valeurs, et un schéma rétrospectif *look-back*.

Le schéma prospectif est mis en œuvre par un algorithme de propagation, le **Forward-Checking** [83] et une heuristique de choix de variable dynamique *dom* [83]. Cette heuristique donne la priorité aux variables de plus petit domaine. En revanche, les valeurs ont un ordre statique basé sur la longueur des chemins associés, avec comme critère les chemins les plus court d'abord, qui potentiellement généreront moins de conflits par rapport à des chemins plus longs. Rappelons ici qu'une variable $V_i \in V$ est associée au canal i et que son domaine $dom[V_i] \in D$ est constitué de tous les chemins possibles reliant le canal i aux tubes amplificateurs.

Dans l'existant, la mise en œuvre de l'algorithme de propagation **Forward-Checking** est particulière et non optimale. Lors de la propagation de l'instanciation courante au nœud i, la réduction du sous-problème inhérent n'est pas persistante au nœud $i+1$, c'est-à-dire que la propagation doit être menée pour toutes les variables précédemment instanciées et la variable courante nouvellement instantiée, cela à chaque nœud.

Le schéma rétrospectif est quant à lui mis en œuvre par le **Conflict Directed Backjumping** (CBJ) [132, 66], basé sur un mécanisme de retour arrière non chronologique qui, lors d'une impasse, permet d'identifier la variable responsable de l'échec. Le *CBJ* est une amélioration du *backjumping* [64, 65, 63]. Ce dernier, lors d'un *dead-end* effectue un retour arrière jusqu'à la variable la plus récente impliquée dans le *dead-end*. Ce retour arrière est rendue possible grâce à la mémorisation de la dernière variable à avoir supprimée une valeur à la variable en échec. Cependant, lors d'une succession d'échec, seul le premier retour en arrière est un *backjump*, les autres retours sont de simple *backtracks*. Contrairement au *backjumping*, lors d'une succession d'échec, le **Conflict Directed Backjumping** effectue des backjumps grâce à la mémorisation d'un ensemble d'explications tout au long de la recherche.

Sur cette base algorithmique, les principales améliorations proposées dans ce travail concernent la phase prospective une nouvelle heuristique de choix de variables et l'utilisation en phase de prétraitement d'un algorithme

de filtrage basé sur la notion de consistance d'arc. Nous proposons également un mécanisme d'apprentissage basé sur la nouvelle heuristique, et des améliorations d'implémentation.

Mais tout d'abord, dans la section suivante, nous allons décrire la structure des CSPs traités dans cette thèse, notamment notre base d'expérimentation.

3.2 Structure des CSPs traités

L'étude de la problématique et l'élaboration des différentes améliorations ont été menées sur la base de trois cas de validation, **A**, **B** et **C** représentant trois niveaux de difficulté. Chaque cas de validation est caractérisé par la matrice de switches associée, le nombre de combinaisons à valider et le nombre canaux actifs qui représente le nombre de variables :

– A : 22 variables et 11 625 120 combinaisons (CSPs)
– B : 21 variables et 25 603 600 combinaisons (CSPs)
– C : 25 variables et 7 312 032 combinaisons (CSPs)

Comme décrit dans la section 1.2.3, la problématique de validation est sujette à une contrainte qui limite le nombre de valeurs par variable. À la fin d'une validation, si des combinaisons non-valides (CSPs non satisfiables) ont été détectées, le processus de validation est relancé pour ces seules combinaisons, mais en incrémentant la borne supérieure de cette contrainte. Ainsi les nouvelles valeurs peuvent potentiellement diminuer le nombre de combinaisons non-valides. Le processus de validation est relancé tant que des combinaisons non-valides sont détectées, sous condition que la borne supérieure de la contrainte n'atteigne pas sa valeur maximale. Il faut noter que le processus de validation peut être lancé avec une borne supérieure égale à sa valeur maximale.

Cette validation incrémentale nous permet de traiter des variantes des cas de validation B et C. Soit B_i et C_i, avec $i = 1 \cdots 4$, des variantes, respectivement de B et de C. Ces variantes résultent de l'ajout de valeurs inhérentes à l'incrémentation de la borne supérieure de la contrainte précédemment exposée. Elles correspondent aux nombres de switches maximum traversés par les chemins.

Le tableau 3.1 illustre, par cas de validation, le nombre de combinaisons, le nombre de combinaisons non-valides et les caractéristiques des CSPs associés aux combinaisons :
– Nombre moyen de valeur, avec une valeur minimale et une valeur maximale
– Densité moyenne du graphe de contrainte, avec une valeur minimale et une valeur maximale
– Dureté moyenne des contraintes, avec une valeur minimale et une valeur maximale

Le cas de validation B_1 a 10 829 485 combinaisons non-valides parmi 25 603 600 combinaisons. À la fin de la validation de B_1, les combinaisons non valides identifiées (10 829 485) vont composer le cas de validation B_2, auquel des valeurs supplémentaires sont ajoutées du fait de l'incrémentation de la borne supérieure de la contrainte limitant le nombre de valeurs par variable. De la même manière, B_3 résulte de B_2, C_2 de C_1 et C_3 de C_2. Les cas de validation B_4 et C_4 correspondent quant à eux, à une validation avec une borne supérieure égale à sa valeur maximale.

Le tableau 3.1 nous permet de constater une hétérogénéité des caractéristiques des CSPs traités, mais avec une certaine tendance correspondant aux CSPs avec un graphe de contrainte très dense, voire complet et une faible dureté des contraintes. Ces caractéristiques nous permettent de définir les CSPs traités comme étant de petite à moyenne taille et d'une difficulté moyenne avec néanmoins une forte proportion de CSPs faciles et une proportion restreinte de CSPs difficiles.

CSPs	Nombre de combinaisons		nbVar	nbVal	Densité	Dureté
	Totale	Non-valide		(moy (min/max))	(moy (min/max))	(moy (min/max))
A_1	11 625 120	240	22	6.62 (5.41/7.82)	0.76 (0.60/0.90)	0.07 (0.06/0.08)
B_1	25 603 600	10 829 485	21	9.0 (7.10/10.33)	0.59 (0.40/0.73)	0.14 (0.12/0.17)
B_2	10 829 485	8 695 364	21	14.0 (11.14/16.71)	0.81 (0.62/0.90)	0.20 (0.17/0.24)
B_3	8 695 364	3 126 532	21	21.23 (16.81/24.95)	0.97 (0.86/1.00)	0.27 (0.24/0.31)
B_4	25 603 600	3 126 532	21	21.96 (16.81/26.95)	0.97 (0.84/1.00)	0.27 (0.24/0.31)
C_1	7 312 032	475 768	25	10.8 (8.88/12.12)	0.67 (0.37/0.75)	0.116 (0.09/0.13)
C_2	475 768	294 354	25	19.4 (16.6/21.31)	0.87 (0.63/0.93)	0.17 (0.15/0.19)
C_3	294 354	23 466	25	34.4 (27.2/37.5)	0.99 (0.93/1.0)	0.25 (0.22/0.26)
C_4	7 312 032	23 466	25	33.7 (27.2/37.6)	0.99 (0.86/1)	0.25 (0.22/0.26)

TABLE 3.1 – Structures des différents CSP

3.3 Phase prospective

L'un des points d'amélioration d'un algorithme de recherche arborescente de type retour arrière est l'ordre d'instanciation des variables, un ordre considéré comme crucial depuis l'avènement de ce type d'algorithme. Les heuristiques généralement utilisées sont *dom* [83], *dom/deg* [13], *dom/ddeg* [13, 150], $dom \oplus deg$ [60] et $dom \oplus ddeg$ [149, 41]. Ces heuristiques exploitent un certain nombre d'informations sur l'état courant de la recherche, telle que la taille des domaines (*dom*) ou le degré des variables (*deg* dans le cas statique et *ddeg* dans le cas dynamique). Aucune information sur l'état passé n'est exploitée.

En 2004, [19] proposent, dans un cadre général, celui des CSPs n-aire, une heuristique de choix de variables originale, *wdeg* (pour weighted degree), dirigée par les conflits. Cette heuristique dynamique est basée sur un poids *weight* associé à chaque contrainte du problème. Pour une contrainte C_j, le poids *weight* est incrémenté de un à chaque conflit impliquant cette dernière. L'évaluation du degré pondéré $\alpha_{wdeg}(V_i)$ d'une variable V_i est alors égale à la somme des poids des contraintes liant V_i et au moins une seconde variable non instantiée ; plus formellement :

– $\alpha_{wdeg}(V_i) = \sum_{V_i \in vars(C_j) \wedge |FutVars(C_j)|>1} weight[C_j]$

, où $|\ FutVars(C_j)\ |$ représente le nombre de variables non instanciées dans $vars(C_j)$. Dans la suite du mémoire, le degré pondéré α_{wdeg} est noté *wdeg*. Dans le même article, les auteurs proposent l'heuristique *dom/wdeg* qui consiste à sélectionner prioritairement la variable avec le plus petit rapport taille courante du domaine de la variable sur le degré pondéré de cette dernière. Cette heuristique, *dom/wdeg* compense l'absence d'un schéma rétrospectif (analyse des conflits) par un apprentissage lui permettant d'orienter la recherche vers les parties difficiles ou inconsistantes du problème.

Les auteurs ont proposé cette heuristique dans le cadre de méthodes prospectives tel que le *Forward-Checking* [83] (FC) et le *Maintien de l'Arc Cohérence* [63, 144] (MAC). Ces méthodes mettent en œuvre des mécanismes de filtrage permettant de modifier le problème selon l'assignation réalisée, ceci grâce à des révisions successives des variables de manière à éliminer les valeurs devenues inconsistantes. Cette révision correspond par exemple à l'utilisation d'un filtrage gros grain (orienté arc dans notre cas). Soit Q l'ensemble des arcs à reviser, un arc étant défini comme un couple *(Contrainte, Variable)*. La révision de l'arc (C_j, V_i) consiste à éliminer les valeurs de $dom(V_i)$ devenues inconsistantes avec C_j.

La fonction *filter*, qui prend en paramètre l'ensemble Q, permet de reviser un certain nombre d'arcs itérativement (l'initialisation de Q est spécifique à FC et à MAC). Lorsqu'un *dead-end* ($dom(V_i) = \emptyset$) est rencontrée lors d'une

Algorithme 7: filtrer(Q : ensemble d'arcs) :booleen

```
1: tantque Q ≠ ∅ faire
2:   Sélectionner et éliminer (C_j, V_i) de Q
3:   si reviser(C_j, V_i) alors
4:     si dom(V_i) = ∅ alors
5:       weight[C_j]++ // incrémentation du compteur de la contrainte
6:       retourner ECHEC
7:     sinon
8:       mettre à jour Q // en fonction de l'algorithme de filtrage
9:     finsi
10:  finsi
11: fin tantque
12: retourner SUCCES
```

révision effective de l'arc (C_j, V_i), le poids *weight* associé à la contrainte C_j est incrémenté de un. Le filtrage se termine alors sur un échec. Dans le cas contraire, l'ensemble Q est mis à jour. La révision est effectuée par la fonction *reviser*.

Algorithme 8: reviser(C_j : contrainte, V_i : Variable) :booleen

```
nbElements ← |dom(V_i)|
pour chaque v ∈ dom(V_i) faire
  si rechercherSupport(C_j, V_i, v) = false alors
    retirer v de dom(V_i)
  finsi
fin pour
retourner nbElements ≠ |dom(V_i)|
```

Les auteurs ont montré l'efficacité d'une telle heuristique face aux heuristiques classiques sur un certain nombre de problèmes académiques structurés et non structurés. Cependant, cette heuristique n'a été présentée que pour des méthodes prospectives, dont le schéma d'exploration est de type retour arrière chronologique (backtrack). On peut donc s'interroger sur l'apport de la transposition d'une telle heuristique pour des méthodes rétrospectives, voir hybrides, tel que la méthode hybride *FC-CBJ* alliant un schéma prospectif et un schéma rétrospectif, mis en œuvre respectivement par le *Forward-Checking* et le *Conflict Directed Backjumping*.

3.4 Heuristique dirigée par les conflits pour une méthode hybride : FC-CBJ

La méthode hybride *FC-CBJ* exploite la capacité de filtrage à moindre coût de la méthode prospective *Forward-Checking* et la capacité de retour arrière de la méthode rétrospective *Conflict Directed Backjumping* qui permet d'identifier la variable source de l'échec. Dans [19], l'heuristique *dom/wdeg* ne dispose pas de cette information (la variable source de l'échec) car elle est appliquée à un schéma de recherche de type retour arrière chronologique (backtrack). Appliquer l'heuristique *dom/wdeg* à la méthode *FC-CBJ* sans prendre en compte la capacité de cette dernière à identifier les variables sources de l'échec risque d'orienter la recherche vers des parties non nécessairement difficiles ou inconsistantes du problème.

Nous proposons d'enrichir la phase d'apprentissage de l'heuristique *dom/wdeg* en redéfinissant le poids *weight* afin d'exploiter la capacité de la méthode rétrospective *CBJ* à identifier la source de l'échec. Il en résulte une

nouvelle heuristique dirigée par les conflits, $dom/wdeg_{cbj}$. Pour des raisons pratiques, l'étude de l'heuristique $dom/wdeg_{cbj}$ est limitée au cadre des CSPs binaires.

Tout d'abord, il est important de remarquer que dans un cadre binaire, il y a équivalence entre associer un poids *weight* à chaque contrainte ou à chaque variable. Pour cela il suffit de remplacer la ligne 5 de l'algorithme 7 par les deux lignes suivantes :

- $weight[V_i][secondVar(C_j, V_i)] + +$
- $weight[secondVar(C_j, V_i)][V_i] + +$

C_j est la contrainte à l'origine d'une impasse lors de la révision de l'arc (V_i, C_j), et $secondVar(C_j, V_i)$ une fonction retournant la seconde variable liée par la contrainte C_j, la première étant la variable V_i (voir l'algorithme 9). Incrémenter le poids *weight* associé à une contrainte C_j d'arité égale à deux revient donc à incrémenter les poids *weight* associés aux deux variables liées par C_j. Le degré pondéré d'une variable V_i s'exprime alors de la façon suivante :

- $\alpha_{wdeg}(V_i) = \sum_{V_k \in V, V_k \in FutVars(V)} weight[V_i][V_k]$

Où $\mid FutVars(V) \mid$ représente l'ensemble des variables non instanciées appartenant à V.

Algorithme 9: filtrer(Q : ensemble d'arcs) :booleen

```
1: tantque Q ≠ ∅ faire
2:    Sélectionner et éliminer (Cj,Vi) de Q
3:    si reviser(Cj,Vi) alors
4:       si dom(Vi) = ∅ alors
5:          weight[Vi][secondVar(Cj, Vi)] + +
6:          weight[secondVar(Cj, Vi)][Vi] + +
7:          retourner ECHEC
8:       sinon
9:          mettre à jour Q
10:      finsi
11:   finsi
12: fin tantque
13: retourner SUCCES
```

Soit $weight_{cbj}$ le poids issue de la redéfinition du mécanisme d'incrémentation du poids *weight*, qui a pour but d'enrichir la phase d'apprentissage de l'heuristique *dom/wdeg* appliquée au *FC-CBJ*. Durant une recherche arborescente de type *FC-CBJ*, lors de l'échec de l'instanciation de la variable courante V_{curr} ($dom(V_{curr}) = \emptyset$ à cause des révisions successives de *FC*), la méthode rétrospective *CBJ* permet d'identifier la variable V_{jump} dont l'affectation courante est responsable de l'échec. La variable V_{jump} doit alors changer de valeur afin de débloquer la situation. L'idée est alors de conserver l'information d'échec entre les deux variables en incrémentant les poids $weight_{cbj}$ associés aux variables V_{curr} et V_{jump}. Le poids $weight_{cbj}$ nous permet ensuite d'évaluer le degré pondéré $\alpha_{wdeg_{cbj}}(V_i)$ de la variable V_i et ainsi de définir la nouvelle heuristique dynamique $dom/wdeg_{cbj}$, qui consiste à sélectionner prioritairement la variable avec le plus petit rapport : taille courante du domaine de la variable, *dom*, sur le degré pondéré $wdeg_{cbj}$ de cette dernière.

Dans le but de mettre en évidence l'intérêt de la redéfinition du mécanisme d'incrémentation du poids *weight* lors de l'adaptation de l'heuristique *dom/wdeg* à la méthode *FC-CBJ* dans un cadre binaire, les poids *weight* et $weight_{cbj}$ ont été associés aux variables. Cependant, il est possible d'adopter la définition de [19]. Pour

cela, les explications d'échec doivent être maintenues sous forme d'ensemble de contraintes menant au conflit, ce qui permet de maintenir la notion de poids associé à chaque contrainte. En cas de conflit, il suffit alors d'incrémenter les poids des contraintes présentes dans le conflit. Ainsi, le degré pondéré d'une variable aurait la même définition que dans [19] et l'heuristique $dom/wdeg_{cbj}$ devient alors applicable dans un cadre n-aires.

3.5 Mécanisme d'apprentissage

Lors de la validation, la succession de CSPs résolus, provenant de l'énumération exhaustive de la combinatoire, peut être assimilée à la résolution d'un CSP dynamique. Soit $P_{(i)}$ un CSP binaire associé à une combinaison i. Ainsi, valider le design d'une matrice de routage consiste à résoudre la séquence $P_{(0)}, P_{(1)}, \cdots, P_{(N)}$ de N CSPs, où chacun est le résultat d'un certain nombre de changements appliqués au précédent, ce qui correspond à la definition d'un Problème Dynamique de Satisfaction de Contraintes (DCSP). Ces changements sont des ajouts et/ou suppressions de valeurs, et activations et désactivations de variables (et/ou de contraintes), qui correspondent respectivement à l'énumération des sous-combinaisons de NPT pannes de TWTAs et à l'énumération des sous-combinaisons de NCA canaux actifs. Pour chaque sous combinaison de canaux actifs, T sous-combinaisons de NPT pannes de TWTAs sont générées.

La particularité de cette sequence de CSPs tient dans le fait que tous les T CPSs, une variable est introduite en lieu et place d'une autre variable. Ceci revient à activer une variable et à en désactiver une autre. Entre les CSPs successifs ayant les mêmes variables actives, des changements de valeurs sont effectués. Ainsi, deux CSPs séparés de $T-1$ CSPs ne diffèrent que d'une variable et de quelques valeurs. Il en résulte une forte similitude entre les CSPs successifs.

La notion de CSP Dynamique (DCSP) a été introduite par [52] comme une suite de CSPs statiques $P_{(0)}, P_{(1)}, \cdots, P_{(N)}$, chacun résultant du précédent par des opérations sur les valeurs (ajout et/ou suppression). Dans [114], un DCSP est une généralisation d'un CSP [157], qui est défini par un ensemble de variables, un état d'activation des variables, un domaine pour chaque variable, et par deux types de contraintes : contraintes de compatibilité qui régissent l'association variable-valeur, et contraintes d'activation qui permettent d'activer ou de désactiver une variable selon l'assignation des autres variables. Une contrainte d'activation est de la forme : Si la valeur v est assignée à la variable V_i, alors $V_j, V_g, \cdot$ deviennent actives. Le terme *conditionnelle* serait peut être plus approprié.

Dans le cadre de la résolution de Problème Dynamique de Satisfaction de Contraintes[52, 114], de nombreux travaux ont été menés sur l'exploitation de l'effort de résolution des CSPs statiques successifs, notamment sur la réutilisation d'une solution précédemment calculée [10, 160], l'exploitation des explications de la non faisabilité [146] et l'adaptation des algorithmes d'arc consistance [12, 42]. Dans la même optique, nous proposons un mécanisme d'apprentissage par la mémorisation des conflits. En fonction de la similitude entre les CSPs successifs, une fois la recherche arborescente menée à terme avec une preuve de non-satisfiabilité ou une preuve de satisfiabilité pour une instance CSP $P_{(i)}$, les poids $weight_{cbj}$ permettent de mettre en évidence les parties difficiles ou inconsistantes de $P_{(i)}$, et potentiellement de $P_{(i+1)}$. Ainsi, durant la résolution de $P_{(i+1)}$, qui résulte d'un certain nombre de changements dans $P_{(i)}$, il serait intéressant de traiter en priorité les parties difficiles ou inconsistantes identifiées durant la résolution de $P_{(i)}$. Ceci permettrait, selon le principe *fail-first*, soit de détecter la non-satisfiabilité de l'instance plus tôt, soit d'accélérer la résolution en traitant en priorité les parties difficiles du problème (simplification des sous-problèmes inhérents). Dans la suite de cette section, ce qui est appliqué au poids $weight_{cbj}$ peut l'être également au poids $weight$.

Le mécanisme d'apprentissage est proposé dans le cadre d'un Problème Dynamique de Satisfaction de Contraintes (DCSP) de T CSPs. Le mécanisme d'apprentissage consiste à initialiser le poids $weight_{cbj}$ de chaque variable du CSP $P_{(i+1)}$ par celui du CSP $P(i)$ ayant les mêmes variables. Quand deux CSPs successifs different d'une

variable (et de quelques valeurs), les poids $weight_{cbj}$ sont réinitialisés à 1, ce qui permet de préserver le caractère discriminant des poids $weight_{cbj}$. Finalement, les poids $weight_{cbj}$ sont réinitialisés tout les T CSPs, d'où un DCSP de T CSPs. Il est important de noter que ce mécanisme est applicable à un DCSP quelconque. Dans la suite du mémoire, l'association de la méthode *FC-CBJ*, de l'heuristique $dom/wdeg_{cbj}$ et du mécanisme d'apprentissage est notée $dom/wdeg^{T}_{cbj}$.

3.6 Pre-traitement

Au début des années 90, la combinaison *Forward-Checking* et *Conflict-directed BackJumping* (*FC-CBJ*) était considérée comme l'approche la plus efficace pour résoudre les instances CSPs. Quelques années plus tard, il a été montré que l'algorithme prospectif MAC[144], qui maintient la cohérence d'arc durant la recherche était plus efficace que *FC-CBJ* pour des instances difficiles de grande taille [13]. Néanmoins, [8] ont montré que l'algorithme du *Forward-Checking* (donc de *FC-CBJ*) était plus performant sur les instances CSPs ayant un graphe de contrainte très dense et une faible dureté des contraintes, ce qui est le cas de la grande majorité des CSPs que nous traitant dans cette problématique.

Des expérimentation menées avec une implementation de l'algorithme de Maintien D'arc Cohérence, basé sur un algorithme d'Arc Cohérence *AC3.2bit* [100], ont montrées l'efficacité de *FC-CBJ* sur *MAC*. Néanmoins, l'utilisation de l'Arc Cohérence *AC3.2bit* en pré-traitement s'est révélée être très efficace. Arc Cohérence *AC3.2bit* est basé sur une représentation bit-vectoriel des domaines des variables et des contraintes. Cette représentation est ensuite exploitée efficacement, par exemple lors de la recherche de support.

La représentation bit-vectoriel des domaines est effectué grâce à un tableau à deux dimension, noté *bitDom*, qui permet d'associer à chaque variable la représentation binaire $bitDom[V]$ du domaine $dom(V)$ de la variable V. Ainsi :
- lors de la restauration de i^{eme} valeur, la seule opération requise est la suivante :
  $bitDom[V][i \quad div \quad 64] \leftarrow bitDom[V][i \quad div \quad 64] \quad \textbf{OR} \quad masks1[i \quad mod \quad 64]$
- Lors de la suppression de la i^{eme} valeur du $dom(V)$, l'unique opération est la suivante :
  $bitDom[V][i \quad div \quad 64] \leftarrow bitDom[V][i \quad div \quad 64] \quad \textbf{AND} \quad masks0[i \quad mod \quad 64]$

div représente la division entière, *mod* le reste de la division, OR et AND les opérateurs logiques. La structure $masks1$ (respectivement $masks0$) est un mot de 64 bits où tous les bits sont à 0 (respectivement à 1) sauf le i^{eme} bit qui est égale à 1 (respectivement à 0). Le lecteur est invité à consulter [100] pour plus de details.

3.7 Implementation

L'implémentation d'un algorithme et des mécanismes participant à la résolution est un facteur déterminant dans l'efficacité d'une méthode de résolution. Dans le cadre de la problématique de validation, un certain nombre d'améliorations d'implémentation ont été apportées afin d'accroître l'efficacité de l'existant, donc de réduire significativement le temps de validation.

La première amélioration a consistée à rendre dynamique le fonctionnement de l'algorithme du *Forward-Checking*. Lors de la résolution, la propagation est bien effectuée après chaque instanciation, mais de manière statique. Lors de la propagation de l'instanciation courante au nœud i, la réduction du sous-problème inhérent n'est pas persistante au nœud $i+1$, c'est-à-dire que la propagation doit être menée pour toutes les variables précédemment instanciées et la variable courante nouvellement instanciée, cela à chaque nœud. L'amélioration a consistée à rendre persistant cette propagation et à mettre en place un mécanisme de restauration des domaines.

L'algorithme 10 présente plus en détaille le fonctionnement du **Forward-Checking** avec les fonctions de sauvegarde et de restauration du contexte.

Algorithme 10: forward-checking($\mathcal{V}$)

$\mathcal{V}$: ensemble des variable à instancier i :instanciation courante
si $\mathcal{V} = \emptyset$ **alors**
  i est une solution
sinon
  Choisir $V \in \mathcal{V}$
  pour tout $v \in D_V$ **faire**
    sauvegarde($\mathcal{V} \backslash \{V\}$)
    si check-forward($V, v, \mathcal{V}$) **alors**
      forward-checking($\mathcal{V} \backslash \{V\}, i \cup \{V \rightarrow v\}$)
    sinon
      restauration($\mathcal{V} \backslash \{V\}$)
    finsi
  fin pour
finsi

Algorithme 11: check-forward(V,v,$\mathcal{V}$)

pour tout $V^{'} \in \mathcal{V}$ **faire**
  pour tout $v^{'} \in D_V$ **faire**
    si $\{V \rightarrow v, V^{'} \rightarrow v^{'}\}$ non consistant **alors**
      $D_{V'} \leftarrow D_{V'} \backslash \{v^{'}\}$
    finsi
  fin pour
  si $D_{V'} = \emptyset$ **alors**
    return faux
  finsi
fin pour
return vrai

Dans le but de faciliter la détection des dead-ends, une structure de donnée, sous la forme d'un tableau à une dimension a été ajoutée afin de comptabiliser le nombre de valeurs restantes pour chaque variable, ceci à chaque nœud. Cela permet de détecter plus rapidement les dead-ends en déterminant la taille d'un domaine sans avoir à le parcourir. Associé à cette structure, des mécanismes de sauvegarde et de restauration ont été mis en place.

Une dernière amélioration a consistée à compacter les domaines toute au long de la résolution, ceci à chaque suppression de valeur. La figure 3.1(a) illustre un tableau à deux dimensions, la première étant les variables, la seconde les valeurs appartenant au domaines de chaque variable. Ce tableau permet de connaître la disponibilité d'une valeur pour une variable donnée : $(a)[i][j] = 1$ signifie que la valeur associée à l'index j est disponible pour la variable i, dans le cas contraire $(a)[i][j] = 0$. Compacter un domaine revient à créer le tableau à deux dimensions illustré par la figure 3.1(b), où sont référencées seulement les valeurs disponibles grâce à leurs index dans le tableau précédent. Ansi, lors de la recherche on a directement accès aux valeurs disponibles. Les domaines sont compactés après chaque suppression. Le tableau 3.1(c) permet de connaître la taille d'un domaine à tous moment de la résolution. Ce tableau est également mis à jour tout au long de la recherche.

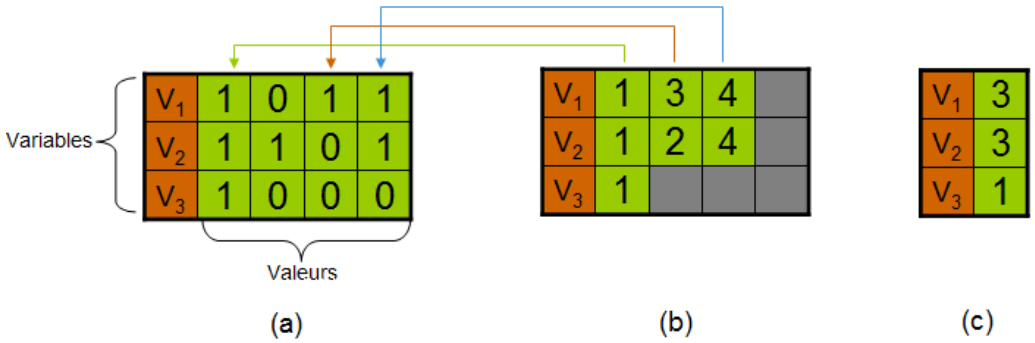


FIGURE 3.1 – Domaines compactés

3.8 Experimentations

Les expérimentations ont été menées sur les cas de validation décrits par le tableau 3.1 de la section 3.2. Ces cas de validation sont présentés sous la forme de 9 CSP Dynamiques : A_1, B_1, B_2, B_3, B_4, C_1, C_2, C_3 et C_4. Dans un premier temps nous allons présenter les résultats expérimentaux concernant les heuristiques dynamiques de choix de variables et le mécanisme d'apprentissage basé sur la mémorisation des poids $weight_{cbj}$. Par la suite, nous présenterons les résultats de l'utilisation de l'algorithme de maintien d'arc cohérence *MAC*. Enfin nous confronterons le nouvel algorithme de validation avec toutes les améliorations, à l'algorithme existant. Dans la suite du mémoire, un cas de validation est appelé DCSP pour *Dynamic Constraint Satisfaction Problem.*

Les tableaux 3.2 et 3.3 présentent des statistiques en nombre d'assignations (#asgs) et de temps de validation (cpu) permettant de départager dom, $dom/wdeg_{cbj}$ et $dom/wdeg$. Ces expérimentations ont été menées avec la nouvelle base algorithmique. On remarque que l'heuristique dom est surclassée par l'heuristique $dom/wdeg_{cbj}$ et $dom/wdeg$. La réduction en terme de nombre nœuds explorés et de temps de résolution est significative. On remarque également que pour l'ensemble des DCSPs, regroupant 86 126 235 CSPs dont 26 594 967 non-satisfiables, l'heuristique $dom/wdeg_{cbj}$, que nous proposons, donne de meilleurs résultats en terme de nombre d'assignations et de temps cpu que l'heuristique $dom/wdeg$. La redéfinition des poids $weight$ a permis d'exploiter plus efficacement la capacité de la méthode rétrospective *CBJ* à identifier la source du conflit. Pour un coût d'implémentation limité, le nombre d'assignations et le temps cpu ont été respectivement réduits de 1,43% à 21,51% et de 0,66% à 17,61%. Ainsi, au lieu d'appliquer $dom/wdeg$ à la méthode hybride *FC-CBJ*, il est plus intéressant d'appliquer son extension, l'heuristique $dom/wdeg_{cbj}$.

Le tableau 3.4 illustre des statistiques en nombre d'assignations (#asgs) et de temps de validation (cpu) permettant d'évaluer l'apport du mécanisme d'apprentissage basé sur la mémorisation des poids $weight_{cbj}$. On remarque une réduction significative de ces deux paramètres pour les DCSPs A_1, B_1, C_1, C_2 et C_4. Par exemple, le temps de validation du DCSP C_4 est égale à environ 13 heures avec le mécanisme d'apprentissage alors qu'il est d'environ 56 heures sans le mécanisme d'apprentissage, soit une réduction de 75.83% en temps cpu. Il est également important de remarquer que la validation des DCSPs B_3, B_4 et C_3 voient leurs temps cpu augmenter (au maximum de 9%) avec le mécanisme d'apprentissage. Néanmoins, ce temps reste inférieur à celui obtenu avec l'heuristique $dom/wdeg$ sans mémorisation. Ainsi, avec un coût d'implémentation très faible, le mécanisme d'apprentissage offre des améliorations significatives sur notre exemple d'application.

Afin d'évaluer l'apport de la combinaison de toutes les améliorations nous avons confronté la base algorithmique de référence (sans améliorations) et la base algorithmique intégrant toutes les améliorations. Le tableau 3.5 illustre cette comparison en se basant sur le temps cpu de validation. Ainsi, le temps cpu a été réduit significativement, par un facteur allons de 1.5 pour A_1 à 105 pour C_4. Notez que pour certain cas de validation, nous avons eu recoure à une approximation du temps cpu du fait de l'extrême lenteur de la validation.

DCSPs		*dom*	*dom/wdeg$_{cbj}$*	*dom/wdeg$_{cbj}$* vs *dom*
A_1	#asgs	665 846 192	548 839 044	**-17,57 %**
	cpu	00 :59 :45	00 :57 :51	**-3.18%**
B_1	#asgs	2 587 519 313	1 628 697 906	**-37.06%**
	cpu	3 :50 :13	03 :30 :57	**-8.37%**
B_2	#asgs	16 916 500 680	7 644 525 648	**-54.81%**
	cpu	18 :32 :30	11 :02 :34	**-40.44%**
B_3	#asgs	36 263 059 221	17 250 317 028	**-52.43%**
	cpu	43 :42 :46	28 :41 :01	**-40.32%**
B_4	#asgs	38 707 994 914	18 860 453 983	**-51.28%**
	cpu	49 :18 :37	31 :24 :56	**-36.29%**
C_1	#asgs	1 189 943 039	696 215 827	**-41.49%**
	cpu	2 :20 :50	01 :58 :34	**-15.81%**
C_2	#asgs	2 682 262 079	2 116 805 967	**-21.08%**
	cpu	3 :34 :01	03 :20 :54	**-6.13%**
C_3	#asgs	5 866 979 097	4 776 536 475	**-18.59%**
	cpu	10 :03 :52	08 :54 :25	**-11.50%**
C_4	#asgs	159 242 151 769	21 146 498 726	**-86.72 %**
	cpu	281 :40 :54	56 :20 :16	**-80.00%**

TABLE 3.2 – *dom* vs *dom/wdeg$_{cbj}$*

Les tableaux 3.6 et 3.7 illustrent les résultats d'expérimentations de l'algorithme de Maintien d'Arc Cohérence basé sur l'Arc Cohérence AC3Bit [100]. Ces résultats permettent de valider le choix de *FC-CBJ* en lieu place de *MAC*. La première série d'expérimentations du tableau 3.6 confronte *MAC-AC3bit* et *FC-CBJ*, tous deux adoptant l'heuristique de choix de variable *dom*. *MAC-AC3bit* permet une réduction significative en terme de nombre nœuds, mais le maintien de la cohérence à chaque nœuds est très coûteux en temps cpu ce qui ne lui permet pas de prendre l'ascendant sur l'algorithme *FC-CBJ*, sauf pour l'instance de test $B3$ où MAC-AC3bit a un léger avantage. Ces résultats sont conforme aux conclusions de [8] : l'algorithme du *Forward-Checking* (donc de *FC-CBJ*) est plus performant sur les instances CSPs ayant un graphe de contrainte très dense et une faible dureté des contraintes, ce qui est le cas de la grande majorité des CSPs que nous traitant dans cette problématique.

La seconde série d'expérimentations du tableau 3.7 confronte *MAC-AC3bit* et *FC-CBJ*, tous deux adoptant l'heuristique de choix de variable *dom/wdeg$_{cbj}$* et le mécanisme de mémorisation périodique du *wdeg$_{cbj}$*. Comme précédemment, *MAC-AC3bit* permet une réduction significative du nombre de nœuds. Cependant, le temps cpu est fortement augmenté. *FC-CBJ* conserve un nette avantage. Il est néanmoins intéressant de constater l'apport significatif de l'heuristique *wdeg$_{cbj}$* et du mécanisme de mémorisation à l'algorithme *MAC-AC3bit*. Cet apport est visible lorsque l'on confronte les résultats du tableau 3.6 pour la méthode *MAC-AC3bit* associée à l'heuristique *dom* et les résultats du tableau 3.7 pour la même méthode mais associée cette fois-ci à l'heuristique *dom/wdeg$_{cbj}$* et au mécanisme de mémorisation périodique du *wdeg$_{cbj}$*. Par exemple, pour l'instance $C4$ nous obtenons une réduction du nombre de nœuds de 91% et du temps cpu de 91%, passant de 12 jours de validation à 1 journée.

DCSPs		$dom/wdeg$	$dom/wdeg_{cbj}$	$dom/wdeg_{cbj}$ vs $dom/wdeg$
A_1	#asgs	556 791 037	548 839 044	**-1,43%**
	cpu	0 day 00 :58 :14	0 day 00 :57 :51	**-0,66%**
B_1	#asgs	1 854 403 849	1 628 697 906	**-12,17%**
	cpu	0 day 03 :45 :21	0 day 03 :30 :57	**-6,39%**
B_2	#asgs	8 444 489 777	7 644 525 648	**-9,47%**
	cpu	0 day 12 :03 :07	0 day 11 :02 :34	**-8,37%**
B_3	#asgs	19 335 093 642	17 250 317 028	**-10,78%**
	cpu	1 day 04 :41 :01	1 day 02 :05 :21	**-9,05%**
B_4	#asgs	21 059 047 647	18 860 453 983	**-10,44%**
	cpu	1 day 10 :11 :17	1 day 07 :24 :56	**-8,11%**
C_1	#asgs	745 606 878	696 215 827	**-6,62%**
	cpu	0 day 02 :02 :07	0 day 01 :58 :34	**-2,91%**
C_2	#asgs	2 696 933 390	2 116 805 967	**-21,51%**
	cpu	0 day 04 :03 :51	0 day 03 :20 :54	**-17,61%**
C_3	#asgs	5 136 698 506	4 776 536 475	**-7,01%**
	cpu	0 day 09 :26 :42	0 day 08 :54 :25	**-5,70%**
C_4	#asgs	22 188 614 328	21 146 498 726	**-4,70%**
	cpu	2 days 08 :36 :03	2 days 08 :20 :16	**-0,46%**

TABLE 3.3 – $dom/wdeg_{cbj}$ vs $dom/wdeg$

DCSPs		$dom/wdeg_{cbj}$	$dom/wdeg^T_{cbj}$	$dom/wdeg^T_{cbj}$ vs $dom/wdeg_{cbj}$
A_1	#asgs	548 839 044	320 962 468	**-41,52%**
	cpu	0 day 00 :57 :51	0 day 00 :45 :59	**-20,51%**
B_1	#asgs	1 628 697 906	1 365 322 208	**-16,17%**
	cpu	0 day 03 :30 :57	0 day 03 :12 :19	**-8,83%**
B_2	#asgs	7 644 525 648	7 398 902 997	**-3,21%**
	cpu	0 day 11 :02 :34	0 day 10 :22 :58	**-5,98%**
B_3	#asgs	17 250 317 028	18 829 599 249	9,16%
	cpu	1 days 02 :05 :21	1 days 04 :01 :24	7,41%
B_4	#asgs	18 860 453 983	20 386 953 037	8,09%
	cpu	0 day 31 :24 :56	0 day 33 :19 :55	6,10%
C_1	#asgs	696 215 827	227 762 909	**-67,29%**
	cpu	0 day 01 :58 :34	0 day 01 :12 :06	**-39,19%**
C_2	#asgs	2 116 805 967	1 739 005 574	**-17,85%**
	cpu	0 day 03 :20 :54	0 day 02 :45 :34	**-17,59%**
C_3	#asgs	4 776 536 475	4 802 963 000	0,55%
	cpu	0 day 08 :54 :25	0 day 08 :58 :31	0,77%
C_4	#asgs	21 146 498 726	5 802 664 153	**-72,56%**
	cpu	2 days 08 :20 :16	0 day 13 :36 :51	**-75,83%**

TABLE 3.4 – Mécanisme d'apprentissage

DCSPs		Existant	avec toutes les amélioration	Ecart
A_1	cpu	0 day 02 :08 :02	0 day 00 :45 :59	**-64.84%**
B_1	cpu	0 day 05 :03 :45	0 day 03 :12 :19	**-36.63%**
B_3	cpu	13 days 00 :45 :03	1 days 04 :01 :24	**-91.04%**
B_4	cpu	$\approx$ 37 days 01 :07 :03	1 days 09 :19 :55	**-96.25%**
C_1	cpu	0 day 05 :30 :12	0 day 01 :12 :06	**-78.18%**
C_2	cpu	$\approx$ 4 days 20 :00 :00	0 day 02 :45 :34	**-97.63%**
C_3	cpu	$\approx$ 15 days 20 :00 :00	0 day 08 :58 :31	**-97.64%**
C_4	cpu	$>$ 60 days 00 :00 :00	0 day 013 :36 :51	**-99.06%**

TABLE 3.5 – Existant et améliorations

DCSPs		FC-CBJ	MAC AC3Bits	FC-CBJ vs MAC AC3Bits
A1	#asgs	665 846 192	328 998 045	-50,59%
	cpu	00 :57 :02	03 :12 :57	238,31%
B1	#asgs	2 587 519 313	995 377 484	-61,53%
	cpu	03 :48 :29	09 :40 :54	154,24%
B2	#asgs	16 916 500 680	2 737 882 013	-83,82%
	cpu	18 :32 :30	1 day 03 :25 :35	47,92%
B3	#asgs	36 263 059 221	3 681 695 476	**-89,85%**
	cpu	1 days 19 :42 :46	1 days 15 :03 :14	**-10,66%**
B4	#asgs	38 707 994 914	4 212 926 315	-89,12%
	cpu	2 days 00 :18 :37	2 days 03 :31 :11	7,01%
C1	#asgs	1 189 943 039	568 912 633	-52,19%
	cpu	02 :20 :50	05 :43 :32	143,93%
C2	#asgs	2 682 262 079	512 305 900	-80,90%
	cpu	03 :34 :01	07 :19 :13	105,23%
C3	#asgs	5 866 979 097	2 945 826 585	-49,79%
	cpu	10 :03 :52	15 :46 :07	56,68%
C4	#asgs	159 242 151 769	89 242 274 565	-43,96%
	cpu	11 days 17 :40 :54	12 days 11 :06 :45.	10,10%

TABLE 3.6 – FC-CBJ vs MAC-AC3bits : *dom*

DCSPs		MAC AC3Bits	FC-CBJ	FC-CBJ vs MAC AC3Bits
A1	#asgs	278 837 492	320 962 468	15,11%
	cpu	03 :00 :15	00 :45 :59	**-74,49%**
B1	#asgs	507 849 357	1 365 322 208	168,84%
	cpu	06 :57 :47	03 :11 :26	**-54,18%**
B2	#asgs	1 005 896 392	7 398 902 997	635,55%
	cpu	15 :45 :47	10 :22 :58	**-34,13%**
B3	#asgs	1 556 118 242	18 829 599 249	1110,04%
	cpu	1 days 01 :11 :26	1 days 04 :01 :24	11,25%
B4	#asgs	1 995 962 531	20 386 953 037	921,41%
	cpu	1 days 13 :41 :55	1 days 06 :19 :55	**-19,54%**
C1	#asgs	199 395 957	227 762 909	14,23%
	cpu	04 :13 :54	01 :12 :06	**-71,60%**
C2	#asgs	512 305 900	1 739 005 574	239,45%
	cpu	07 :19 :13	02 :45 :34	**-62,30%**
C3	#asgs	807 050 150	4 802 963 000	495,13%
	cpu	10 :46 :07	08 :58 :31	**-16,65%**
C4	#asgs	1 092 269 858	5 802 664 153	431,25%
	cpu	1 days 1 :45 :57	13 :36 :51	**-47,16%**

TABLE 3.7 – FC-CBJ vs MAC-AC3bits : $dom/wdeg^{T}_{cbj}$

CHAPITRE 4

TENTATIVES INFRUCTUEUSES

Ce court chapitre présente les tentatives infructueuses d'amélioration et de réduction de la combinatoire.

4.1 Réduction de la combinatoire

4.1.1 No-Good

Un no-good est une assignation partielle qui ne peut être étendue à aucune solution réalisable. Transposé à la problématique de validation, du point de vue de la combinatoire, un no-good peut être défini comme étant une combinaison partielle dont aucune extension à une combinaison complète n'est valide. Ce type de no-good, s'il existe permettrait de réduire la combinatoire en concluant, pour un no-good donné, que toutes ces extensions sont non-valides. Ces extensions ne seront donc pas examinées ce qui a pour effet de réduire le temps de validation. Lors de la validation d'une matrice de switches, un certain nombre de combinaisons sont dites non valides. L'idée est de trouver pour chaque combinaison non valide un sous ensemble maximal de canaux actifs pour lesquels il existe une solution réalisable partielle. L'existence de l'un de ces sous ensembles signifie que quelque soit le sous ensemble complémentaire ajouté, pour un ensemble de tube en pannes définit il n'existe pas de solution.

La problématique de detection des no-goods a été modélisée sous la forme d'un **Programme Linéaire en Nombre Entier (PLNE)**. Soit la variable binaire V_{ij} indiquant si le chemin j est sélectionné par le canal i.

- Soit NCA l'ensemble des canaux actifs.
- Soit P_i l'ensemble des chemins associés au canal i.
- Soit C_j l'ensemble des chemins incompatibles avec le chemin j.

Le model Linéaire peut être exprimé de la façon suivante :

$$\text{maximize} \sum_{i \in NCA} \sum_{j \in P_i} V_{ij} \tag{4.1}$$

subject to

$$\sum_{j \in P_i} V_{ij} \leq 1, \forall i \in NCA \tag{4.2}$$

$$V_{ij} + V_{i'j'} \leq 1, \forall i \in NCA, \forall j \in P_i, \forall i' \in NCA, \forall j' \in P_{i'} \tag{4.3}$$

$$V_{ij} \in 0, 1, \forall i \in NCA, \forall j \in P_i \tag{4.4}$$

Les résultats obtenus sont loin d'être satisfaisant, en termes de temps de résolution mais surtout pour la qualité des No-Goods obtenus. Par exemple, pour le cas de validation B, avec 21 canaux actifs parmi 24 et 4 tubes en pannes parmi 25, les No-Goods obtenus sont des sous ensembles de 19 à 20 canaux, les sous ensembles de 20 canaux étant majoritaires. Ainsi, pour un ensemble de 21 canaux actifs, 24 à 276 combinaisons peuvent être identifiées comme étant non valides, ceci sans résolution. Cependant, le temps de résolution d'une combinaison est de l'ordre de 10^{-3}s contre 2s pour le programme linéaire. Le temps de calcul des No-Good est alors trop coûteux en comparaison du gain obtenu en termes de réduction de la combinatoire.

4.1.2 Symétrie

La notion de symétrie est présente entre la matrice de switches en section d'entrée et en section de sortie. Elle peut être également présente entre deux sous-matrices d'une matrice de switches en section de sortie. Détecter une telle symétrie permettrait de réduire la combinatoire. Soit deux sous matrice A et B, chacune composée de 20 canaux et 22 tube amplificateurs. Soit une validation à 18 canaux actifs et 4 pannes de tubes amplificateurs. Lors de l'énumération de la combinatoire, un certain nombre de combinaisons auront la totalité des canaux actifs dans la matrice A ou dans la matrice B. Dans ce cas la, il suffit de valider les combinaisons de l'une des deux matrices. Néanmoins, si lors de la validation des combinaisons de la matrice A (par exemple), certaines combinaisons se trouvaient être non valides, dans ce cas, il est nécessaire de mettre en place des hypothèses sur les connexions entre les deux matrices. Selon le cas, ces hypothèses peuvent se ramener à remplacer un ensemble de connexions, soit par des canaux virtuels, soit par des tubes virtuels.

Dans cette thèse, nous avons implémenté un algorithme de détection des symétries, basé sur un algorithme de recuit simulé. Les symétries proposées étaient cohérentes mais l'automatisation de leur exploitation était plus delicate. Par manque de temps nous avons privilégié d'autres pistes d'amélioration.

4.1.3 Arbre de recherche complet

Le processus de validation actuel énumère tous les CSPs pour les résoudres successivement. Du fait du schéma d'énumération, deux CSPs successifs ne diffèrent que de quelques valeurs, ou d'une variable et de quelques valeurs. Pour un sous ensemble de canaux actifs fixé, la différence en terme de valeurs provient de l'énumération de la combinatoire des tubes amplificateurs.

L'idée de l'arbre de recherche complet consiste à résoudre un CSP défini par l'ensemble des canaux actifs et sans pannes de tubes. Lors de la résolution, la combinatoire de pannes de tubes amplificateurs est générée toute au long de la résolution. Pour ce faire, la résolution consiste à trouver toutes les solutions. Ces solutions nous permettent de déterminer les combinaisons valides. Une fois la résolution terminée, les combinaisons non

valides sont déduites des combinaisons valides. Une solution nous permet de caractériser une combinaison car les canaux actifs sont définis par les variables et les chemins associés aux variables nous permettent de définir les pannes de tubes par déductions.

Afin de réduire l'espace de recherche, du fait de l'exploration de toutes les solutions, nous avons mis en place un mécanisme de filtrage nous permettant de ne pas redécouvrir les mêmes combinaisons. Ce mécanisme s'est révélé très coûteux en temps de calcul, ce qui a eu pour effet l'abandon de cette piste malgré une réduction significative du nombre de nœud.

4.2 Apprentissages

4.2.1 Réutilisation de solutions

Du fait de la forte similitude entre les CSPs successifs traités dans cette problématique, il nous est apparu pertinent de réutiliser une solution réalisable d'un CSP i lors de la résolution du CSP $i+1$. Le mécanisme mis en place était très simple : il suffisait de placer en debut de chaque domaine, la valeur assignée à la variable courante lors de la précédente résolution (si une solution a été trouvée). Ainsi, les valeurs utilisées par la solution précédente sont examinées en premier, si cette dernière existe.

La réutilisation d'une solution est effectuée soit entre deux CSPs successif ayant le même ensemble de variables et quelques valeurs de différence, ou entre deux CSPs séparés de $T-1$ combinaisons, ayant une variable et quelques valeurs de différences.

Séduisante sur le papier, cette idée s'est révélée inéfficace dans la pratique. La réutilisation d'une solution précédemment calculée a eu pour impact d'augmenter le nombre de nœuds et donc de détériorer les performances de résolution, que ce soit pour les CSPs succéssifs ou les CSPs séparés de $T-1$ combinaisons.

4.2.2 Last Conflict reasoning

Dans le cadre de l'apprentissage lors de la résolution, nous avons essayé d'appliquer une idée simple proposée par Lecoutre [29]. [29] propose une méthode d'apprentissage nommée Last Conflict reasoning (LC). Cette méthode se base sur un raisonnement sur les conflits pour guider le développement de l'arborescence de recherche vers les variables à l'origine de l'inconsistance. À chaque étape de la résolution, l'heuristique de choix de variables sélectionne la dernière variable ayant conduit à un *deadend*, si une telle variable existe.

Dans la pratique, cette idée s'est révélée peu convaincante en terme de réduction du nombre de nœuds et donc en terme de performance de résolution.

CHAPITRE 5

CONCLUSION

5.1 contributions

Le travail présenté dans cette thèse s'inscrit dans un contexte industriel pointu et fortement concurrentiel, ce qui implique la nécessité d'être à la pointe de la technologie et de la technique. Le processus de validation du design d'une matrice de switches est une étape primordiale lors d'un appel d'offre. La capacité de validation en un temps réduit est indispensable à la bonne conduite des appels d'offre dont dépend l'avenir de la société. Dans cette thèse nous avons proposé des améliorations algorithmiques permettant de répondre à l'accroissement de la complexité des satellites de télécommunication en réduisant le temps de validation du design des matrices de switches.

Le processus de validation adopté consiste en une énumération exhaustive de millions, voire de plusieurs milliards de CSPs binaires dont il faut prouver la satisfiabilité ou la non satisfiabilité. La résolution d'un CSP fait appel à une méthode complète sous la forme d'une recherche arborescente de type depth-first, avec à la fois un schéma prospectif « look-ahead », et un schéma rétrospectif « look-back », respectivement le *forward-checking* et le *conflict-directed back jumping*.

Dans ce travail de thèse nous proposons des améliorations significatives de l'algorithme de validation. Nous proposons une nouvelle heuristique de choix de variable, $dom/wdeg_{cbj}$ inspirée de l'heuristique $dom/wdeg$ [19]. $dom/wdeg$ a été proposée dans le cadre de méthodes prospectives tel que le *Forward-Checking* et le *Maintien de l'Arc Cohérence*. $dom/wdeg$ et $dom/wdeg_{cbj}$ exploitent les échecs rencontrés au long de la résolution grâce à la pondération de poids associés aux contraintes (aux variables dans un cadre binaire). Contrairement à $dom/wdeg$, l'heuristique $dom/wdeg_{cbj}$ que nous proposons exploite la capacité de la méthode rétrospective *CBJ* à identifier la source de l'échec, ceci grâce à la redéfinition du poids *weight*. Ainsi, $dom/wdeg_{cbj}$ est plus efficace que $dom/wdeg$. $dom/wdeg_{cbj}$ constitue une contribution dans cette thèse.

Basé sur cette nouvelles heuristique, nous proposons également un mécanisme d'apprentissage basé sur la mémorisation des pondérations entre les CSPs successifs, permettant ainsi une réduction significative du temps de validation. Les pondérations sont remise à zéro tous les T CSPs. Ce mécanisme d'apprentissage constitue la seconde contribution.

La structure des CSPs traités ne permettant pas l'utilisation d'une méthode prospective du type MAC, nous avons proposé d'utiliser l'algorithme d'Arc Cohérence *AC3-bits* en prétraitement. Enfin, nous avons effectué des améliorations d'implementation, notamment sur la gestion des domaines des variables.

Toutes ces améliorations ont permis une réduction significative du temps de validation. Cette réduction a été observée sur des cas de tests qui représentent des cas réels de validation de design de matrice de switches.

Ces améliorations ont permit la validation en 6 jours d'une matrice de switches comprenant 50 milliard de combinaisons, avec 100 % des combinaisons valide. Cette validation a été effectuée en parallélisant le calcul sur 4 machines. Sans ces améliorations, cette validation n'aurait été possible qu'au bout de plusieurs mois dans les mêmes conditions. Un second exemple significatif concerne la validation d'une matrice comprenant cette fois-ci 7 millions de combinaisons mais dont la validation sans les améliorations aurait pris plusieurs mois. Cette validation a été effectuée dans le cadre d'une réponse à un appel d'offre. La validation a été possible en quelques heures seulement (13 heures). Une approximation du temps de validation nécessaire sans les améliorations nous a donné un facteur de réduction de ce même temps de 1420. Ces exemples de validations constituent une contribution industrielle importante.

Finalement, le temps de résolution d'un CSPs, lors d'une validation oscille entre 10^{-3s} et $10^{-6}s$.

5.2 perspectives

La problématique de validation est une problématique assez simple à appréhender. Les difficultés principales sont d'une part une forte combinatoire et d'autre part une certaine hétérogénéité des CSPs traités, cependant avec une forte proportion de CSPs de faible voire de moyenne difficulté (voir le tableau 3.1).

Cette hétérogénéité peut être exploitée en effectuant une première passe de validation, sans résolution, afin d'analyser les CSPs du point de vue de la densité du graphe de contrainte et de la dureté des contraintes, ceci afin d'identifier les CSPs difficiles et de les résoudre grâce à une méthode de maintien d'arc cohérence par exemple. Les autres CSPs quant à eux seront résolut avec l'algorithme *FC-CBJ*.

Les améliorations futures devront nécessairement passer par une réduction de la combinatoire, en exploitant la symétrie, soit d'un point de vue topologique ou du point de vue de la structure même des CPSs. Durant cette thèse, un travail a été effectué sur la détection de la symétrie topologique, ceci grâce à un algorithme de type recuit simulé. Il permettait d'identifier deux sous-matrices symétriques. Cette voie n'a pas été approfondie par manque de temps.

Nous considérons qu'il est inutile d'investir plus de temps dans l'amélioration du *FC-CBJ* du fait des performances actuelles (10^{-3s} à $10^{-6}s$ pour résoudre un CSP).

Troisième partie

Conclusion Générale

Conclusion

Le travail présenté dans cette thèse s'inscrit dans un contexte industriel pointu et fortement concurrentiel, ce qui implique la nécessité d'être à la pointe de la technologie et de la technique. Que ce soit pour la problématique de routage ou la problématique de validation, nous avons proposé des solutions innovantes pour répondre aux besoins. Ainsi, nous avons proposé une méthode de multi-routage réaliste et efficace permettant aux équipes d'accommodation de gagner en efficacité. Nous avons également proposé des améliorations significatives de l'algorithme de validation qui se sont traduits par une réduction importante des temps de validation, ainsi une validation de plusieurs mois se réduit à plusieurs heures.

Les principales contributions de cette thèse réside dans la proposition d'une méthode de multi-routage efficace basée sur des mécanismes novateurs, notamment l'algorithme de recherche de plus court chemin grand voisinage basé sur les masques de direction, et le méthode de réparation globale avec tous les mécanismes qu'elle met en oeuvre pour proposer un routage réaliste.

Elle réside également dans la proposition d'une nouvelle heuristique de choix de variable, $dom/wdeg_{cbj}$ inspirée de l'heuristique $dom/wdeg$ [19], et d'un mécanisme d'apprentissage basé sur cette nouvelle heuristique. Contrairement à $dom/wdeg$, l'heuristique $dom/wdeg_{cbj}$ que nous proposons exploite la capacité de la méthode rétrospective CBJ à identifier la source de l'échec, ceci grâce à la redéfinition du poids *weight*. Ainsi, $dom/wdeg_{cbj}$ est plus efficace que $dom/wdeg$.

Le mécanisme d'apprentissage que nous proposons est basé sur la mémorisation des pondérations, des poids *weight* entre les CSPs successifs, permettant ainsi une réduction significative du temps de validation. Les pondérations sont remise à zéro tous les T CSPs.

Le lecteur pourra se référer aux conclusions propres à chaque problématique, le chapitre 6 de la première seconde pour la problématique de multi-routage et le chapitre 5 de la seconde partie pour la problématique de validation.

BIBLIOGRAPHIE

[1] E. Aarts and J. Lenstra. *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., Chichester, UK, 1997.

[2] L.C. Abel. On the ordering of connections for automatic wire routing. *IEEE Transactions on Computers*, 21 :1227–1233, 1972.

[3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows : Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[4] Christoph Albrecht. Provably good global routing by a new approximation algorithm for multicommodity flow. In *Proceedings of the ACM International Symposium on Physical Design*, pages 19–25, 2000.

[5] Krzysztof Apt. *Principles of Constraint Programming*. Cambridge University Press, New York, NY, USA, 2003.

[6] A. Asmara and U. Nienhuis. Automatic piping system in ship. In *Proceedings of the 5th International Conference on Computer Applications and Information Technology in the Maritime Industries*, Leiden, 2006.

[7] A. Asmara and U. Nienhuis. Automatic piping system implementation : A real case. In *Proceedings of the 6th International Conference on Computer Applications and Information Technology in the Maritime Industries*, 2007.

[8] C. Assef and S. Lakhdar. De fc a mac : un algorithme paramétrable pour la résolution de csp. In *Premières Journées Francophones de Programmation par Contraintes (JFPC'05)*, pages 267–276, Lens, France, 2005.

[9] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3) :345–405, 1991.

[10] J. Maloney B. N. Freeman-Benson and A. Borning. An incremental constraint solver. *Communications of the ACM*, 33(1) :54–63, 1990.

[11] P. Bertier and B. Roy. Procédure de résolution pour une classe de problèmes pouvant avoir un caractère combinatoire. *Cahiers du Centre d'Études de Recherche Opérationnelle (CERO)*, 6 :202–208, 1964.

[12] C. Bessière. Arc-Consistency in Dynamic Constraint Satisfaction Problems. In *Proceeding of The Ninth National Conference on Artificial Intelligence*, pages 221–226, 1991.

[13] C. Bessiere and R. Regin. Mac and combined heuristics : two reasons to forsake fc (and cbj ?) on hard problems. In *Proceedings of CP'96*, pages 61–75, 1996.

[14] C. Bessière. Arc-consistency and arc-consistency again. *Artificial Intelligence*, 65(1) :179–190, 1994.

[15] C. Bessière. Constraint propagation. Technical Report LIRMM 06020, CNRS/University of Montpellier, March 2006.

[16] C. Bessière and J.-C. Regin. Arc consistency for general constraint networks : preliminary results. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 398–404, Nagoya, Japan, 1997.

[17] C. Bessière and J.-C. Regin. Enforcing arc consistency on global constraints by solving subproblems on the fly. In *Proceedings of the 5th International Conference on Principles and Practice of Constraint Programming (CP'99)*, pages 103–117, London,UK, 1999. Springer-Verlag.

[18] C. Bessière and J.-C. Régin. Refining the basic constraint propagation algorithm. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 309–315, 2001.

[19] F. Boussemart and al. Boosting Systematic Search by Weighting Constraints. In *Proceedings of the ECAI'O4*, pages 146–150, 2004.

[20] J. Bresina. Heuristic-biased stochastic sampling. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI'96)*, pages 271–278, Portland, OR, USA, 1996.

[21] M. Burstein and R. Pelavin. Hierarchical wire routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD.2(4) :223–234, 1983.

[22] C. Hane C. Barnhart and P. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48(2) :318–326, 2000.

[23] R. Anbil C. Barnhart, E. L. Johnson and L. Hatay. A column-generation technique for the long-haul crew-assignment problem. *Optimization in industry 2 : Mathematical programming and modeling techniques in practice*, pages 7–24, 1994.

[24] A. Chmeiss ans L. Sais C. Bessiere. Neighbourhood-based variable ordering heuristics for the Constraint Satisfaction Problem. In *Proceedings of CP'01*, pages 565–569, 2001.

[25] E.C. Freuder C. Bessière and J.-C. Régin. Using constraint metaknowledge to reduce arc consistency computation. *Artificial Intelligence*, 65(1) :125–148, 1999.

[26] C. K. Wong C. Chiang and M. Sarrafzadeh. A weighted steiner tree-based global router with simultaneous lenght and density minimization. *IEEE Transactions on Computer-Aided Design*, 13(12) :1461–1469, 1994.

[27] S. Lin C. K. Cheng, J. Lillis and N. Chang. *Interconnect Analysis and Synthesis*. Wiley Interscience, New York, NY, USA, 2000.

[28] F. Boussemart C. Lecoutre and F. Hemery. De ac3 à ac7. In *Proceedings of JFPLC'03*, 2003.

[29] S. Tabary C. Lecoutre, L. Sais and V. Vidal. Last conflict based reasoning. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI'06)*, pages 133–137, Trento, Italy, 2006.

[30] Z. Cao, T. Jing, J. Xiong, Y. Hu, L. He, and X. Hong. Dprouter : A fast and accurate dynamic-pattern-based global routing algorithm. In *ASP-DAC '07 : Proceedings of the 2007 Asia and South Pacific Design Automation Conference*, pages 256–261, Washington, DC, USA, 2007. IEEE Computer Society.

[31] Y. J. Chang, Y. T Lee, and T. Ch. Wang. Nthu-route 2.0 : a fast and stable global router. In *ICCAD '08 : Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 338–343, Piscataway, NJ, USA, 2008. IEEE Press.

[32] M. Cho, K. Lu, K. Yuan, and D. Z. Pan. Boxrouter 2.0 : Architecture and implementation of a hybrid and robust global router. In *ICCAD '07 : Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design*, pages 503–508, Piscataway, NJ, USA, 2007. IEEE Press.

[33] M. Cho and D. Z. Pan. Boxrouter : a new global router based on box expansion and progressive ilp. In *DAC '06 : Proceedings of the 43rd annual Design Automation Conference*, pages 373–378, New York, NY, USA, 2006. ACM.

[34] C. Chu. Flute - fast lookup table based wirelenght estimation techniques. In *Proceeding of ICCAD 2004*, pages 44–47, 2004.

[35] C. Chu and W. Yiu-Chung. Flute - fast lookup table based rectilinear steiner minimal tree algorithm for vlsi design. *IEEE Transactions on Computer-Aided Design*, 17(1) :70–83, 2008.

[36] M. Clerc. Discrete particle swarm optimization illustrated by the traveling salesman problem. 2000.

[37] J. Cong, L. He, C. k. Koh, and P. H. Madden. Performance optimization of vlsi interconnect layout. *Integration, the VLSI Journal*, 21 :1–94, 1996.

[38] M. C. Cooper. An optimal k-consistency algorithm. *Artificial Intelligence*, 41 :89–95, 1989.

[39] B. Selman C.P. Gomes and al. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning*, 24 :67–100, 2000.

[40] G. B. DANTZIG and P. WOLFE. Decomposition principle for linear programs. *Operations Research*, 8 :101–111, 1960.

[41] D.Brelaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22 :251–256, 1979.

[42] R. Debruyne. Les algorithmes d'arc-consistence dans les csp dynamiques. *Revue d'intelligence artificielle*, 9(3) :239–267, 1995.

[43] R. Dechter. Enhancement schemes for constraint processing : backjumping,learning, and cutset decomposition. *Artificial Intelligence*, 41(3) :273–312, 1990.

[44] R. Dechter. Enhancement schemes for constraint processing : Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41 :273–312, 1990.

[45] R. Dechter and D. Frost. Backjumping-based backtracking for constraint satisfaction problem. *Artificial Intelligence*, 136 :147–188, 2002.

[46] R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality of A*. *Journal of Association for Computing Machinery*, 32(3) :505–536, 1985.

[47] C. Van der Velden, C. Bil, X. Yu, and A. Smith. Towards a knowledge based cable router for aerospace vehicles. In *Information and Knowledge Engineering Conference*, LasVegas, US, 2006.

[48] C. Van der Velden, C. Bil, X. Yu, and A. Smith. An intelligent system for automatic layout routing in aerospace design. *Innovations in Systems and Software Engineering*, 3(2) :117–128, 2007.

[49] C. Van der Velden, C. Bil, X. Yu, and A. Smith. An intelligent system for rule based automated routing. In *the 5th Australasian Congress on Applied Machanics, ACAM*, 2007.

[50] C. Van der Velden, C. Bil, X. Yu, and A. Smith. An intelligent system for routing automation. In *Innovative Production Machines and Systems, Virtual Conference*, pages 2–13, 2008.

[51] R. Detcher. *Constraint Processing*. Morgan Keufmann, 2003.

[52] R. Detcher and A. Detcher. Belief maintenance in dynamic constraint networks. In *Proceedings of the 7th National Conference on Artifical intelligence*, pages 37–42, St. Paul, MN, USA, 1988.

[53] R. Detcher and I. Meiri. Experimental evaluation of preprocessing techniques in Constraint Satisfaction Problems. In *Proceedings of IJCAI'89*, pages 271–277, 1989.

[54] M. DIB. *Tabu-NG : hybridation de programmation par contraintes et recherche locale pour la résolution de CSP*. PhD thesis, L'Université de Technologie de Belfort-Montbeliard, 2010.

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

[56] R. C. Eberhart and J. A. Kennedy. New optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, pages 39–43, Nagoya, Japan, 1995.

[57] P. Van Beek F. Rossi and T. Walsh. *Handbook of constraint programming*, volume 35. 2006.

[58] L. R. Ford. and D. R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, 50 :97–101, 1958.

[59] E.C. Freuder. A sufficient condition for backtrack-free search. *Journal of the ACM*, 29(1) :24–32, 1982.

[60] D. Frost and R. Detcher. Look-ahead value ordering for Constraint Satisfaction Problems. In *Proceedings of IJCAI'95*, pages 572–578, 1995.

[61] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, USA, 1978.

[62] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *In Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 300–309, 1997.

[63] J. Gaschnig. *Performance measurement and analysis of certain search algorithms*. PhD thesis, 1979.

[64] J. Gashnig. A generak Backtracking algorithm that eliminates most redundant tests. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, page 457, 1977.

[65] J. Gashnig. Experimental case studies of backtrack vs. Waltz-type vs. new algorithms for satisfacing assignment problems. In *Proceedings of the Canadian Artificial Intelligence Conference*, pages 268–277, 1978.

[66] I. P. Gent and I.P. Underwood. The logic of search algorithms : Theory and applications. In *Proceedings of the Third International Conference on Principles and Practice of Constraint Programming (CP'97)*, pages 77–91, 1997.

[67] P.C. GILMORE and R.E. GOMORY. A linear programming approach to the cutting-stock problem. *Operations Research*, 9 :849–859, 1961.

[68] M. L. Ginsberg. Dynamic Backtracking. *Journal of Artificial Intelligence Research*, 1 :25–46, 1993.

[69] M.L. Ginsberg. Dynamic backtracking. *Journal of Artificial Intelligence Research (JAIR)*, 1 :25–46, 1993.

[70] F. Glover. Tabu search - part i. *ORSA Journal on Computing*, 1(3) :190–206, 1989.

[71] F. Glover. Tabu search - part ii. *ORSA Journal on Computing*, 2 :4–32, 1990.

[72] D ; E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[73] S. Golomg and L. Baumert. Backtrack programming. *Journal of Association for Computing Machinery*, 12 :516–524, 1965.

[74] D. Grimes and R. J.Wallace. Learning from failures in constraint satisfaction search. In *AAAIWorkshop on Learning for Search*, Boston, Massachusetts, USA, 2006.

[75] M. Grötschel, A. Martin, and R. Weismantel. The steiner tree packing problem in vlsi design. *Math. Program.*, 78(2) :265–281, 1997.

[76] R. Guirardello and R. E. Swaney. Optimization of process plant layout with pipe routing. *Computers and Chemical Engineering*, 30(1) :99–114, 2005.

[77] M. Lederhose H. Schmidt-Traub, T. Holtkötter and P. Leuders. An approach to plant layout optimization. *Chemical Engineering Technology*, 22(2) :105–109, 1999.

[78] T. Holtkötter H. Schmidt-Traub, M. Köster and N. Nipper. Conceptual plant layout. *Computers and Chemical Engineering Supplement*, 22((suppl issue)) :499–504, 1998.

[79] C. H. Hsu H. Y. Chen and Y. W. Chang. High-performance global routing with fast overflow reduction. In *ASP-DAC '09 : Proceedings of the 2009 Asia and South Pacific Design Automation Conference*, pages 582–587, Piscataway, NJ, USA, 2009. IEEE Press.

[80] G. T. Hamachi and J. K. Ousterhout. A switchbox router with obstacle avoidance. In *DAC '84 : Proceedings of the 21st Design Automation Conference*, pages 173–179, Piscataway, NJ, USA, 1984. IEEE Press.

[81] M. Hanan. On steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics*, 14(2) :255–265, 1966.

[82] P. Hansen and B. Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44(4) :279–303, 1990.

[83] R. M. Haralick and G. L. Elliott. Increasing tree search efficiency for Constraint Satisfaction Problems. *Artificial Intelligence*, 14 :263–313, 1980.

[84] N. Hasan and C. L. Liu. A force-directed maze router. In *Advanced Research in VLSI*. The Press MIT, 1987.

[85] K. L. Hoffman and M. Padberg. Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39(6) :657–682, 1993.

[86] J. Holland. Outline for a logical theory of adaptive systems. *Journal of the Association of Computing Machinery*, 9(3) :297–314, 1962.

[87] J. Hu and S. S. Sapatnekar. A survey on multi-net global routing for integrated circuits. *Integration, the VLSI Journal*, 31 :1–49, 2001.

[88] Y. K. Hwang and N. Ahuja. Gross motion planning—a survey. *ACM Comput. Surv.*, 24(3) :219–291, 1992.

[89] T. Ito. Piping layout wizard : Basic concepts and its potential for pipe route planning. In *Proceedings of the 11th international conference on Industrial and engineering applications of artificial intelligence and expert systems*, pages 438–447, London, UK, 1998. Springer-Verlag.

[90] T. Ito. A genetic algorithm approach to piping route path planning. *Journal of Intelligent Manufacturing*, 10 :103–114, 1999.

[91] T. Ito. Route planning wizard : Basic concept and its implementation. In *Proceedings of the 15th international conference on Industrial and engineering applications of artificial intelligence and expert systems*, pages 547–556, London, UK, 2002. Springer-Verlag.

[92] P. Siarry J. Dréo, A. Pétrowski and E. Taillard. *Métaheuristiques pour l'optimisation difficile*. Eyrolles, 2003.

[93] P. Galinnier J .K. Hao and M. Habib. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Artificial Intelligence*, 13(2) :283–324, 1999.

[94] R. Bayardo Jr. and D. Miranker. A complexity analysis of space-bounded learning algorithms for the constraint satisfaction problem. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 298–304, Portland, Oregon, USA, 1996.

[95] A. B. Kahng and G. Robins. *On Optimal Interconnections for VLSI*. Kluwer Academic Publishers, Boston, MA, USA, 1995.

[96] J. A. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, NJ, 1995.

[97] J. A. Kennedy and R. C. Eberhart. A discrete binary version of the particle swarm algorithm. In *Proceedings of the World Multiconference on Systemics,Cybernetics and Informatics 1997*, pages 4104–4109, Piscataway, NJ, 1997.

[98] B. Korte, H. J. Prömel, and A. Steger. Steiner trees in VLSI-layout. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, editors, *Paths, Flows, and VLSI-Layout*, pages 185–214. Springer-Verlag, Berlin, Germany, 1990.

[99] D. R. Fulkerson L. R. Ford. Flows in networks. Technical report, Princeton University Press, 1962.

[100] C. Lecoutre and J. Vion. Enforcing arc consistency using bitwise operations. *Constraint Programming Letters(CPL)*, 2 :21–35, 2008.

[101] C. Y. Lee. An algorithm for path connections and its applications. *IRE Transactions on Electronic Computers*, EC-10(2) :364–365, 1961.

[102] T. Lengauer and M. Lügering. Provably good global routing of integrated circuits. *SIAM J. on Optimization*, 11(1) :1–30, 2000.

[103] Th. Lengauer. *Combinatorial algorithms for integrated circuit layout*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[104] Th. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[105] G. Lin and al. An efficient branch-and-bound algorithm for the assignment of protein backbone nmr peaks. In *Proceedings of the IEEE Computer Society Bioinformatics Conference (CSB'02)*, –, 2002.

[106] D.G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 2nd edition, june 1984.

[107] R. Abdallah M. Dib and A. Caminada. Arc-consistency in constraint satisfaction problems : A survey. In *Proceedings of 2nd International Conference on Computational Intelligence, Modelling and Simulation*, pages 28–30, Bali, Indonesia, September 2010.

[108] A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1) :118–126, 1977.

[109] A.K. Mackworth and Freuder. The complexity of some polynomial network consitency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25(1) :65–74, 1985.

[110] A. Martin and R. Weismantel. Packing paths and steiner trees : Routing of electronic circuits. *CWI Quarterly*, 6 :185–204, 1993.

[111] L. McMurchie and C. Ebeling. Pathfinder : a negotiation-based performance-driven router for fpgas. In *FPGA '95 : Proceedings of the 1995 ACM third international symposium on Field-programmable gate arrays*, pages 111–117, New York, NY, USA, 1995. ACM.

[112] G. Meixner and U. Lauther. A new global router based on a flow model and linear assignment. In *Proceeding of IEEE international Conference on Computer-Aided Design*, pages 44–47, 1990.

[113] K. Mikami and K. Tabuchi. A computer program for optimal routing of printed circuit conductors. In *Proceedings of the International Federation of Information Processing Congress*, volume 2, pages 1475–1478, 1968.

[114] S. Mittal and B. Falkenhainer. Dynamic constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artifical intelligence*, pages 25–32, 1990.

[115] I. Méndez-Díaz and P. Zabala. A branch-and-cut algorithm for graph coloring. *Discrete Appl. Math.*, 154(5) :826–847, 2006.

[116] M. D. Moffitt. Maizerouter : Engineering an effective global router. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 27(11) :2017–2026, 2008.

[117] M. D. Moffitt, J. A. Roy, and I. L. Markov. The coming of age of (academic) global routing. In *ISPD '08 : Proceedings of the 2008 international symposium on Physical design*, pages 148–155, New York, NY, USA, 2008. ACM.

[118] R. Mohr. A correct path consistency algorithm and an optimal generalized arc consistency algorithm. Technical Report 87-R-030, CRIN, 1987.

[119] R. Mohr and T. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, 28 :225–233, 1986.

[120] R. Mohr and G. Masini. Good old discrete relaxation. In *Proceedings of the European Conference on Artificial Intelligence (ECAI'88)*, pages 651–656, 1988.

[121] E. F. Moore. The shortest path through a maze. *in Ann. Comput. Lab. of Harvard University (Harvard University Press, Cambridge, MA)*, 30 :285–292, 1959.

[122] R. Debruyne N. Jussien and P. Boizumault. Maintaining arc-consistency within dynamic backtracking. *Principles and Practice of Constraint Programming*, pages 249–261, 2000.

[123] R. Nair. A simple yet effective technique for global wiring. *IEEE Transactions on Computer-Aided Design*, CAD-6(2) :165–172, 1987.

[124] M. M. Ozdal and M. D. F. Wong. Archer : a history-driven global routing algorithm. In *ICCAD '07 : Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design*, pages 488–495, Piscataway, NJ, USA, 2007. IEEE Press.

[125] N. J. Nilsson P. E. Hart and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2) :100–107, 1968.

[126] N. J. Nilsson P. E. Hart and B. Raphael. Correction to "a formal basis for the heuristic determination of minimum cost paths". *SIGART Newsletter*, (37) :28–29, 1972.

[127] M. Pan and C. Chu. Fastroute 2.0 : A high-quality and efficient global router. In *ASP-DAC '07 : Proceedings of the 2007 Asia and South Pacific Design Automation Conference*, pages 250–255, Washington, DC, USA, 2007. IEEE Computer Society.

[128] M. Pan and Ch. Chu. Fastroute : a step to integrate global routing into placement. In *ICCAD '06 : Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, pages 464–471, New York, NY, USA, 2006. ACM.

[129] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1994.

[130] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization : algorithms and complexity*. Prentice Hall, Englewood Cliffs, NJ, USA, 1982.

[131] J. H. Park and R. L. Storch. Pipe-routing algorithm development : Case study of a ship engine room design. *Expert Systems with Applications*, 23(3) :299–309, 2002.

[132] P. Prosser. Hybrid algorithms for the Constraint Satisfaction Problem. *Computational Intelligence*, 9(3) :268–299, August 1993.

[133] P. Prosser. Mac-cbj : maintaining arc consistency with conflict-directed backjumping. Research Report 95/177, Dept. of Computer Science, University of Strathclyde, 1995.

[134] J. Li R. C. Carden and C. K. Cheng. A global router with a theoretical bound on the optimal solution. *IEEE Transactions on Computer-Aided Design*, 15(2) :208–216, 1996.

[135] E. Bozorgzadeh R. Kastner and M. Sarrafzadeh. Predictable routing. In *Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, pages 110–114, Piscataway, NJ, USA, 2000. IEEE Press.

[136] C.P. Gomes R. Williams and B. Selman. Backdoors to typical case complexity. In *Proceedings of IJCAI'03*, 2003.

[137] P. Raghavan and C. D. Thompson. Multiterminal global routing : A deterministic approximation scheme. *Algorithmica*, 6 :73–82, 1991.

[138] J.-C. Régin. A filtering algorithm for constraints of difference in csps. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pages 362–367, 1994.

[139] H. Richert and G. Gruhn. A numeric heuristic system for plant wide pipe routing. *Computers and Chemical Engineering Supplement*, 23 :735–738, 1999.

[140] R. L. Rivest and Ch. M. Fiduccia. A "greedy" channel router. In *DAC '82 : Proceedings of the 19th Design Automation Conference*, pages 418–424, Piscataway, NJ, USA, 1982. IEEE Press.

[141] J. A. Roy and I. L. Markov. High-performance routing at the nanometer scale. In *ICCAD '07 : Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design*, pages 496–502, Piscataway, NJ, USA, 2007. IEEE Press.

[142] C. D. Gelatt S. Kirkpatrick and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598) :202–208, 1983.

[143] M. SEHYUN S. S. KANG and S. H. HAN. A design expert system for auto-routing of ship pipes. *Journal of Ship Production*, 15(1) :1–9, 1999.

[144] D. Sabin and E. Freuder. Contradicting conventional wisdom in constraint satisfaction. In *Proceedings of the PPCPA'94*, Seattle WA, 1994.

[145] T. Schiex and G. Verfaillie. Nogood recording for static and dynamic constraint satisfaction problem. *International Journal of Artificial Intelligence Tools*, 3(2) :187–207, 1994.

[146] T. Schiex and G. Verfaillie. Nogood Recording for Static and Dynamic Constraint Satisfaction Problems. *IJAIT*, 3(2) :187–207, 1994.

[147] F. Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *J. ACM*, 37(2) :318–334, 1990.

[148] E. Shragowitz and S. Keel. A global router based on a multicommodity flow model. *Integr. VLSI J.*, 5(1) :3–16, 1987.

[149] B.M. Smith. The brelaz heuristic and optimal static orderings. In *Proceedings of CP'99*, pages 405–418, Alexandria,VA, 1999.

[150] B.M Smith and S.A. Grant. Trying harder to fail first. In *Proceedings of ACAI'98*, pages 249–253, 1998.

[151] J. Soukup. Fast maze router. In *DAC '78 : Proceedings of the 15th Design Automation Conference*, pages 100–102, Piscataway, NJ, USA, 1978. IEEE Press.

[152] F. Soumis. Decomposition and column generation. Technical report, Les Cahiers du GERAD, 1997.

[153] R.M. Stallman and G. J. Sussman. Forward reasoning and dependency directed backtracking in a system for computer aided circuit analysis. *Artificial Intelligence*, 9 :135–196, 1977.

[154] Yann T. *Tissu Numérique Cellulaire à Routage et Configuration Dynamiques*. PhD thesis, École Polythechnique Fédérale de Lausanne, Lausanne, 2005.

[155] B. S. Ting and B. N. Tien. Routing techniques for gate array. *IEEE Transactions on Computer-Aided Design*, CAD-2(4) :301–312, 1983.

[156] E. Tsang. *Foundations of constraint satisfaction*. Academic Press, London and San Diego, 1993.

[157] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press Limited, Sandiego, 1993.

[158] M.R.C van Dongen. $ac3_d$ and efficient arc-consistency algorithm with a low space complexity. In *Proceedings of CP'02*, pages 755–760, 2002.

[159] V. V. Vazirani. *Approximation Algorithms*. Springer, March 2004.

[160] G. Verfaillie and T. Schiex. Solution reuse in dynamic constraint satisfaction problems. In *Proceeding of the Twelfth National Conference on Artificial Intelligence*, pages 307–312, 1994.

[161] M. Tamminen W. K. Luk, P. Sipala and al. A hierarchical global wiring algorithm for custom chip design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 6(4) :518–533, 1987.

[162] M. Wallace. Practical applications of constraint programming. *CONSTRAINTS*, 1 :139–168, 1996.

[163] T. Watanabe and Y. Sugiyama. A new routing algorithm and its hardware implementation. In *Proceedings of the 23rd ACM/IEEE Design Automation Conference*, pages 574–580, Piscataway, NJ, USA, 1986. IEEE Press.

[164] Y. Xu, Y. Zhang, and Ch. Chu. Fastroute 4.0 : Global router with efficient via minimization. In *ASP-DAC '09 : Proceedings of the 2009 Asia and South Pacific Design Automation Conference*, pages 576–581, Piscataway, NJ, USA, 2009. IEEE Press.

[165] T. Yoshimura and E. S. Kuh. Efficient algorithms for channel routing. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 1(1) :25–35, 1982.

[166] Y. Zhang, Y. Xu, and Ch. Chu. Fastroute3.0 : a fast and high quality global router based on virtual capacity. In *ICCAD '08 : Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 344–349, Piscataway, NJ, USA, 2008. IEEE Press.

[167] Y. Zhang and R.H.C Yap. Making ac3 an optimal algorithm. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 316–321, 2001.