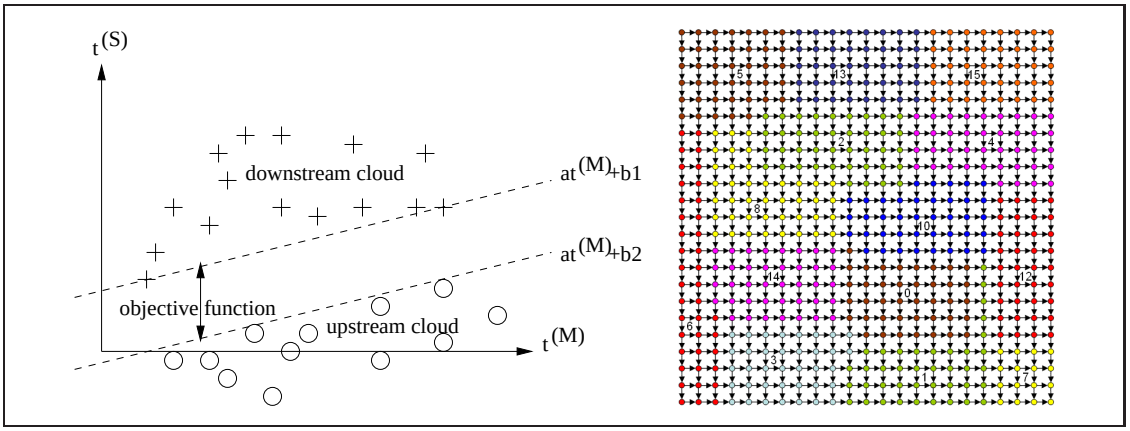
Contributions à l'optimisation combinatoire pour l'embarqué : des autocommutateurs cellulaires aux microprocesseurs massivement parallèles

Renaud Sirdey

Par M. Renaud Sirdey

Contributions à l'optimisation combinatoire pour l'embarqué : des autocommutateurs cellulaires aux microprocesseurs massivement parallèles

Thèse présentée
pour l'obtention du diplôme
d'Habilitation à Diriger des Recherches de l'UTC.



Soutenue le 29 novembre 2011
Spécialité : Technologies de l'Information et des Systèmes

UNIVERSITÉ DE TECHNOLOGIE DE COMPIÈGNE

CONTRIBUTIONS À L'OPTIMISATION COMBINATOIRE POUR L'EMBARQUÉ : DES AUTOCOMMUTATEURS CELLULAIRES AUX MICROPROCESSEURS MASSIVEMENT PARALLÈLES

THÈSE D'HABILITATION À DIRIGER DES RECHERCHES

présentée et soutenue publiquement
par
M. Renaud Sirdey
Ingénieur-chercheur au Commissariat à l'Énergie Atomique
le 29 novembre 2011
à l'Institut de Physique Théorique

devant le jury composé de :

M. Walid Ben-Ameur, Professeur à l'Institut Télécom (rapporteur)
M. Jacques Carlier, Professeur à l'Université de Technologie de Compiègne (président)
M. Philippe Chrétienne, Professeur à l'Université Pierre et Marie Curie (rapporteur)
M. Vincent David, Professeur à l'Institut National des Sciences et Techniques Nucléaires
M. Dritan Nace, Professeur à l'Université de Technologie de Compiègne
M. Marc Sevaux, Professeur à l'Université de Bretagne Sud (rapporteur)

À Delphine, Benoît, Clément et Marion.

Remerciements

Je tiens à exprimer mes plus sincères remerciements à Messieurs Jacques Carlier et Dritan Nace, tous les deux Professeurs à l'Université de Technologie de Compiègne, pour notre collaboration scientifique qui perdure depuis 2003 ainsi que pour avoir coordonné ma thèse d'Habilitation à Diriger des Recherches.

Mes remerciements vont ensuite à Messieurs Walid Ben-Ameur, Professeur à l'Institut Télécom, Philippe Chrétienne, Professeur à l'Université Pierre et Marie Curie, et Marc Sevaux, Professeur à l'Université de Bretagne Sud, qui m'ont fait l'honneur d'accepter de rapporter la présente thèse.

Je remercie également Monsieur Vincent David, Professeur à l'Institut National des Sciences et Techniques Nucléaires, qui dirige le Laboratoire Systèmes Temps Réel Embarqués ainsi que Messieurs Jean-René Lèquepeys et Thierry Collette, chefs successifs du Département Architectures, Conception de Circuits et Logiciels Embarqués du Commissariat à l'Énergie Atomique.

Je tiens à remercier l'ensemble des membres de l'équipe "Calcul intensif embarqué", en particulier Messieurs Pascal Aubry, Loïc Cudennec, Paul Dubrulle, François Galéa, Thierry Goubier et Stéphane Louise. Cela n'a pas tous les jours été facile, mais nous avons bel et bien réussi en moins de trois ans et à partir de rien à faire exister une nouvelle technologie de programmation parallèle, compilateur et support d'exécution inclus, ainsi qu'à amorcer son transfert industriel.

Merci également à mes doctorants : Sergiu Carpov, désormais docteur, qui a travaillé en ordonnancement, Oana Stan (tous les deux co-encadrés avec Jacques et Dritan), qui travaille en optimisation sous incertitude, et Simon Fau (co-encadré avec G. Gogniat, Professeur à l'UBS, et C. Fontaine, cryptologue à l'ENSTB) qui consacre sa thèse aux systèmes de chiffrement homomorphiques.

Je suis aussi redevable envers Simon Bliudze, Sergiu Carpov, Paul Dubrulle, Thierry Goubier et Oana Stan qui ont bien voulu relire des parties de ce mémoire.

Je remercie aussi Madame Sandrine Schlutig, physicienne au synchrotron Soleil, pour une passionnante visite de cet équipement le matin de la soutenance.

Enfin, j'exprime tout mon amour "plus haut que la dernière hauteur du monde" à mon épouse, Delphine, et aux trois merveilleux soleils de notre vie, Benoît, Clément et Marion. Chacun avec leur spécialité : les tours de Hanoï (il y a là un petit effet "papa combinatoricien"), la génération de mots de passe aléatoires (et colorés) et les œuvres d'art (très colorées) sur manuscrit en cours de rédaction...

Table des matières

Introduction

La recherche opérationnelle est née des casse-têtes logistiques imposés aux alliés durant la seconde guerre mondiale. Le terme "opérationnelle" doit donc être compris au sens militaire des opérations et ancre historiquement cette discipline dans le concret des problèmes de terrain. Le chercheur opérationnel est un mathématicien appliqué utilisant une boîte à outils allant de la théorie des graphes à l'optimisation combinatoire en passant par les statistiques et par l'informatique théorique afin de modéliser et de résoudre des problèmes ayant trait à la technologie ou aux affaires humaines. Problèmes au départ souvent posés de manière informelle et qu'il n'a généralement pas le privilège de choisir. Ce faisant, il utilise ou développe des outils mathématiques ad hoc sans a priori autre que celui de résoudre le problème posé de manière opérationnelle, donc adapté au contexte et aux contraintes de l'application. En sus des objectifs de qualités des solutions obtenues, ces dernières relèvent pour l'essentiel du génie logiciel : contraintes sur le temps d'exécution, sur le temps de mise en œuvre, sur la maintenabilité ou encore sur le volume du logiciel résultant, notamment.

Par la suite, depuis la fin du vingtième siècle qui s'est accompagnée d'une généralisation de l'utilisation de calculateurs dans tous les domaines de la technologie, les ingénieurs n'ont de cesse que de complexifier leurs créations et ce à tous les niveaux. Pour ne citer que deux exemples, les autocommutateurs, autrefois constituants élémentaires des infrastructures de télécommunications, deviennent des systèmes répartis mettant en jeu des mécanismes complexes de tolérance aux pannes, et les microprocesseurs, autrefois de simples composants, deviennent parallèles et ressemblent de plus en plus aux supercalculateurs du début des années quatre-vingt-dix [1]. Une conséquence de tout ceci est que l'omniprésence de problèmes difficiles d'optimisation discrète n'est plus l'apanage des grandes infrastructures, comme les grands réseaux de télécommunications ou de distribution d'énergie.

Dès lors qu'il est nécessaire d'utiliser efficacement et de manière maîtrisée un ensemble de ressources inter-dépendantes et en quantités limitées, la pertinence pratique des techniques issues de la recherche opérationnelle s'impose ou se renforce. Et c'est là une tendance forte dans de nombreux domaines technologiques.

1. Mettre en perspective l'architecture du Cray T3D, commercialisé en 1993, avec celles des processeurs (massivement) multicœurs présents sur le marché ou émergents.

Si l'on considère le domaine de la compilation, par exemple, l'utilisation des techniques d'optimisation combinatoire est longtemps restée confinée à l'étape de finalisation de la production de code, le "backend". Mentionnons notamment les problèmes d'affectation des registres et d'ordonnancement d'instructions pour les processeurs VLIW. Or, l'avènement des multicœurs massifs, véritables calculateurs parallèles sur puce, induit une complexification importante du processus de compilation et décuple le besoin en techniques d'optimisation à tel point que l'on peut dire, de manière imagée certes mais sans exagération, qu'une chaîne de compilation pour une telle architecture est une véritable usine à théorie des graphes. Qui plus est, comme nous le verrons dans les pages de ce mémoire, on trouve dans ce contexte un "cocktail explosif" d'instances de grande taille, de données incertaines ainsi que de multiplicité des critères à optimiser.

Notre contribution à l'optimisation combinatoire pour l'embarqué se situe dans ce contexte. Dans le cadre de nos responsabilités opérationnelles, d'architecte système dans l'industrie des télécommunications puis d'encadrement d'équipe de recherche en parallélisme embarqué au CEA, nous avons eu à poser, à formaliser et à résoudre de nombreux problèmes combinatoires initialement non ou mal posés (voire dans certains cas mal résolus), parfois en arrivant à nous raccrocher, par un travail de modélisation mathématique adéquat, à des techniques connues (notamment les flots et la programmation linéaire), parfois en développant nos propres méthodes (mal nécessaire pour l'ingénieur pressé, pain béni pour le chercheur). Nous avons également eu à plusieurs reprises à faire preuve de pédagogie afin de faire adhérer à ces solutions un public d'ingénieurs parfois méfiants envers des techniques mathématiques qu'ils ne connaissent pas, qu'ils ont oubliées ou dont ils ne perçoivent pas la portée applicative systématique.

Ce mémoire d'Habilitation à Diriger des Recherches est constitué de deux parties : l'une consacrée à des applications de l'optimisation combinatoire en télécommunications, l'autre en parallélisme embarqué.

La première partie couvre donc une partie des travaux en optimisation combinatoire que nous avons menés dans l'industrie des télécommunications, entre 2001 et 2007.

Le chapitre 1 présente une sélection de problèmes industriels que nous avons résolus à l'aide des techniques issues de la recherche opérationnelle. Il s'agit notamment de problèmes de configuration de cellules radios (affectation bipartie), de répartition de cellules sur des liens (bin-packing) et de partage équitable de ressources (théorie des systèmes électoraux), de minimisation de consommation électrique (sac à dos non linéaire), d'affectation d'applications de traitement d'appels, ainsi que leurs réplicas passifs, de manière à garantir certaines propriétés de tolérance aux pannes (partitionnement sous contraintes), ainsi que d'allocation dynamique de ressources de vocodage (flots). Ces travaux très appliqués ont donné lieu à autant de méthodes de résolution qui sont ou ont été en opération dans les réseaux de téléphonie mobile de grands opérateurs.

Le chapitre 2 revient brièvement sur nos travaux de thèse de doctorat consa-

crés à l'étude d'un problème d'ordonnancement à contraintes de ressources consistant, étant donnée une répartition arbitraire admissible de processus de traitement d'appels sur les processeurs d'un autocommutateur réparti, à trouver une séquence d'opérations (migrations de processus sans effet sur le service ou arrêts temporaires) de moindre impact par le biais de laquelle une autre répartition arbitraire, et fixée à l'avance, peut être obtenue. La principale contrainte résidant dans le fait que la capacité des processeurs du système ne doit pas être dépassée durant la reconfiguration.

Enfin, le chapitre 3 est consacré à l'étude des problèmes de synchronisation des horloges posés par l'introduction de technologies de transport par paquets dans les réseaux GSM, traditionnellement orientés circuit, afin d'en réduire les dépenses d'exploitation (opex). Pour ce problème, nous avons développé une approche simple et robuste, basée sur la résolution d'un programme linéaire dans $\mathbb{R}^3$ et permettant de synchroniser avec une précision extrêmement élevée (conformément aux exigences du standard GSM) la fréquence de deux horloges au travers d'un réseau paquet asynchrone sur lequel circulent des flux temps réel bidirectionnels de paquets estampillés. Cette approche a été déployée sur le terrain en conditions réelles.

La seconde partie de ce mémoire, quant à elle, présente une partie des travaux en optimisation que nous avons menés et que nous menons actuellement au sein de la Direction de la Recherche Technologique du Commissariat à l'Énergie Atomique, à Saclay, depuis fin 2007.

Le chapitre 4 présente succinctement nos activités de recherche sur le parallélisme embarqué. Ces travaux s'articulent autour des trois thèmes suivants : langages et modèles de programmation parallèles adaptés à de multiples contextes applicatifs (traitement du signal et de l'image, notamment), chaînes de compilation pour processeurs parallèles (que ce soit en termes de modélisation, de dimensionnement, de preuve et d'optimisation ainsi qu'en termes plus techniques d'édition des liens complexe et de manipulation des binaires), modèles d'exécution et architecture système. Ces travaux définissent le socle de notre contribution en optimisation combinatoire appliquée au parallélisme embarqué.

Dans le chapitre 5 nous étudions plusieurs problèmes combinatoires difficiles en relation avec le placement/routage de réseaux de processus sur architectures parallèles clusterisées : problèmes de partitionnement de graphes sous contraintes de capacité multidimensionnelles (pour la décomposition du réseau de processus en sous-ensembles de tâches cluster-compatibles sur le plan des ressources afin de minimiser les échanges inter-clusters), problèmes d'affectation quadratique (de manière à placer les sous-ensembles de tâches sur les clusters physiques en tenant compte d'un critère de distance, lié à la latence induite par le plan de routage) et problèmes de multi-flot avec contraintes d'unicité de chemins (pour le calcul des chemins d'acheminement sur le réseau sur puce sous contraintes de capacité des liens du réseau). Nous étudions également une version stochastique de notre problème de partitionnement de graphe.

Le chapitre 6 est consacré à une approche du problème de la vérification de l'absence de deadlock et du dimensionnement des tampons associés aux canaux d'un réseau de processus (pour une exécution sans deadlock, bornée en mémoire) à l'aide de la programmation linéaire. En particulier, nous avons montré comment ramener la preuve de l'absence de deadlock dans un tel réseau à l'incohérence d'un système linéaire d'inégalités en nombres entiers. Nous avons également étendu cette approche afin de fournir un oracle polynomial permettant de décider si un réseau de processus flot de données peut être exécuté sans deadlock, en mémoire bornée (et par là même de fournir une borne supérieure sur la taille des tampons permettant une telle exécution). Cet oracle est bien évidemment sûr, au sens où, si il peut éliminer des systèmes corrects, il ne laisse par contre pas passer de systèmes incorrects.

Enfin, le chapitre 7 résume les travaux réalisés dans le cadre de la thèse de Sergiu Carpov sur les problématiques combinatoires d'ordonnancement en relation avec le préchargement (possiblement spéculatif) de données pour les réseaux de processus flot de données

Ce mémoire se conclut par un chapitre sur nos perspectives de recherche dans le domaine de l'optimisation combinatoire et de la recherche opérationnelle.

Première partie

Problématiques d'affectation de ressources dans les autocommutateurs pour la téléphonie cellulaire

Les travaux présentés dans cette première partie ont été menés entre juillet 2001 et octobre 2007 dans le cadre de mes responsabilités d'architecte système au sein de la division Sous-Système Radio de la R&D de l'équipementier télécom Nortel.

Le chapitre 1 présente une sélection de problèmes industriels réels que nous avons résolus à l'aide des méthodes de la recherche opérationnelle. Parfois en nous raccrochant à des techniques connues, parfois en développant nos propres outils. Ces travaux très appliqués ont donné lieu à autant de méthodes de résolution opérationnelles qui sont ou ont été déployées en situation réelle.

Le chapitre 2 revient brièvement sur nos travaux de thèse de doctorat consacrés à l'étude d'un problème d'ordonnancement à contraintes de ressources. Il s'agit là encore d'un problème de terrain puisque posé dans le cadre de l'opérabilité d'un système temps réel réparti à haute disponibilité.

Enfin, le chapitre 3 est consacré à l'étude des problèmes de synchronisation des horloges posés par l'introduction de technologies de transport par paquets dans les réseaux GSM. Il s'agissait en 2007 d'un problème critique pour l'industrie des télécommunications mobiles, forcée d'évoluer vers ces technologies pour des raisons de coût sans pour autant mettre en péril le bon fonctionnement des réseaux. De nouveau, la méthode présentée, fondée sur la programmation linéaire, a été déployée en conditions réelles.

Chapitre 1

Une sélection de problèmes d'affectation de ressources

1.1 Introduction

Historiquement, l'industrie des télécommunications a toujours été l'un des principaux pourvoyeurs des problèmes combinatoires les plus redoutables, principalement dans les domaines liés à la conception, au dimensionnement et à l'exploitation des grands réseaux. La conception des équipements en charge de l'écoulement du trafic au cœur de ces réseaux, les autocommutateurs, se trouve aussi être à la source de problèmes combinatoires aussi divers que variés.

De manière abstraite, un autocommutateur est un système qui fournit une quantité limitée de certaines ressources (circuits, ports, liens, CPU, mémoire, etc.) interdépendantes qu'il convient d'utiliser efficacement. Dans un tel contexte, l'omniprésence de problématiques combinatoires n'est en rien surprenante. De plus, le caractère temps réel embarqué de ces systèmes impose un nombre important de contraintes, par exemple sur le temps d'exécution ou sur la taille du logiciel à embarquer, quant aux solutions algorithmiques utilisables, en particulier lorsqu'il s'agit de résoudre des problèmes NP-difficiles.

Dans ce chapitre, nous illustrons la diversité des problèmes combinatoires d'affectation de ressources que nous avons rencontrés en conception d'autocommutateurs pour la téléphonie cellulaire au travers d'une sélection représentative de problèmes réels. En particulier, toutes les méthodes de résolution présentées dans ce chapitre sont ou ont été déployées sur le terrain en situation réelle.

Pour se fixer les idées, le contexte industriel, est le sous-système radio (BSS) des réseaux GSM, le sous-ensemble du réseau responsable des transmissions sur l'interface radio, qui est à la charge des stations de base (BTS), ainsi que de la gestion de la ressource radio et du pilotage des transferts inter-cellulaires d'appels (handover), sous la responsabilité des contrôleurs de stations de base (BSC). En quelques mots, toujours pour se fixer les idées, une BTS est une entité com-

prenant des émetteurs-récepteurs (TRX). Les BTS couvrent, avec des porteuses différentes, des zones géographiques appelées cellules. Enfin, un BSC est un autocommutateur connecté d'un côté à plusieurs BTS et, de l'autre côté, au cœur de réseau. Il est en particulier responsable du premier niveau de concentration du trafic ainsi que du pilotage des BTS pour ce qui est de la gestion des ressources sur l'interface radio. Cf. par exemple Mouly & Pautet (1992) ou Lagrange et al. (2000) pour plus de détails sur l'architecture des réseaux GSM.

Ces travaux ont fait l'objet d'un article dans le journal *4OR* (Sirdey, 2007a).

1.2 Configuration optimale de cellule radio

Cette section traite du problème de configuration optimale d'une cellule radio par l'affectation de groupes de canaux radios prédéfinis appelés trames TDMA, aux émetteurs-récepteurs (TRX) d'une BTS (Sirdey, 2006a). Un TRX est un modem radio élémentaire pouvant émettre ou recevoir de manière continue sur une fréquence donnée.

1.2.1 Préliminaires

Soit T un ensemble de groupes de canaux radios. Soit X un ensemble de TRX. Un groupe peut être affecté à au plus un TRX et un TRX peut se voir affecter au plus un groupe. Également, l'ensemble X est augmenté d'un TRX factice noté d, qui peut se voir affecter un nombre arbitraire de groupes. Un groupe affecté au TRX d est effectivement non affecté et lorsqu'au moins un groupe n'est pas affecté on dit que la cellule est en mode dégradé dans la mesure où une partie du service configuré n'est pas rendu.

En raison de contraintes matérielles que nous ne détaillerons pas ici, un groupe ne peut pas forcément être affecté à n'importe quel TRX. L'ensemble des TRX auxquels un groupe $t \in T$ peut être affecté est noté $X(t)$. Notons que pour tout $t \in T$, d appartient à $X(t)$.

De plus, un ensemble S de services est défini. Chaque groupe $t \in T$ requiert un ensemble $S(t) \subseteq S$ de services et chaque TRX $x \in X$ fournit un ensemble de services noté $S(x) \subseteq S$. Un groupe $t \in T$ peut être affecté à un TRX $x \in X$ qui ne fournit pas l'intégralité des services requis par t, c'est-à-dire tel que $S(t) \setminus S(x) \neq \emptyset$. Lorsque cela arrive, la cellule est également en mode dégradé.

Enfin, chaque groupe possède un niveau de criticité $0, 1, 2, \ldots$ Par exemple, certains groupes doivent être affectés sinon la cellule ne peut passer aucun appel (typiquement le groupe du canal balise). Le niveau de criticité d'un tel groupe est généralement égal à 0 (le plus critique).

Grossièrement, le problème consiste alors à trouver la "meilleure" affectation des groupes aux TRX au sens d'une fonction économique définie à la section suivante.

1.2.2 Fonction économique

Pour chaque groupe $t \in T$ et chaque TRX $x \in X(t) \cup \{d\}$, on introduit les variables suivantes :

$$\delta_{tx} = \begin{cases} 1 & \text{si } t \text{ est affecté à } x, \\ 0 & \text{sinon.} \end{cases}$$

Nous allons maintenant définir une hiérarchie de fonctions économiques qui reflète les différentes facettes du problème.

Tout d'abord, les niveaux de criticité doivent être pris en compte. En particulier, les groupes de plus haute criticité doivent être affectés même si ce faisant tous les groupes de criticité moindre ne peuvent l'être. Ceci peut être réalisé en minimisant la fonction économique suivante :

$$f_1(\delta) = \sum_{t \in T} W(C(t))\delta_{tx}, \tag{1.1}$$

où $C(t)$ est le niveau de criticité du groupe $t \in T$ et, H étant la valeur la plus élevée de niveau de criticité, où la fonction de pondération W est définie récursivement par $W(H) = 1$ et, pour $c = H - 1, \ldots, 0$,

$$W(c) = W(c+1)(|T(c+1)| + 1),$$

avec $T(c) = \{t \in T : C(t) = c\}$.

Deuxièmement, le service rendu doit être pris en compte. Cela se fait en minimisant la fonction économique suivante :

$$f_2(\delta) = \sum_{t \in T} \sum_{x \in X(t) \setminus \{d\}} |S(t) \setminus S(x)|\delta_{tx}. \tag{1.2}$$

Soulignons que le terme $|S(t) \setminus S(x)|$ peut être remplacé par une somme pondérée sur l'ensemble $S(t) \setminus S(x)$ lorsque les services ont des impacts différents.

La troisième fonction économique permet de tenir compte des ajouts, des retraits et des défaillances de TRX. Par exemple, un TRX peut tomber en panne à tout moment et, lorsque la défaillance survient, le système doit reconfigurer la cellule de manière à assurer le meilleur fonctionnement possible avec un TRX de moins. La nouvelle affectation est ensuite obtenue en appliquant une séquence de réaffectations, sachant que lors d'une réaffectation, tous les appels en cours sur les canaux du groupe sont perdus et que ces canaux sont indisponibles pendant toute la durée de l'opération. Soit $R(t)$ le coût de réaffectation du groupe t (qui est typiquement fonction de son niveau de criticité) et soit

$$\gamma_{tx} = \begin{cases} 1 & \text{si } t \text{ est présentement affecté à } x, \\ 0 & \text{sinon.} \end{cases}$$

Il suit que $t \in T$ doit être réaffecté si et seulement si $\sum_{x \in X(t) \setminus \{d\}} (1 - \gamma_{tx}) \delta_{tx} = 1$. En conséquence, l'impact minimum est réalisé en minimisant :

$$f_3(\delta) = \sum_{t \in T} \sum_{x \in X(t) \setminus \{d\}} R(t)(1 - \gamma_{tx}) \delta_{tx}. \tag{1.3}$$

Enfin, les aspects liés aux services sont de nouveau pris en compte par la minimisation de la fonction suivante :

$$f_4(\delta) = \sum_{t \in T} \sum_{x \in X(t) \setminus \{d\}} (H - C(t) + 1) |S(x) \setminus S(t)| \delta_{tx}. \tag{1.4}$$

Ce dernier objectif permet de s'assurer, autant que faire se peut, que les groupes les plus critiques sont affectés aux TRX qui leurs sont le plus adaptés. Ceci de manière à diminuer la probabilité qu'ils soient victimes d'une réaffectation, par exemple sur défaillance TRX.

Pour obtenir la fonction économique du problème, les fonctions (1.1), (1.2), (1.3) et (1.4) sont agrégées en une seule fonction économique (linéaire)

$$f(\delta) = \sum_{i=1}^{4} \lambda_i f_i(\delta), \tag{1.5}$$

où les λ_i ont été choisis de manière à obtenir une hiérarchie stricte de critères i.e., de manière à garantir qu'un gain arbitrairement élevé dans un critère $j > i$ ne justifie pas une perte ne serait-ce que d'une unité dans le critère i. En ce sens, il s'agit d'un problème multicritère que les exigences applicatives permettent de ramener à un problème monocritère.

1.2.3 Méthode de résolution

Dans la mesure où la fonction économique (1.5) est linéaire, il est aisé de voir que le problème peut être modélisé par un flot sur un réseau de transport et résolu avec un algorithme de flot de coût minimum (en réalité il s'agit même d'un problème de couplage sur un graphe biparti mais l'introduction du TRX factice perturbe quelque peu la structure du problème).

La figure 1.1 donne un exemple de réseau de transport pour une instance à 3 groupes et 4 TRX. Les sommets s, t et d représentent respectivement la source, le puits et le TRX factice. Seules les capacités des arcs sont indiquées (le coût d'un arc reliant s à un sommet ou un sommet à t est de 0, pour les autres arcs le coût est donné conformément à la section précédente). L'existence systématique d'une solution est garantie par le fait que la capacité de l'arc $\{d, t\}$ est égale au nombre de groupes. Sur cet exemple, $X(t_1) = \{x_1, x_3, x_4, d\}$ et $X(t_2) = X(t_3) = \{x_2, d\}$ (rappelons que d appartient toujours à $X(t_i)$), en conséquence, t_2 ou t_3 sera affecté à d, i.e., non affecté. Étant donné un s-t-flot complet de coût minimum sur ce

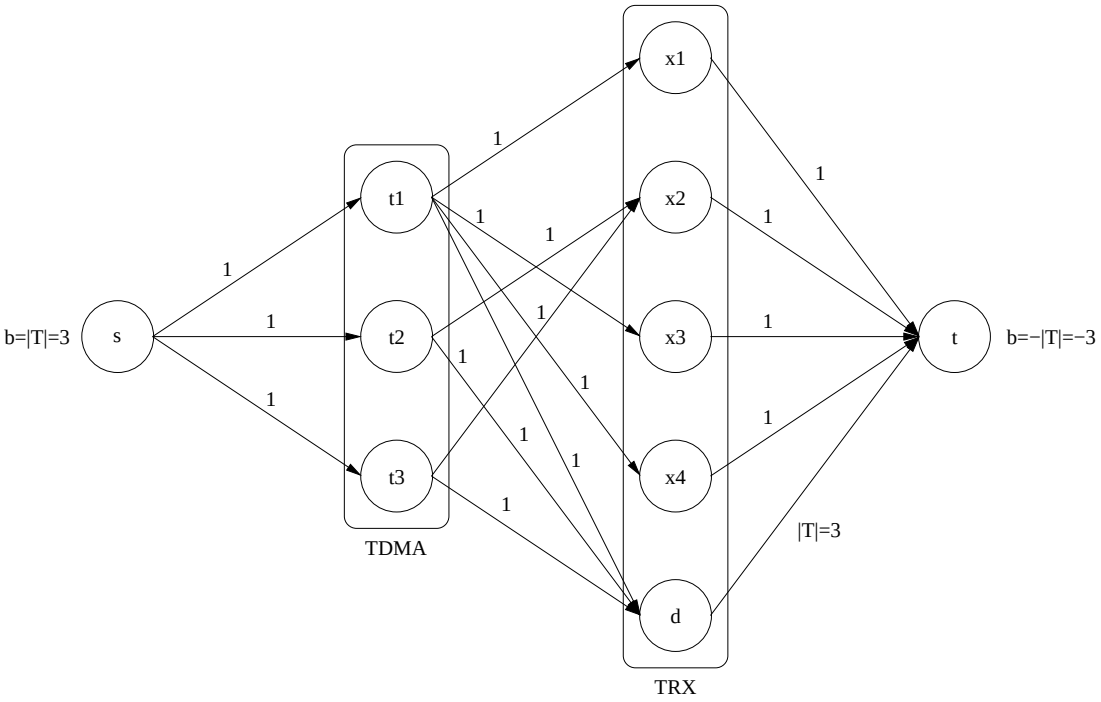


FIGURE 1.1 – Exemple de réseau de transport pour une instance du problème de configuration de cellule radio de la section 1.2 avec 3 groupes et 4 TRX.

réseau, le seul arc $\{t_i, x_j\}$ ou $\{t_i, d\}$ sur lequel s'écoule une unité depuis le sommet t_i indique le TRX, x_j ou d, auquel t_i est affecté.

In fine, notre implémentation s'est basée sur l'algorithme des plus courts chemins successifs (Ahuja et al., 1993 ; Korte & Vygen, 2000) en arithmétique entière sur 64 bits. Cette méthode a été déployée sur le terrain en situation réelle. Également, il est important de souligner que la méthode a été implémentée en moins de 200 lignes de code et qu'elle a remplacé avec succès les quelques milliers de lignes de code d'un algorithme ad hoc implémenté dans une version antérieure du système qui, en plus d'être difficilement maintenable, était connu pour ne pas toujours fonctionner de manière satisfaisante.

1.3 Gestion d'une interface PCM

Cette section est consacrée à un problème d'attribution de circuits sur les liens PCM [1] reliant deux autocommutateurs (Sirdey, 2004a). Les liens fournissent des circuits que l'on doit affecter à des cellules radios sachant que tous les circuits affectés à une cellule doivent l'être sur le même lien (on dit alors que la cellule est affectée au lien). Généralement, que ce soit à cause de la configuration ou

1. Un lien à modulation par impulsions codées (PCM) est un lien de communication offrant soit 31 (standard européen) soit 24 (standard nord-américain) canaux à 64 kbit/s.

de l'indisponibilité d'un certain nombre de liens, le besoin en circuits excède la capacité totale des liens ce qui signifie que certaines cellules se verront affecter un nombre de circuits inférieur à leur besoin. L'objectif est alors lié à la minimisation de cette demande insatisfaite.

1.3.1 Préliminaires

Soit C un ensemble de cellules radios et soit P un ensemble de liens PCM. Pour une cellule $c \in C$, soit $w(c)$ son besoin en circuits et, pour un lien $p \in P$, soit $K(p)$ sa capacité en circuits.

Généralement, comme nous l'avons déjà indiqué, la demande totale en circuits excède l'offre c'est-à-dire,

$$\sum_{c \in C} w(c) > \sum_{p \in P} K(p),$$

et le problème qui nous intéresse consiste dans un premier temps à trouver une affectation $f : C \longrightarrow P$ des cellules aux liens qui minimise

$$\sum_{p \in P} \max \left(0, \sum_{c \in C : f(c) = p} w(c) - K(p) \right).$$

Il se trouve que ce problème est équivalent à un problème de bin-packing à conteneurs "extensibles" (EBP) étudié par Dell'Olmo et al. (1998) qui consiste (en paraphrasant ci-dessus), étant donnée une liste de n nombres positifs $\alpha_1, \ldots, \alpha_n$ et m conteneurs $b_1, \ldots, b_m$ de capacités $k_1, \ldots, k_m$, à trouver une affectation $f : \{1, \ldots, n\} \longrightarrow \{1, \ldots, m\}$ telle que

$$\sum_{i=1}^{m} \max(k_i, \ell(b_i)),$$

où

$$\ell(b_i) = \sum_{j=1 \ldots n : f(j) = i} \alpha_j,$$

est minimum.

1.3.2 Méthode de résolution

Bien que le problème EBP soit NP-difficile au sens fort, il existe des algorithmes de résolution approchée simples avec des garanties de performance plutôt raisonnables.

En effet, dans le cas où les conteneurs sont homogènes sur le plan de la capacité, Dell'Olmo et al. (1998) ont établi que l'heuristique LPT, bien connue,

qui consiste à trier les objets par ordre de poids noncroissant dans une liste et qui affecte, à chaque itération, le premier objet de la liste au conteneur le moins chargé, est telle que pour toute instance I du problème EBP on a

$$\frac{\mathrm{LPT}(I)}{\mathrm{OPT}(I)} \leq \frac{13}{12} \approx 1.0833. \tag{1.6}$$

De plus, quand les conteneurs sont hétérogènes on sait (Dell'Olmo & Speranza, 1999), sous l'hypothèse (raisonnable) que

$$\max_{i=1,\ldots,n} \alpha_i \leq \min_{i=1,\ldots,m} k_i, \tag{1.7}$$

que l'heuristique LPT (légèrement modifiée pour choisir le conteneur avec la plus grande capacité résiduelle plutôt que la plus petite charge) est telle que pour toute instance I du problème on a [2],

$$\frac{\mathrm{LPT}(I)}{\mathrm{OPT}(I)} < 2(2-\sqrt{2}) \approx 1.1716. \tag{1.8}$$

Tout ceci motive l'algorithme suivant pour notre problème : trier les cellules par ordre noncroissant des besoins en circuits, $w_{c_1} \geq \ldots \geq w_{c_{|C|}}$, pour $i = 1$ à $|C|$ s'il existe un lien $p \in P$ avec une capacité résiduelle positive, i.e., telle que $K(p) > \sum_{j=1:f(c_j)=p}^{i-1} w_{c_j}$, alors affecter c_i au lien possédant la plus grande capacité résiduelle, sinon affecter c_i au lien proportionnellement le moins chargé, i.e., le lien qui minimise

$$\frac{1}{K(p)} \sum_{j=1:f(c_j)=p}^{i-1} w_{c_j}.$$

Il est clair que cet algorithme est équivalent à LPT lorsque les liens sont homogènes et donc qu'il fournit les mêmes garanties de performance (données par l'équation (1.6)) ainsi que lorsque les liens sont hétérogènes (un cas rare, mais néanmoins possible, dans notre contexte applicatif) auquel cas ses garanties de performance sont données par l'équation (1.8), sachant que l'hypothèse (1.7) est bien satisfaite dans notre cas.

Bien que l'algorithme finalement implémenté ait été un peu plus complexe pour tenir compte de quelques contraintes supplémentaires que nous ne détaillerons pas ici, le fait que des heuristiques aussi simples (et rapides) fournissent des garanties de performance acceptables a été particulièrement utile. En particulier, dans les discussions avec les équipes de validation qui sont très douées pour trouver des petits cas pathologiques mettant en défaut l'optimalité des solutions trouvées par un tel l'algorithme. La question étant alors de savoir s'il fallait y remédier ou pas sachant que dès lors que les garanties de performance étaient respectées, l'algorithme était conforme à ses spécifications. Ceci a permis d'éluder un certain nombre de demandes d'améliorations de fait illégitimes et donc de complexifier indûment le logiciel résultant.

2. Une borne serrée de $\frac{8}{7} \approx 1.1429$ est conjecturée.

1.3.3 L'analogie "parlementaire"

Nous allons maintenant considérer le problème de la répartition équitable des circuits d'un lien entre les cellules qui lui ont été affectées, lorsque le besoin excède la capacité.

Considérons un pays composé d'un certain nombre de départements et, démocratie représentative oblige, doté d'un parlement avec un petit nombre de sièges (devant la population totale du pays).

Un problème central, en théorie des systèmes électoraux, consiste à trouver une répartition équitable des sièges entre les départements sous certaines contraintes, en particulier celle que chaque département se voit allouer au moins un siège.

Il est généralement admis qu'un algorithme de répartition équitable doit satisfaire les exigences suivantes (Balinski & Young, 2001) :

1. lorsque le nombre de sièges augmente, chaque département doit se voir allouer au moins le même nombre de sièges (pas de paradoxe de l'Alabama) ;
2. lorsque la population d'un département augmente et que celle d'un autre diminue, le dernier ne doit pas se voir allouer plus de sièges au détriment du premier ;
3. la répartition doit être stable si l'on considère un sous-ensemble des départements et la somme des sièges qui leurs ont été alloués ;
4. lorsqu'elle est réalisable, l'algorithme doit trouver une solution parfaitement proportionnelle.

Il apparaît qu'un algorithme très simple, l'algorithme de Sainte-Laguë, répond à toutes ces exigences (ibid.), contrairement à des méthodes plus connues telles l'algorithme des plus forts restes.

Si l'on revient à notre problème d'allocation de circuits, l'analogie "parlementaire" consiste alors simplement à voir les cellules comme un ensemble de départements, leurs demandes en circuits comme les populations, le lien PCM comme un parlement et sa capacité comme le nombre de sièges de ce dernier. Plus généralement, cette analogie est parfaitement pertinente dans toute situation où un certain nombre de consommateurs doivent se partager équitablement une ressource disponible en quantité insuffisante pour satisfaire l'intégralité des besoins.

1.4 Maximisation d'une durée de vie sur batterie

Dans cette section, nous étudions un problème qui consiste à minimiser la consommation électrique d'une station de base (BTS) pendant une période d'indisponibilité du réseau électrique, sous une contrainte de service (Sirdey, 2005).

La forme de ce problème est dépendante de l'architecture matérielle de la BTS. En l'occurrence, la BTS est composée de cabinets contenant des modules radios eux-même contenant des TRX. Chaque TRX est dédié à une et une seule des cellules gérées par la BTS. La contrainte de service (définie par l'opérateur) consiste, pour chaque cellule, en la donnée du nombre de TRX qui doivent rester opérationnels aussi longtemps que possible en cas d'indisponibilité du réseau électrique. Dans la mesure où la consommation électrique de la station est proportionnelle au nombre de modules radios opérationnels (et non au nombre de TRX), la consommation minimum est réalisée, dans le cas monocabinet, quand la contrainte de service est satisfaite en maintenant le plus petit nombre possible de modules en état opérationnel (i.e., alimentés). Dans le cas multicabinet, par contre, on obtient un problème min-max : dans la mesure où chaque cabinet possède sa propre batterie indépendante, la durée pendant laquelle la contrainte de service va être satisfaite dépend de la durée de vie de la batterie du cabinet qui porte le plus grand nombre de modules radio alimentés.

Cette fonction a été introduite de manière à maximiser la durée pendant laquelle un réseau GSM peut maintenir la couverture d'une zone géographique, avec une capacité amoindrie, après une défaillance catastrophique du réseau électrique. Par exemple, après une catastrophe naturelle, les réseaux GSM ont, à plusieurs reprises, joué un rôle crucial en continuant à fournir des services d'appels d'urgence et même en aidant à la localisation des victimes alors que les réseaux filaires ne fonctionnent plus.

1.4.1 Préliminaires

Considérons une BTS composée d'un ensemble K de cabinets ainsi que d'un ensemble R de modules radios, chaque module contenant jusqu'à 3 TRX. Également, la BTS gère un ensemble C de cellules.

Soit W une matrice $|R| \times |C|$ à valeurs entières respectivement indicées sur les lignes et sur les colonnes par les éléments de R et de C et telle qu'étant donnés un module r et une cellule c, W_{rc} donne le nombre de TRX dédiés à la cellule c configurés sur le module r. Également, soit L une matrice $|R| \times |K|$ à valeurs binaires dont les indices de lignes et de colonnes sont respectivement les éléments de R et de K et telle qu'étant donné un module r et un cabinet k, $L_{rk} = 1$ si et seulement si le module r se trouve dans le cabinet k. Enfin, pour chaque cellule, un nombre N_c est donné qui spécifie le nombre de TRX à maintenir opérationnels pendant une période d'indisponibilité du réseau électrique.

À chaque module, on associe une variable binaire x_r telle que

$$x_r = \begin{cases} 1 & \text{si le module } r \text{ reste alimenté,} \\ 0 & \text{sinon.} \end{cases}$$

Le problème peut alors être formalisé à l'aide du programme mathématique

non linéaire suivant :

$$\begin{cases} \text{Minimiser } \max\limits_{k\in K} \sum\limits_{r\in R} L_{rk}x_r, & \\ \text{s. l. c.} & \\ \sum\limits_{r\in R} W_{rc}x_r \geq N_c & \forall c \in C, \\ x_r \in \{0,1\} & \forall r \in R. \end{cases} \tag{1.9}$$

Sous l'hypothèse que $\sum_{r\in R} L_{rk} = L$ (i.e., que chaque cabinet contient le même nombre de modules [3]), alors en posant $x_r = 1 - y_r$ et en réarrangeant un peu les termes on obtient le programme mathématique suivant :

$$\begin{cases} \text{Maximiser } \min\limits_{k\in K} \sum\limits_{r\in R} L_{rk}y_r, & \tag{1.10} \\ \text{s. l. c.} & \\ \sum\limits_{r\in R} W_{rc}y_r \leq \sum\limits_{r\in R} W_{rc} - N_c & \forall c \in C, \tag{1.11} \\ y_r \in \{0,1\} & \forall r \in R. \end{cases}$$

Énoncé en ces termes, il s'agit d'un problème de sac à dos max-min multidimensionnel.

Étant donnée une solution optimale du programme mathématique ci-dessus, il convient d'éteindre tous les modules pour lesquels $y_r = 1$ et, pour chaque cellule, d'éteindre $\sum_{r\in R} W_{rc}x_r - N_c$ TRX (sachant qu'aucune combinaison de TRX ne peut permettre d'améliorer la fonction économique sinon quoi la solution ne serait pas optimale).

1.4.2 Le problème du sac à dos max-min

Le problème du sac à dos max-min monodimensionnel a été étudié par Yu (1996), voir également Kellerer et al. (2004). Ce problème se présente lorsque l'on a à remplir un sac à dos avec une sélection d'objets de manière à maximiser le profit minimum sur un ensemble de scénarios fini, chaque objet ayant un profit dépendant du scénario (typiquement, l'intérêt d'emporter un parapluie dans ses bagages dépend du temps à destination). Ce problème a été initialement introduit pour une application de génie financier.

3. Lorsque cela n'est pas le cas il suffit d'ajouter des modules factices avec, pour un tel module, $W_{rc} = 0$ pour chaque cellule c.

Le problème de sac à dos max-min monodimensionnel s'exprime comme suit :

$$\begin{cases} \text{Maximiser } \min\limits_{k=1,\ldots,t} \sum\limits_{j=1}^{n} p_{kj} x_j, \\ \text{s. l. c.} \\ \sum\limits_{j=1}^{n} w_j x_j \leq c, \\ x_j \in \{0,1\} \qquad\qquad j = 1,\ldots,n, \end{cases} \tag{1.12}$$

Lorsque t est fixé, le problème n'est que faiblement NP-difficile et peut être résolu en temps pseudopolynomial par la programmation dynamique. L'algorithme de résolution se base sur la famille de problèmes suivante :

$$\begin{cases} z_j(d; v_1,\ldots,v_t) = \text{Maximiser } \min\limits_{k=1,\ldots,t} \sum\limits_{i=1}^{j} (p_{ki} x_i + v_k), \\ \text{s. l. c.} \\ \sum\limits_{i=1}^{j} w_i x_i \leq d, \\ x_i \in \{0,1\} \qquad\qquad i = 1,\ldots,j, \end{cases} \tag{1.13}$$

où $z_n(c; 0,\ldots,0)$ est la solution du problème (1.12). On a alors la relation de récurrence :

$$z_{j+1}(d; v_1,\ldots,v_t) = \begin{cases} z_j(d; v_1,\ldots,v_t) & \text{si } d < w_{j+1}, \\ \max\big\{ z_j(d; v_1,\ldots,v_t), & \\ \quad z_j(d - w_{j+1}; v_1 + p_{1,j+1},\ldots,v_t + p_{t,j+1}) \big\} & \text{sinon.} \end{cases} \tag{1.14}$$

avec

$$z_0(d; v_1,\ldots,v_t) = \min_{k=1,\ldots,t} v_k.$$

L'algorithme résultant est en $O(ncP^t)$, avec $P = \max_{k=1,\ldots,t} \sum_{j=1}^{n} p_{kj}$.

Le problème (1.12) se généralise sans difficulté à plusieurs dimensions donnant lieu à un problème de sac-à-dos max-min multidimensionnel, i.e. :

$$\begin{cases} \text{Maximiser } \min\limits_{k=1,\ldots,t} \sum\limits_{j=1}^{n} p_{kj} x_j, \\ \text{s. l. c.} \\ \sum\limits_{j=1}^{n} w_{ij} x_j \leq c_i \qquad\qquad i = 1,\ldots,m, \\ x_j \in \{0,1\} \qquad\qquad j = 1,\ldots,n, \end{cases}$$

où toutes les entrées (p_{kj}, w_{ij} et c_i) sont des entiers non négatifs.

1.4.3 Algorithme de résolution

Comme dans le cas du sac à dos linéaire (Minoux, 1983 ; Kellerer et al., 2004), d (dans les équations (1.13) et (1.14) ci-dessus) peut être interprété de manière vectorielle et les formules de récurrence ci-dessus peuvent être utilisées pour donner un algorithme en

$$O\left(n\left(\prod_{i=1}^{m} c_i\right)P^t\right),$$

donc pseudopolynomial, pour le sac à dos max-min multidimensionnel, lorsque t et m sont fixés.

Dans la mesure où $n = |R|$, $c_i \leq O(|R|)$ et $0 \leq v_k \leq \sum_{j=1}^{n} p_{kj} \leq |R|$, la complexité d'un tel algorithme, pour notre problème, est en

$$O\left(|R|^{|C|+|K|+1}\right),$$

donc polynomiale pour $|C|$ et $|K|$ fixés. Cela conduit à un algorithme en $O(|R|^7)$ pour notre application dans la mesure où 3 est une borne supérieure pour $|C|$ et pour $|K|$. Bien entendu, une telle complexité n'est pas satisfaisante en pratique et empêche une implémentation basée sur la programmation dynamique puisque les implémentations de ce type réalisent toujours leur pire cas en termes de temps et d'espace. À la place, nous avons donc utilisé une version vectorielle de la règle de récurrence ci-dessus dans le cadre d'un algorithme de recherche arborescente en profondeur d'abord utilisant une structure de données, appelée le cache, qui permet de filtrer l'intégralité de la redondance d'exploration. Le cache est une structure de données associative dont la clef correspond à un unique entier associé aux vecteurs à valeurs entières de la forme $(j; d_1, \ldots, d_m; v_1, \ldots, v_t)$ $(0 \leq j \leq n$, $0 \leq d_i \leq c_i$ and $0 \leq v_k \leq \sum_{j=1}^{n} p_{kj})$. Un élément du cache est alors définie comme une paire

$$\{z_j(d_1, \ldots, d_m; v_1, \ldots, v_t), x_j\},$$

où x_j est une variable booléenne qui indique si oui ou non l'objet j appartient à la solution optimale du sous-problème $z_j(d_1, \ldots, d_m; v_1, \ldots, v_t)$, ce dernier champ permettant de reconstruire la solution une fois l'exploration terminée. Donc, le sous-arbre enraciné au nœud $(j; d_1, \ldots, d_m; v_1, \ldots, v_t)$ de l'arbre d'exploration est exploré si et seulement si le cache ne contient pas déjà un élément associé à ce nœud. S'il est exploré alors un élément est ajouté au cache à la fin de l'exploration du sous-arbre. Notons que l'on suppose que le cache est implémenté à l'aide d'une structure de données à accès logarithmique (Knuth, 1998).

L'algorithme ci-dessus peut être vue comme la réinterprétation d'une formule de récurrence issue de la programmation dynamique comme une règle de dominance utilisée dans une recherche arborescente. Ce type de réinterprétation est discuté plus en détails dans Ibaraki (1977).

En pratique, il se trouve que cet algorithme est capable de résoudre les plus grosses instances réelles (3 cabinets, 3 cellules et 24 TRX par cellule) en un peu plus de 1 seconde sur un PC portable lambda, un temps de calcul totalement acceptable puisque de toute façon les indisponibilités du réseau électrique d'une durée inférieure à la minute sont filtrées par le système. La table 1.1 illustre les performances de l'algorithme, à noter que pour toutes les instances $\sum_r W_{r1} = \sum_r W_{r2} = \sum_r W_{r3} = 24$.

N_1	N_2	N_3	# mod.	# TRX	CPU (s)	taille cache
12	12	12	24	72	0.510	91 022
5	7	11	24	72	0.971	161 178
6	6	6	24	72	1.201	200 190
7	6	4	24	72	1.221	199 422
2	2	2	24	72	1.331	223 463

TABLE 1.1 – Illustration des performances de notre algorithme de recherche arborescente pour le problème de maximisation d'une durée de vie sur batterie (section 1.4) sur un ensemble d'instances de la plus grande taille pratique.

1.5 Conclusion

Nous avons présenté, dans ce chapitre, une sélection de problèmes représentative du contexte des autocommutateurs pour la téléphonie cellulaire.

Cette sélection ne pouvait pas être exhaustive. En particulier, nous avons également travaillé sur des problèmes de partitionnement consistant à affecter des processus de traitement d'appels, ainsi que leurs réplicas passifs, aux processeurs d'un BSC à architecture répartie de manière à garantir certaines propriétés d'équirépartition de la charge et de tolérance aux pannes, tout en respectant des contraintes de capacité multidimensionnelles sur les processeurs (Sirdey, 2004b). Ces travaux ont fait l'objet d'un article dans les actes de l'*International Network Optimization Conference* (Sirdey et al., 2003). On peut également mentionner une application de la théorie des flots dans le contexte d'un problème de configuration dynamique de DSP dédiés à l'encodage d'appels voix, où la fonction économique optimisée de manière en-ligne par le système était définie comme le coût d'un flot de coût minimum sur un réseau de transport adéquat paramétré par des prévisions à court terme d'intensité du trafic et par les capacités de traitement des configurations induites par les décisions possibles de configuration ou de reconfiguration des DSP (Sirdey et al., 2005). Ce système d'allocation était alors capable d'adapter sa capacité de traitement à un parc de mobiles de capacités hétérogènes, inconnues au démarrage et changeantes au cours du temps.

Chapitre 2

Ordonnancement de migrations de processus

2.1 Introduction

Ce chapitre résume succinctement les travaux que j'ai réalisés dans le cadre de ma thèse de doctorat, thèse que j'ai menée de mai 2004 à mars 2007, *à ma propre initiative et en parallèle de mes responsabilités opérationnelles d'architecte système* au sein de la division R&D Sous-Système Radio de Nortel (Sirdey, 2007c, 2008a). Ces travaux, qui ont été réalisés en collaboration avec Jacques Carlier, Hervé Kerivin et Dritan Nace (avec Jacques et Dritan en tant que directeurs de thèse), s'inscrivent naturellement dans le thème de cette première partie. Philippe Baptiste, Professeur à l'École Polytechnique, et Ridha Mahjoub, Professeur à l'Université Paris Dauphine, ont rapporté ma thèse.

Cette thèse a traité de l'étude d'un problème d'ordonnancement NP-difficile au sens fort à contraintes de ressource : le *problème de reconfiguration*. Ce problème était lié à l'opérabilité de certains systèmes répartis temps réel à haute disponibilité, tels un BSC [1] à l'époque commercialisé par Nortel.

En quelques mots, étant donné un système réparti composé de processeurs sur lesquels tournent des processus de traitement d'appels, notre problème consistait à ordonner les déplacements de certains de ces processus de telle manière que des contraintes de capacité sur les processeurs du système ne soient jamais violées. Les situations de blocage étaient alors résolues en arrêtant temporairement des processus donc en renonçant temporairement à rendre une partie du service, l'objectif étant naturellement d'avoir le moins possible recours à ce mécanisme.

Dans notre contexte industriel, ce problème s'est présenté lorsqu'il s'agissait de restaurer, en heure creuse mais sans arrêt du système, des propriétés (par

1. Un BSC (Base Station Controller) est un autocommutateur en charge de la gestion de la ressource radio et du premier niveau de concentration du trafic dans un réseau GSM (se reporter à la section 1.1 pour quelques détails supplémentaires).

exemple l'équirépartition de la charge de traitement d'appels) qui se perdaient au gré des inévitables défaillances de processeurs et des modifications dynamiques de l'ensemble des équipements (principalement des stations de base) gérés par le système. D'autres motivations telles que l'obtention de répartitions de processus favorables à certaines stratégies de mises à jour logicielle sans arrêt du système méritent également d'être mentionnées.

2.2 Premiers résultats

2.2.1 Énoncé du problème, complexité

Soit l'ensemble des processus et l'ensemble des processeurs d'un système réparti. Un état du système consiste en une distribution d'un sous-ensemble des processus sur les processeurs de manière à ce que les ressources consommées sur ces derniers n'excèdent pas leur capacité. Un processus peut être déplacé d'un processeur vers un autre de deux manières : soit il est migré, auquel cas il consomme des ressources sur les processeurs source et cible du déplacement pour la durée de la migration, soit il est interrompu, c'est-à-dire arrêté puis ultérieurement redémarré sur le processeur cible du déplacement. Soit M l'ensemble des déplacements nécessaires afin de faire passer le système d'un état arbitraire, l'état initial, vers un autre, l'état final, notre problème consiste à trouver $I \subset M$ et σ un ordonnancement réalisable des déplacements de $M \setminus I$ tel que $\sum_{m \in I} c_m$ soit minimum, c_m étant le coût d'interrompre le processus associé à m et I l'ensemble des déplacements interrompus (les interruptions étant toutes réalisées au début).

Ce problème est NP-difficile au sens fort par restriction au problème 3-partition.

La figure 2.1 représente une instance du problème de reconfiguration pour un système à 10 processeurs, une ressource et 46 processus. La capacité de chacun des processeurs dans l'unique ressource est égale à 30 et la somme des consommations des processus est égale à 281. Les figures d'en haut et d'en bas représentent respectivement l'état initial et l'état final. Par exemple, le processus 23 doit être déplacé du processeur 2 vers le processeur 6.

2.2.2 Cas particuliers polynomiaux

Nos travaux sur les cas particuliers polynomiaux se basent sur la notion de multigraphe de transfert que nous avons introduite. Le multigraphe de transfert est défini comme le multigraphe [2] orienté dont les sommets sont les processeurs et les arcs les déplacements de processus.

2. Les arcs parallèles sont autorisés, les boucles ne le sont pas.

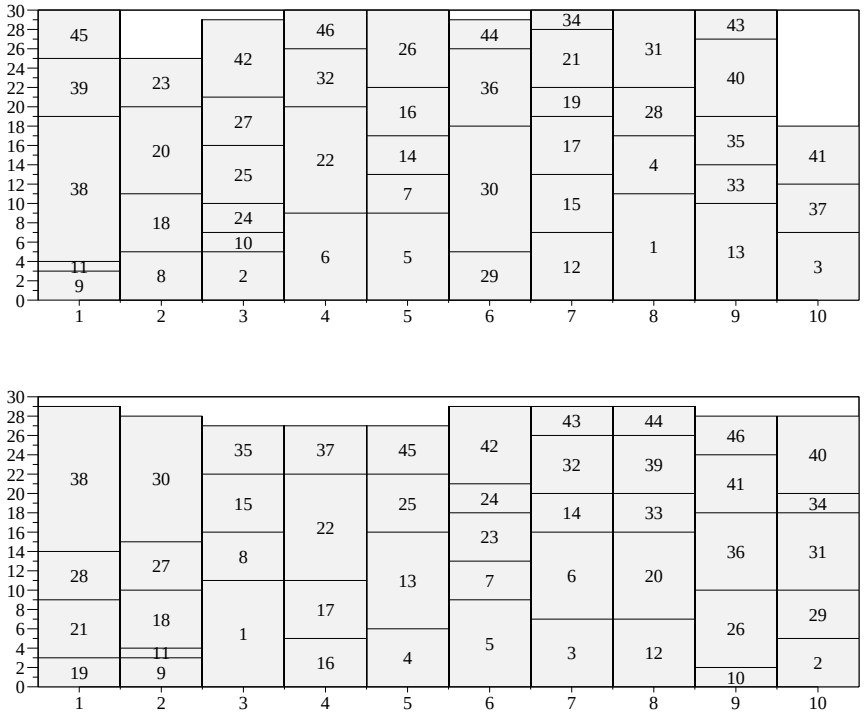


FIGURE 2.1 – Exemple d'instance du problème de reconfiguration.

Lorsque le multigraphe de transfert ne contient pas de circuit, nous avons montré que les déplacements peuvent toujours être ordonnés de manière à ce qu'aucune interruption ne soit nécessaire. Un tel ordonnancement se déduit aisément d'un ordre topologique des sommets du graphe, c'est-à-dire d'une fonction f de l'ensemble des sommets vers l'ensemble des entiers qui vérifie $f(v) < f(w)$ pour tout arc $\{v, w\}$ appartenant au graphe.

De plus, lorsque le multigraphe de transfert contient des circuits, nous avons montré qu'il convient de traiter ses composantes fortement connexes indépendamment et selon un ordre qui se déduit d'un ordre topologique des sommets du graphe de transfert réduit (graphe obtenu en contractant les composantes fortement connexes). Ceci nous donne une première méthode de décomposition.

Par ailleurs, lorsque les processeurs n'offrent qu'un seul type de ressources et lorsque tous les processus ont la même consommation, nous avons montré que le problème de reconfiguration peut être résolu de manière efficace. Pour ce faire, nous avons proposé un algorithme, démontré sa validité et prouvé que sa complexité est bien polynomiale.

Ces résultats ont fait l'objet d'un article dans le *European Journal of Operational Research* (Sirdey et al., 2007).

2.3 Résolution exacte

2.3.1 Un algorithme de branch-and-bound

Sur le plan de la résolution exacte dans le cas général, nous avons d'abord approché le problème de manière combinatoire, c'est-à-dire en exploitant directement sa structure dans le cadre d'un algorithme d'énumération par séparation et évaluation ou branch-and-bound.

Au lieu de chercher, à partir de l'état initial, à atteindre l'état final, il s'avère plus fructueux, dans le cadre d'un tel algorithme, de chercher à améliorer la pire des solutions (celle qui consiste à interrompre tous les déplacements), en évitant certaines interruptions. Selon ce point de vue, une solution partielle est représentée par un triplet composé d'un ensemble d'interruptions définitives, d'un ensemble d'interruptions non définitives et d'un ordonnancement admissible des déplacements non interrompus. Typiquement, un déplacement par interruption peut être évité si la capacité résiduelle du processeur source de ce déplacement est toujours supérieure à sa consommation durant l'exécution de l'ordonnancement des déplacements non interrompus. Si c'est le cas alors le processus déplacé peut rester sur le processeur source durant l'exécution de l'ordonnancement et, ensuite, être déplacé vers le processeur cible.

Cette manière de voir le problème nous a permis d'obtenir des bornes inférieures, la plus intéressante d'entre elles nécessitant la résolution d'un problème de sac à dos par processeur. Nous avons utilisé cette borne dans le cadre d'un

algorithme de branch-and-bound exploitant aussi une règle de dominance assez générale. Cette dernière permet d'éliminer plusieurs sources importantes de redondance. L'une d'entre elles, par exemple, concerne les ordonnancements totaux équivalents car ils induisent les mêmes ordonnancements restreints pour chacun des processeurs.

Nous avons illustré la pertinence pratique de cet algorithme en l'utilisant pour attaquer des instances difficiles[3]. En particulier, nous avons pu résoudre exactement des instances ayant jusqu'à 190 déplacements en moins de vingt minutes. En pratique néanmoins, si le temps de calcul est limité à vingt minutes, l'algorithme n'a de bonnes chances de succès que jusqu'à de l'ordre de 50 déplacements.

Cet algorithme a été publié dans l'article au *European Journal of Operational Research* susmentionné.

2.3.2 Approche polyèdrale

Toujours sur le plan de la résolution exacte dans le cas général, nous avons ensuite abordé le problème de manière géométrique, à l'aide de l'approche polyèdrale.

Cette approche trouve ses fondements dans une interprétation géométrique des problèmes combinatoires : lorsque la fonction économique est linéaire, c'est l'étude du polytope (un polytope est un polyèdre borné) correspondant à l'enveloppe convexe des vecteurs caractéristiques des solutions du problème qui guide la conception de l'algorithme de résolution. En particulier, les inégalités nécessaires à la description de ce polytope (on les appelle des facettes) sont extrêmement pertinentes quant à la mise en œuvre d'algorithmes de résolution exacte performants.

L'application des méthodes polyédriques à notre problème a tout d'abord requis l'étude théorique de deux polytopes que nous avons introduits : le polytope des sous-tournois sans-circuit et le polytope des programmes de déplacements, le premier étant une relaxation du second. Nous avons, en particulier, étudié les propriétés élémentaires de ces polytopes (dimension et facettes élémentaires, notamment), exhibé plusieurs classes d'inégalités qui en définissent des facettes et étudié les problèmes de séparation associés. L'idée était de commencer par l'étude du polytope des sous-tournois sans-circuit, d'un abord plus direct que celui des programmes de déplacements, de manière à faciliter l'étude théorique de ce dernier. Cette approche s'est avérée fructueuse : nous avons pu montrer que les classes de facettes que nous avons identifiées pour le polytope des sous-tournois sans circuit définissent aussi des facettes du polytope des programmes de déplacements sous des hypothèses peu restrictives. Aussi, nous avons exhibé deux classes d'inégalités, les contraintes dites de s- et de t-recouvrement, qui définissent des facettes du polytope des programmes de déplacements, sous des

3. En termes de taille, mais aussi de capacité résiduelle des processeurs.

conditions peu restrictives, et qui sont séparables en temps pseudopolynomial par résolution de quelques problèmes de sac à dos.

Le passage à la pratique s'est fait par le biais d'un algorithme de branch-and-cut dont le principal ingrédient est une relaxation linéaire comprenant un nombre exponentiel de contraintes qui définissent des facettes du polytope des programmes de déplacements. Cette relaxation, dont on peut théoriquement venir à bout en temps pseudopolynomial à l'aide de l'algorithme de l'ellipsoïde, est résolue à chaque nœud de l'arbre de recherche par un algorithme de coupe.

Là encore, nous avons illustré la pertinence pratique de cet algorithme en l'utilisant pour attaquer des instances difficiles. En particulier, nous avons pu résoudre exactement des instances ayant jusqu'à 119 déplacements (pour un système à 70 processeurs) en moins de quatre heures. En pratique cependant, si le temps de calcul est limité à quatre heures, l'algorithme n'a de bonnes chances de succès que jusqu'à de l'ordre de 80 déplacements. Lorsque la taille des instances augmente, l'algorithme a néanmoins été en mesure de fournir des solutions approchées de bonne qualité c'est-à-dire, ici, prouvées ne se situer qu'à moins de 5% (généralement moins) de l'optimum, à la quasi-totalité des instances (de taille allant jusqu'à 180 déplacements) sur lesquelles nous l'avons essayé, toujours sous la contrainte d'une limitation du temps de calcul à quatre heures.

Notons aussi que nous avons utilisé l'algorithme de recuit simulé présenté ci-après pour obtenir une bonne solution initiale, ce qui permet un premier élagage significatif de l'arborescence de recherche. Ceci illustre la complémentarité pratique entre une méthode de résolution approchée soignée et un algorithme polyédrique, complémentarité qui s'avère d'autant plus pertinente que les instances sont de grande taille.

Ces résultats ont été publiés dans un numéro spécial *Polyhedra and Combinatorial Optimization* du journal *RAIRO Operations Research* (Sirdey & Kerivin, 2007) ainsi que dans deux rapports de recherche (Sirdey & Kerivin, 2006 ; Kerivin & Sirdey, 2006).

2.4 Résolution approchée

2.4.1 Résolution par recuit simulé

En matière de résolution approchée, enfin, nous avons donc abordé le problème à l'aide de la méthode du recuit simulé.

Sur le plan théorique nous avons introduit la notion de solution (α, β)-acceptable, où β définit une exigence de qualité (distance à l'optimum) et où α est la probabilité de satisfaire cette exigence. Nous avons ensuite utilisé la théorie des chaînes de Markov, sous-jacente à la méthode du recuit simulé, afin de dégager des conditions sous lesquelles des solutions (α, β)-acceptables peuvent être obtenues. Enfin, nous avons appliqué ces résultats de manière à obtenir un algorithme

pseudopolynomial de résolution approchée pour notre problème.

Bien que notre raisonnement ne soit au final qu'heuristique, car il nous faudrait fournir une caractérisation pratique de la vitesse de convergence de certaines chaînes de Markov vers leur loi stationnaire [4], nous l'avons vérifié empiriquement.

À cet effet, nous avons paramétré notre algorithme de manière à ce qu'il produise des solutions $(95\%, 5\%)$-acceptables, c'est-à-dire des solutions situées à moins de 5% de l'optimum 95 fois sur 100, et, grâce aux résultats obtenus à l'aide de nos méthodes de résolution exacte, montré que l'algorithme a effectivement fourni des solutions 5%-acceptables à 97.74% des 1020 instances difficiles sur lesquelles nous l'avons essayé. Ceci, ainsi que l'analyse du nombre d'essais nécessaires à l'obtention d'une solution 5%-acceptable, suggère que l'algorithme est, en pratique, capable de produire des solutions $(95\%, 5\%)$-acceptables (avec une probabilité comprise entre 94.08% et 96.76%).

Cette approche a fait l'objet d'un article dans le *Journal of Heuristics* (Sirdey et al., 2009a).

2.4.2 GRASP

En complément des algorithmes présentés ci-dessus, nous avons travaillé sur une heuristique rapide, peut-être moins satisfaisante sur le plan théorique, basée sur GRASP (Greedy Randomized Adaptive Search Procedure), une métaheuristique, introduite dans les années 90 par Feo & Resende (1989, 1995), dont l'intérêt réside dans la combinaison des méthodes gloutonnes avec les méthodes de voisinage. Il s'agit d'une méthode à démarrages multiples : à chaque étape un algorithme glouton randomisé est utilisé pour construire une solution réalisable dont le voisinage est exploré. La meilleure solution ainsi obtenue est conservée.

Nous avons introduit un algorithme par construction progressive dont le principe repose sur l'insertion non invalidante de déplacements dans un ordonnancement réalisable. Cet algorithme possède un certain nombre de bonnes propriétés : à chaque étape tous les déplacements se voient offrir plusieurs opportunités d'insertion et toutes les interruptions sont remises en question. Une fois randomisé à la manière de Hart & Shogan (1987), cet algorithme nous donne le premier ingrédient de la méthode GRASP. Nous avons ensuite complété ce schéma par une méthode de voisinage simple, basée sur une stratégie d'échange entre les déplacements interrompus et ceux qui ne le sont pas : il s'agit simplement de s'assurer qu'interrompre un déplacement non interrompu ne permet pas d'insérer, sans l'invalider, un déplacement interrompu plus coûteux dans l'ordonnancement réalisable.

Malgré une distance moyenne à l'optimum satisfaisante sur notre base d'instances (1.68%), cet algorithme souffre de pouvoir systématiquement manquer l'optimum sur certaines petites instances pathologiques. En ce sens, bien qu'elle

4. Il s'agit là d'un problème extrêmement difficile.

soit significativement moins coûteuse en temps de calcul que notre algorithme de recuit, la qualité des solutions obtenues à l'aide de cette approche s'avère moins stable sur les petites instances, ce qui n'est pas nécessairement un problème fondamental en pratique.

Cet algorithme a été publié bien après la soutenance dans un numéro spécial *Metaheuristics and Real-World Problems* de l'*International Journal of Innovative Computing and Applications* (Sirdey et al., 2010).

2.5 Perspectives

Dans notre mémoire de thèse, nous avions identifié un élargissement du champ d'application de ces travaux aux problèmes de reconfiguration de chemins de routage dans les réseaux MPLS, suite à une suggestion de Klopfenstein (2008). Nous avions ainsi rédigé un premier rapport technique en ce sens (Sirdey, 2006b).

Aujourd'hui, il est à prévoir que ces travaux trouvent un nouveau terrain d'application dans le domaine de la compilation pour les architectures multicœurs (cf. chapitre 4). Ceci, en particulier, lorsqu'il va s'agir d'adapter les algorithmes d'affectation de ressources présentés au chapitre 5 afin de tenir compte de scénarios de défaillances clusters, par exemple, et de calculer hors ligne des plans de migration de tâches sous contraintes de ressources cluster et réseau. Des premiers travaux en ce sens ont été réalisés en collaboration avec Thomas Megel, doctorant au sein de mon laboratoire d'appartenance et sont en cours de publication.

Chapitre 3

Synchronisation d'horloges

3.1 Introduction

Ce chapitre traite du problème de la synchronisation des horloges dans les réseaux de téléphonie mobile posé par l'introduction de technologies de transport par paquets. Ces travaux ont été réalisés en 2007 et sont le fruit d'une collaboration avec François Maurice, à l'époque "fellow" architecte système au sein de la division R&D Sous-Système Radio de Nortel et aujourd'hui chez Alcatel-Lucent. Ils ont donné lieu à une publication dans le journal *4OR* (Sirdey & Maurice, 2008), un brevet (Maurice & Sirdey, 2010) et surtout à une méthode de synchronisation des horloges opérationnelle et aujourd'hui déployée.

Autour de 2007, l'accession de la technologie GSM à un marché de masse (plusieurs milliards d'utilisateurs dans plusieurs centaines de pays) a induit une pression élevée quant à la maîtrise des dépenses d'investissement (capex) et d'exploitation (opex) des réseaux. Sur le plan des opex, une amélioration clef consistait à adapter l'infrastructure de transmission synchrone traditionnelle du GSM à des technologies IP, intrinsèquement asynchrones mais de coût beaucoup plus faible. Ce point était particulièrement critique sur l'interface entre les stations de base (BTS), les "antennes" du réseau, et les contrôleurs de stations de base (BSC), autocommutateurs en charge du premier niveau de concentration du trafic [1]. Ceci dans la mesure où la majeure partie des opex de transmission pouvait être attribuée à ce sous-ensemble du réseau.

Cette tâche n'était pas sans difficulté.

En particulier, un des prix à payer pour l'efficacité spectrale du standard GSM est qu'un réseau ne peut fonctionner correctement que si les BTS émettent une fréquence balise extrêmement précise et stable sur l'interface radio afin de maintenir les mobiles synchronisés avec le réseau. Au niveau de la BTS, l'exigence

1. Nous renvoyons le lecteur au traité de Mouly & Pautet (1992) pour plus détails sur l'architecture des réseaux GSM. Cf. également section 1.1.

normative (3GPP, 2001) de précision est de ± 50 parties par milliard [2] (PPB). En d'autres termes, lorsque l'horloge du réseau bat un milliard de fois, celle de la BTS doit battre 1000000000 ± 50 fois. Précisons également que, pour ce qui concerne le standard GSM, seule importe la fréquence, la phase pouvant être ignorée.

Jusqu'à l'introduction des technologies de transport par paquets, l'approche usuelle consistait à installer une horloge de référence de très haute précision au BSC [3] et à maintenir la calibration de l'oscillateur local de la BTS à l'aide de la fréquence de synchronisation des signaux de son interface TDM (qui sont par construction asservis à l'horloge du BSC). Dans une telle configuration, une BTS avec un oscillateur à quartz de qualité modérée pouvait tenir l'exigence des 50 PPB quasiment indéfiniment. Dès lors que la BTS perd son interface TDM, et donc les précieux signaux de synchronisation qu'elle véhicule, elle devient isolée de sa source de synchronisation. Trois principales solutions ont été envisagées pour résoudre ce problème : installer une horloge GPS externe pour synchroniser la BTS, y embarquer un oscillateur haute précision (par exemple, une petite horloge au rubidium) ou trouver un moyen de véhiculer des données de synchronisation entre le BSC et la BTS via le réseau paquet asynchrone. Pour plusieurs raisons évidentes de coût mais aussi de faisabilité de déploiement, c'est cette dernière approche qui a été retenue.

Dans ce chapitre, nous présentons donc une méthode simple et robuste permettant d'estimer le biais de l'horloge d'une BTS relativement à celle de son BSC, lorsque que des flux temps réel de paquets estampillés [4] non nécessairement de taille fixe (par exemple les appels voix paquétisés) s'écoulent du BSC vers la BTS (flux descendant) ainsi que de la BTS vers le BSC (flux montant). Sur une période de temps relativement courte, la BTS collecte des paires de dates d'envoi et de réception, respectivement relativement au BSC et à la BTS pour le flux descendant et relativement à la BTS et au BSC pour le flux montant [5], et utilise ces données afin d'estimer le biais de son horloge locale par résolution, nous allons le voir, d'un programme linéaire dans $\mathbb{R}^3$. Ce biais estimé sert alors à appliquer une correction à l'oscillateur local.

Dans la suite de ce chapitre, les horloges du BSC et de la BTS sont respectivement désignées horloges maître et esclave.

2. Exigence qui peut être relachée à 100 PPB pour une certaine classe de petites BTS.

3. Correspondant soit à l'horloge de référence du cœur de réseau (qui est soit une horloge au césium soit une horloge au rubidium asservie sur le GPS) propagée sur l'infrastructure TDM du réseau, soit asservie sur le GPS avec holdover G.812 (ITU-T, 2004).

4. En pratique, afin d'éviter le bruit de mesure du à la traversée des couches logicielles de la pile de protocoles lors de l'encodage d'un paquet, l'estampillage est réalisé au niveau matériel juste avant (respectivement après) son envoi (respectivement sa réception) par la couche physique. Une conséquence de ceci est que l'estampille d'un paquet ne peut être véhiculée dans le paquet lui même mais dans un paquet de suivi (IEEE, 2007), typiquement le paquet suivant dans le flux temps réel.

5. Les estampilles du flux montant doivent donc être redescendues à la BTS.

3.2 Préliminaires sur les horloges

3.2.1 Terminologie

Nous rappelons dans cette section quelques notions fondamentales concernant les horloges (Moon et al., 1999 ; Bi et al., 2006).

Une horloge est une fonction continue par morceaux $C : \mathbb{R} \longrightarrow \mathbb{R}$ deux fois dérivable sauf sur un ensemble fini de points. Soit C_A et C_B deux horloges. Le décalage ("offset" en anglais) de C_A relativement à C_B à l'instant t est donné par $C_A(t) - C_B(t)$. La fréquence, i.e. le taux de progression de l'horloge, de C_A à l'instant t correspond à $C'_A(t)$. Le biais ("skew" en anglais) de C_A relativement à C_B à l'instant t est défini par $C'_A(t) - C'_B(t)$. Également, le rapport d'horloge de C_A relativement à C_B à l'instant t est donné par $C'_A(t)/C'_B(t)$. Enfin, la dérive ("drift" en anglais) de C_A relativement à C_B à l'instant t correspond à $C''_A(t) - C''_B(t)$.

Sur une période de temps suffisamment courte (typiquement quelques minutes), la dérive est généralement négligée et, en conséquence, la fréquence, le biais et le rapport d'horloge peuvent être considérés constant.

Les notions de biais et de rapport d'horloge sont utilisées de manière interchangeable dans ce chapitre (en fait, nous travaillons principalement avec le rapport d'horloge) et il est trivial de passer de l'un à l'autre dans la mesure où $\delta = C'_A - C'_B = C'_A - C'_A/\alpha = (1 - 1/\alpha)\, C'_A$ avec $\alpha = C'_A/C'_B$.

3.2.2 État de l'art

L'approche vraisemblablement la plus simple pour estimer un biais d'horloge consiste, sur la base de mesures monodirectionnelles, à mettre en abscisse les intervalles de temps inter-réceptions de paquets (mesurés avec l'horloge esclave) et en ordonnée les intervalles inter-émissions (mesurés avec l'horloge maître) puis à procéder à une régression linéaire. Cette approche, néanmoins, est connue pour donner des résultats médiocres (Paxson, 1998) et n'est pas exempte d'hypothèses difficilement vérifiables (finitude des moments d'ordre 1 et 2 notamment) sur la loi de probabilité du délai d'acheminement des paquets.

Duda et al. (1987) ont proposés plusieurs approches motivées par une autre interprétation géométrique du problème : en mettant en ordonnée les temps mesurées avec l'horloge esclave (dates de réception pour les paquets descendants et date d'émission pour les paquets montants) et en abscisse les temps mesurés avec l'horloge maître (dates d'émission pour les paquets montants et date de réception pour les paquets descendants), on obtient un petit couloir vide qui encadre la droite liant le temps maître et le temps esclave (voir la section 3.3 et la figure 3.2). Par contre, la largeur du couloir dépend du délai minimum d'acheminement des paquets qui est généralement non négligeable. Par essence, l'approche que nous proposons à la section 3.3 est également fondée sur cette interprétation géométrique.

Moon et al. (1999) ont été les premiers à proposer une méthode basée sur la programmation linéaire pour l'estimation d'un biais d'horloge. Leur approche consiste à chercher à positionner une droite sous le (et aussi proche que possible du) nuage de points obtenu en mettant les mesures de délais d'acheminement en abscisse et les dates d'envois en ordonnée. Le coefficient directeur de la droite donne alors une estimation du biais et trouver la droite requiert la résolution d'un programme linéaire dans $\mathbb{R}^2$. Cette approche a par la suite été raffinée par Zhang et al. (2002) qui ont proposé d'autres fonctions économiques et également d'exploiter l'enveloppe convexe du nuage de points afin de tolérer les réinitialisations de l'horloge maître (un événement qui ne peut se produire dans le cadre de notre contexte applicatif au sens où l'horloge du BSC n'est jamais réinitialisée et, dans l'éventualité d'un redémarrage du BSC, toutes les BTS qui sont sous sa responsabilité redémarrent également).

D'autres approches incluent Aweya et al. (2006) (paquets de taille fixe sans exploitation de données additionnelles dédiées à la synchronisation), Khlifi & Grégroire (2006) (qui adresse explicitement des systèmes à faible exigence de précision) ainsi que Bi et al. (2006) et Wang et al. (2004).

Exception faite de l'approche de Duda et al. (1987), toutes les méthodes mentionnées ci-dessus se basent uniquement sur des mesures unidirectionnelles. L'utilisation de mesures bidirectionnelles, cependant, augmente significativement la robustesse de la méthode à un coût additionnel relativement faible (les mesures montantes peuvent être descendues au nœud esclave par batchs périodiques, éventuellement compressés). Également, à notre connaissance, aucune des méthodes susmentionnées n'a démontré une capacité à se conformer au niveau d'exigence extrême du standard GSM. Comme nous l'avons déjà indiqué, notre approche peut être considérée comme étant un raffinement de la méthode de Duda et al. (1987) qui exploite la robustesse induite par la disponibilité de mesures bidirectionnelles y compris lorsque le délai d'acheminement minimum des paquets est grand. De plus, notre méthode est simple, intuitive et implémentable en temps linéaire.

3.3 Une approche par programmation linéaire

3.3.1 Principe

Soit n et m respectivement le nombre de paquets descendants et montants. Soit $t_i^{(\mathrm{M})}$ et $t_i^{(\mathrm{S})}$ les dates d'émission et de réception du i-ème paquet descendant respectivement mesurées à l'aide des horloges maître et esclave. Soit $\tau_j^{(\mathrm{S})}$ et $\tau_j^{(\mathrm{M})}$ les dates d'émission et de réception du j-ème paquet montant respectivement mesurées à l'aide des horloges esclave et maître. Aussi, α et β dénotent respectivement le biais et le décalage de l'horloge esclave relativement à l'horloge maître, i.e. lorsque l'horloge maître indique t_0 l'horloge esclave indique $\alpha t_0 + \beta$.

Enfin, $D_i^{(d)} > 0$ et $D_j^{(u)} > 0$ sont deux variables aléatoires qui correspondent respectivement aux délais d'acheminement descendant et montant.

Comme illustré sur la figure 3.1(a), pour le i-ème paquet descendant, on a

$$t_i^{(\mathrm{S})} = \alpha t_i^{(\mathrm{M})} + \beta + \alpha D_i^{(d)}. \tag{3.1}$$

D'où,

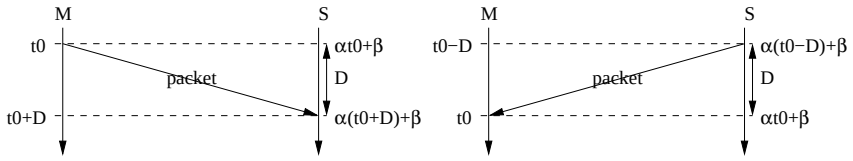
$$t_i^{(\mathrm{S})} \geq \alpha t_i^{(\mathrm{M})} + \beta. \tag{3.2}$$

Également, ainsi qu'illustré sur la figure 3.1(b), pour le j-ème paquet montant, on a

$$\tau_j^{(\mathrm{S})} = \alpha \tau_j^{(\mathrm{M})} + \beta - \alpha D_j^{(u)}. \tag{3.3}$$

D'où,

$$\tau_j^{(\mathrm{S})} \leq \alpha \tau_j^{(\mathrm{M})} + \beta. \tag{3.4}$$



(a) Flux descendant. (b) Flux montant.

FIGURE 3.1 – Illustration du principe d'estampillage.

En conséquence, la droite $t^{(\mathrm{S})} = \alpha t^{(\mathrm{M})} + \beta$ est coincée entre le nuage de points descendants obtenu en mettant $t_i^{(\mathrm{S})}$ en ordonnée et $t_i^{(\mathrm{M})}$ en abscisse (qui est au dessus) et le nuage de points montants qui s'obtient en mettant $\tau_i^{(\mathrm{S})}$ en ordonnée et $\tau_i^{(\mathrm{M})}$ en abscisse (nuage qui est en dessous de la droite). Par contre, ainsi qu'indiqué par Duda et al. (1987), une approche qui chercherait une unique droite, soit par une forme de régression linéaire ou soit par séparation des deux nuages de points, risque de ne fournir qu'une précision médiocre dans le cas d'un délai minimum d'acheminement des paquets non nul, cas de figure qui est plus la règle que l'exception. Du coup, plutôt que de chercher une unique droite, notre approche consiste à insérer un "couloir" aussi large que possible entre les deux nuages de points, c'est-à-dire deux droites parallèles aussi éloignées que possible et séparant les deux nuages. Voir la figure 3.2.

3.3.2 Formulation linéaire

Clairement, les inégalités (3.2) et (3.4) peuvent être interprétées comme les contraintes d'un programme linéaire.

Soit $t^{(\mathrm{S})} = \alpha t^{(\mathrm{M})} + \beta_1$ et $t^{(\mathrm{S})} = \alpha t^{(\mathrm{M})} + \beta_2$ les frontières haute et basse du couloir recherché, respectivement. Une estimation de α peut alors être obtenue

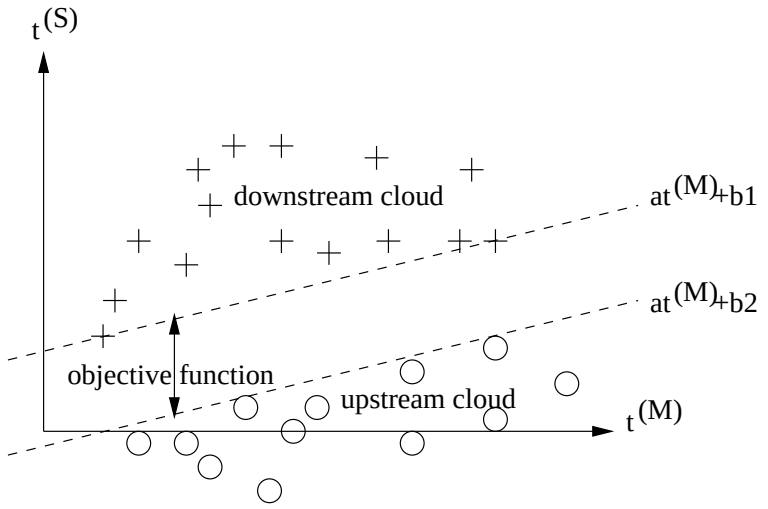


FIGURE 3.2 – Illustration du principe de fonctionnement de notre méthode de synchronisation.

par résolution du programme linéaire suivant :

$$\begin{cases} \text{Maximiser } \beta_1 - \beta_2 \\ \text{s. l. c.} \\ \alpha t_i^{(\mathrm{M})} + \beta_1 \leq t_i^{(\mathrm{S})},\ i \in \{1, \ldots, n\}, \\ \alpha \tau_j^{(\mathrm{M})} + \beta_2 \geq \tau_j^{(\mathrm{S})},\ j \in \{1, \ldots, m\}. \end{cases}$$

Par construction, ce programme ne possède que trois inconnues et il est bien connu que le problème de la programmation linéaire dans $\mathbb{R}^2$ et $\mathbb{R}^3$ peut être résolu en temps linéaire du nombre de contraintes (Megiddo, 1983 ; Dyers, 1983), ici $n+m$. Bien entendu, une implémentation sur étagère de l'algorithme du simplex, ou de tout autre algorithme pour la PL, peut également être utilisée si appropriée (ce qui est une question de génie logiciel, voir la discussion section 3.3.3).

Également, remarquons que $(\beta_1 + \beta_2)/2$ fournit un estimateur grossier du décalage de l'horloge esclave relativement à l'horloge maître. À l'évidence, la validité de cet estimateur dépend également du degré de symétrie du comportement du réseau.

3.3.3 Aspects systèmes

Au niveau système, la présente méthode requiert de la part du nœud esclave :

1. de collecter les données temporelles descendantes et montantes ;
2. de résoudre le programme linéaire de la section précédente ;
3. d'utiliser le biais estimé pour appliquer une correction bornée (par exemple limitée à 10 PPB par période) à son oscillateur local.

Ces étapes doivent être répétées en continu, avec une période de collecte des données suffisamment courte pour que l'hypothèse de constance du biais reste raisonnable (typiquement quelques minutes).

Également, dans la mesure où la dérive d'horloge est par nature un phénomène lent, il est raisonnable de partir du principe que l'on peut filtrer une estimation de biais qui serait trop différente de l'estimation obtenue sur la période précédente. En effet, bien que l'on peut escompter que la présente méthode soit robuste à certains événements réseaux comme des changements de chemins de routage, dans la mesure où l'on utilise des mesures bidirectionnelles, il ne peut être exclu que certaines conditions transitoires du réseau conduisent de temps à autres à des résultats aberrants.

Il convient aussi de souligner que la résolution du programme linéaire de la section 3.3.2 n'est pas une tâche critique sur le plan du temps d'exécution (typiquement, un budget de quelques secondes de calcul peut être alloué à cette tâche). En conséquence, le choix entre l'utilisation d'une implémentation sur étagère de l'algorithme du simplex et l'implémentation d'un algorithme spécialisé dans la résolution de programmes linéaires de faible dimension repose sur des considérations de génie logiciel : quantité de logiciel à embarquer (les solveurs génériques ayant tendance à être relativement lourds), temps de développement ("off-the-shelf" versus propriétaire), support et maintenance, etc.

3.4 Résultats expérimentaux

Cette section fournit des éléments empiriques permettant de conclure que notre approche par programmation linéaire peut se conformer aux exigences de précision du standard GSM dans deux situations typiques : une BTS connectée à son BSC via un WAN privé et une BTS (dite de classe de pico) connectée à son BSC via l'Internet public et une connexion ADSL.

3.4.1 Configuration WAN privé

Initialement, afin d'avoir une idée des conditions dans lesquelles notre méthode allait devoir opérer, nous avons procédé à de l'ordre de 10000 pings d'une station de travail située en banlieue de Londres depuis une machine située en région parisienne, ces deux machines faisant partie du même WAN privé. Pendant cette expérience, le RTD (Round Trip Delay) moyen était de 27.70 ms avec un écart-type de 11.19 ms et les RTD minimum et maximum étaient de 26 et de 196 ms, respectivement. Approximativement 0.01% des paquets ont été perdus.

Dans la mesure ou seul des délais unidirectionels nous intéressent, nous avons fait l'hypothèse que le réseau avait un comportement symétrique durant l'expérience et divisé les données de ping par 2. Ensuite, de manière à pouvoir faire des simulations réalistes, nous avons fait l'hypothèse d'un délai d'acheminement

distribué selon une loi de Weibull[6] dont nous avons estimé les paramètres de position, de forme et d'échelle[7].

Donc, pour la configuration WAN privé, nous avons simulé l'envoi d'un paquet toutes les 5 ms dans les deux directions et tiré les délais $D_i^{(d)}$ et $D_i^{(u)}$ (équations (3.1) et (3.3), respectivement) selon la loi de Weibull susmentionnée[8]. Nous cherchions ensuite à estimer un biais de +20 PPB soit, $\alpha = 1.000000020$.

La table 3.1 résume nos résultats expérimentaux. La colonne "Durée" indique la durée de la période de collecte des données, la colonne "# paquets" donne le nombre de paires de dates d'envoi et de réception qui ont été collectées et les colonnes "Moy.", "É.-t.", "Min" et "Max" donne des statistiques de base (en unités PPB) pour l'erreur d'estimation, i.e. $|\hat{\alpha} - \alpha|$, où $\hat{\alpha}$ dénote l'estimation obtenue par résolution du programme linéaire de la section 3.3.2. Les programmes linéaires ont été résolus avec l'implémentation du simplex de COIN-OR qui, sur le plan des performances, est capable de traiter des programmes avec de l'ordre de 250 000 contraintes en moins d'une seconde.

Durée	# paquets	Moy.	É.-t.	Min	Max
10 secs	2 000	0.06743	0.13902	0.00001	0.74213
1 min	12 000	0.02789	0.04952	2.7×10^{-6}	0.25863
10 mins	120 000	0.00451	0.01065	2×10^{-7}	0.08190

TABLE 3.1 – Statistiques de $|\hat{\alpha} - \alpha|$ en fonction de la durée, pour la configuration WAN privé. Les chiffres sont en PPB et ont été obtenus sur 100 simulations.

Les résultats donnés dans la table 3.1 suggèrent bien que, dans la configuration

6. On rappelle que la loi de Weibull est définie comme

$$f(x;\tau,a,b) = ab^{-a}(x-\tau)^{a-1}e^{-(\frac{x-\tau}{b})^a},$$

où τ, a et b sont respectivement les paramètres de position, de forme et d'échelle (Saporta, 1990). La loi de Weibull est communément utilisée pour modéliser le délai d'acheminement des paquets dans les réseaux paquets asynchrones (Norros, 1995 ; Papagiannaki et al., 2003).

7. On rappelle que pour la loi de Weibull on a $\mu = b\Gamma\left(1+\frac{1}{a}\right) + \tau$ (moyenne), $\sigma^2 = b^2\left(\Gamma\left(1+\frac{2}{a}\right) - \Gamma\left(1+\frac{1}{a}\right)^2\right)$ (variance) et $\gamma_1 = \frac{E[(X-\mu)^3]}{\sigma^3} = \frac{\Gamma\left(1+\frac{3}{a}\right)b^3 - 3(\mu-\tau)\sigma^2 - (\mu-\tau)^3}{\sigma^3}$ (skewness). On peut alors procéder à une estimation des paramètres de la loi par la méthode des moments. On a (par la moyenne) $\hat{\tau} = \bar{x} - \hat{b}\Gamma\left(1+\frac{1}{\hat{a}}\right)$ ainsi que (par la variance) $\hat{b} = \sqrt{\frac{s^2}{\Gamma\left(1+\frac{2}{\hat{a}}\right) - \Gamma\left(1+\frac{1}{\hat{a}}\right)^2}}$ et, enfin, par le skewness, l'équation

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3 = \Gamma\left(1+\frac{3}{\hat{a}}\right)\hat{b}^3 - 3(\hat{\mu}-\hat{\tau})\hat{\sigma}^2 - (\hat{\mu}-\hat{\tau})^3$$

où les valeurs chapeautées sont toutes fonctions de $\hat{a}$ e.g., $\hat{\mu} \equiv \hat{\mu}(\hat{a}, \hat{b}(\hat{a}), \hat{\tau}(\hat{a}, \hat{b}(\hat{a})))$, équation qui peut être résolue numériquement sans difficulté.

8. Notons qu'il aurait aussi tout à fait été possible d'utiliser des techniques de bootstraping.

WAN privé, notre méthode a le potentiel d'estimer des biais d'horloge avec une précision sub-PPB, dès lors que la durée d'observation est de taille raisonnable. Ces résultats sont discutés plus en détails à la section 3.5.

3.4.2 Configuration Internet public

Comme dans le cas de la configuration WAN privé, nous avons commencé par réaliser de l'ordre de 10000 pings d'une station de travail située au sein d'un WAN privé depuis une machine connectée à l'Internet public via une connexion ADSL. Durant cette expérience, le RTD moyen était de 64.06 ms avec un écart-type de 28.04 ms et les RTD minimum et maximum étaient de 55 et 849 ms, respectivement. De l'ordre de 0.33% des paquets ont été perdus.

De nouveau, comme dans la section précédente, les données de ping ont été divisées par 2 et ont servi à l'estimation des paramètres de position, de forme et d'échelle d'une loi de Weibull.

En conséquence, pour la configuration Internet public, nous avons simulé l'envoi d'un paquet toutes les 20 ms dans les deux directions où $D_i^{(d)}$ et $D_i^{(u)}$ (équations (3.1) et (3.3), respectivement) étaient tirés selon la loi de Weibull susmentionnée. Nous avons alors cherché à retrouver un biais de +40 PPB soit, $\alpha = 1.000000040$. Rappelons que l'exigence des ±50 PPB est relâchée à ±100 PPB pour les BTS de classe pico (3GPP, 2001).

La table 3.2 résume les résultats obtenus (se référer à la section précédente pour une description des différentes colonnes de cette table).

Durée	# paquets	Moy.	É.-t.	Min	Max
10 secs	500	0.75066	1.13316	0.00029	6.72219
1 min	3 000	0.04291	0.05569	0.00002	0.27037
10 mins	30 000	0.01065	0.01640	3.8×10^{-6}	0.08255

TABLE 3.2 – Statistiques de $|\hat{\alpha} - \alpha|$ en fonction de la durée, pour la configuration Internet public. Les chiffres sont en PPB et ont été obtenus sur 100 simulations.

3.5 Discussion

Les résultats expérimentaux donnés dans les deux sections précédentes suggèrent donc que notre approche fournit un niveau de précision suffisant dès lors que la période de collecte des données est supérieure à la minute (en résumé, une erreur pire cas d'approximativement 0.3 PPB a été observée pour les configurations WAN privé, pour un biais de 20 PPB, et Internet public, pour un biais de 40 PPB). Il suit qu'une période de collecte entre 1 et 10 mins paraît raisonnable aussi bien en termes de précision que de validité de l'hypothèse de constance du biais.

À notre connaissance, ces caractéristiques expérimentales de la méthode ont par la suite été confirmées en laboratoire à l'aide de simulateurs de trafic réseau, puis en situation réelle.

À noter également que la présente approche à été adaptée et étendue par Poirier et al. (2010) au problème de synchronisation hors ligne des traces d'exécution d'un système réparti.

Deuxième partie

Compilation pour les architectures de processeurs massivement parallèles

Les travaux présentés dans cette seconde partie sont menés depuis novembre 2007 dans le cadre de mes activités de recherche et de responsable d'équipe de recherche au sein du Laboratoire Systèmes Temps Réel Embarqués de la Direction de la Recherche Technologique du Commissariat à l'Énergie Atomique, à Saclay.

Le chapitre 4 est un chapitre de contexte et reflète les travaux de l'ensemble des membres de l'équipe "Calcul intensif embarqué" du LaSTRE. Nous y présentons succinctement un langage de programmation parallèle adapté aux nouvelles architectures de processeurs massivement multicœurs pour l'embarqué, un modèle d'exécution parallèle pour de tels systèmes ainsi que l'architecture logicielle d'une chaîne de compilation permettant de faire le lien entre les deux.

Cette technologie de compilation est bien réelle : elle fait l'objet d'un travail de prototypage avancé, mobilisant une dizaine d'ingénieurs-chercheurs du CEA depuis 2008, et d'un transfert technologique vers une startup de l'industrie du semiconducteur. Il s'agit donc bien d'une technologie industrielle en devenir.

De nombreux problèmes difficiles d'optimisation discrète se présentent dans le cadre de la mise en œuvre d'une telle chaîne de compilation.

En conséquence, les autres chapitres de cette partie, les chapitres 5, 6 et 7, présentent des travaux en optimisation combinatoire qui ont donné lieu à des algorithmes de résolution intégrés à cette chaîne de compilation, ou qui ont vocation à l'être.

Chapitre 4

Compilation flot de données

4.1 Introduction

En guise de prélude à notre contribution à l'optimisation combinatoire dans le domaine des systèmes embarqués nous présentons succinctement, dans ce chapitre, l'approche de la programmation et de la compilation pour les architectures de processeurs massivement multicœurs que nous avons développée ces dernières années. Ces travaux, qui définissent le contexte principal des travaux en optimisation présentés dans cette partie, sont le fruit d'une étroite collaboration avec Vincent David ainsi qu'avec l'ensemble des membres de l'équipe "Calcul intensif embarqué" du LaSTRE et s'effectuent principalement dans le cadre d'un laboratoire commun entre le CEA et la société Kalray, startup créée en juillet 2008, dont l'objectif est de concevoir et développer une nouvelle génération de circuits parallèles programmables multiapplications pour l'embarqué. Le développement de cette technologie de compilation mobilise depuis 2008 une équipe d'une petite dizaine d'ingénieurs-chercheurs, l'équipe "Calcul intensif embarqué" susmentionnée, dont je me suis vu confié la responsabilité. Cette technologie est aujourd'hui à l'état de prototype avancé : il existe une première version complète d'une chaîne de compilation qui a passé son baptême du feu industriel sur des applications de référence complexes (notamment un encodeur H.264 complet) jusqu'à la construction d'un binaire et à son exécution sous l'égide d'un micro-noyau que nous avons aussi prototypé. Cette technologie a également d'ores et déjà fait l'objet d'une première vague de transfert industriel en 2011.

Confidentialité oblige, nous ne pouvons pas vraiment donner de détails sur l'architecture Kalray. Nous nous contenterons d'indiquer qu'il s'agit d'un microprocesseur comportant un certain nombre de clusters, qui sont des petits calculateurs parallèles MIMD composés de quelques cœurs de calcul et dotés d'une mémoire partagée, interconnectés par un réseau paquet asynchrone sur puce de topologie torique, borné en latence. Cette technologie permet d'intégrer de l'ordre de 200 à 1000 cœurs de calcul, fonctionnant à des fréquences relativement faibles,

au sein d'une unique puce avec un budget thermique très réduit. Cf. Kalray (2011) pour plus de détails.

Si ses bases ont été jetées en même temps que celles de l'informatique[1], le parallélisme est longtemps resté l'apanage des spécialistes des grands centres de recherche. Ce n'est que depuis la fin de ce que l'on appelle désormais l'ère Moore qu'il devient nécessaire de rendre le parallélisme, c'est-à-dire *l'écriture, la mise au point, l'exécution performante et maîtrisée et la maintenance de programmes parallèles*, accessibles au plus grand nombre. Ceci afin de permettre une exploitation viable des architectures de processeurs parallèles, dites multicœurs, qui sont désormais monnaie courante.

4.2 Un modèle de programmation parallèle

4.2.1 Modèles de calcul flot de données

Le principal écueil de la programmation parallèle est la synchronisation entre les tâches. Quiconque a déjà eu à mettre au point un programme parallèle non trivial le sait. Nos premières réflexions nous ont donc conduit à chercher à nous débarrasser de cette synchronisation ou tout au moins de son explicitation. Nous nous sommes donc tournés vers des modèles dit flot de données, introduits par Lee & Parks (1995). Tout comme les réseaux de processus de Kahn (1974), les réseaux flot de données exhibent une certaine forme de déterminisme : pour un jeu de données d'entrée fixé, les résultats des calculs effectués par le réseau sont indépendants des aléas d'exécution (par exemple liés à l'ordonnancement de l'exécution des tâches ou à l'indéterminisme inhérent aux temps d'exécution unitaires). Une propriété fondamentale lorsqu'il s'agit de faire du calcul parallèle !

Un réseau de processus flot de données consiste simplement en un ensemble de tâches qui communiquent via des canaux FIFO unidirectionnels non bornés[2], et uniquement via ses canaux. Dans ce modèle, toutes les synchronisations inter-tâches se font donc de manière implicite, par les données. Pour se fixer les idées, un tel réseau peut être représenté par un graphe orienté dont les sommets sont les tâches et dont les arcs sont les canaux de communication. Les tâches, quant à elles, sont modélisées par des machines à états dont les transitions spécifient l'effet de la tâche sur ses canaux d'entrée/sortie sur le plan des quantités de données consommées et produites, respectivement.

1. Un peu d'histoire : Colossus, le premier ordinateur, construit à Bletchley Park par l'ingénieur britannique Thomas Flowers pendant la seconde guerre mondiale, pour la cryptanalyse des systèmes de chiffrement Allemands de la série SZ40 (nom de code allié : "Tunny"), était un calculateur parallèle dans le sens le plus moderne du terme (Copeland, 2006).

2. Insistons sur le fait qu'il ne s'agit là que d'un modèle de calcul, c'est-à-dire d'un modèle abstrait qui ne présage en rien de l'implémentation sur la cible qui en est le produit de compilation. En particulier, les canaux de communication doivent in fine être implémentés par le biais de tampons mémoires dimensionnés statiquement.

Il existe tout un bestiaire de modèles flot de données selon les contraintes que l'on impose aux machines à état des tâches (Girault et al., 1999). Les plus connus sont le modèle synchrone (Lee & Messerschmitt, 1987), dans lequel les machines à état des tâches sont limitées à un état et une transition, et le modèle cyclostatique (Bilsen et al., 1996), dans lequel les machines à état des tâches correspondent à des circuits. Dans la mesure où ces deux modèles excluent toute forme de non déterminisme, ils sont décidables (notamment les deux propriétés fondamentales que sont l'absence d'interblocage et l'exécution en mémoire bornée sont décidables). Cette propriété de décidabilité se perd dès lors que l'on ajoute le moindre indéterminisme (typiquement la possibilité d'existence de tâches de type switch et select) à ces modèles (Buck, 1993), sans prendre de précautions sémantiques. Comme nous le verrons au chapitre 6, cela n'empêche pas l'analyse formelle de réseaux issus des modèles généraux, mais cette analyse devient nécessairement approchée (tout en restant bien évidemment sûre).

Dans la lignée des approches de Lee (1988) et de Gao et al. (1992), et comme nous le verrons à la section suivante, le modèle de calcul sous-jacent au langage que nous avons défini, ΣC, étend le modèle cyclostatique avec une forme d'indéterminisme qui se veut bénigne sur le plan de la décidabilité.

4.2.2 Le langage ΣC

Le modèle de calcul étant déterminé, encore faut-il lui associer un langage de programmation digne de ce nom. Nous avons donc défini le langage de programmation ΣC dont les principales caractéristiques sont d'avoir une puissance d'expression adaptée à une large gamme d'applications (essentiellement en traitement du signal et de l'image), de permettre la maîtrise d'applications complexes via une approche par composants, de permettre l'expression naturelle d'un parallélisme massif, à mettre en adéquation par compilation avec les ressources offertes par le système, de permettre l'expression de programmes proches du fonctionnement système (ce qui est intéressant pour la programmation de librairies spécialisées, ou en début de vie d'un produit) et, enfin, de correspondre à une extension "naturelle" du langage C de manière à éviter, autant que faire se peut, de dérouter les programmeurs expérimentés des domaines applicatifs visés. La philosophie générale est donc d'assumer le parallélisme en facilitant l'expression de ce dernier qui est ensuite retravaillé par compilation ou, plus exactement, mis en adéquation avec les ressources du système.

L'objet de cette section n'est pas de fournir une spécification détaillée du langage ΣC, pour cela nous renvoyons le lecteur vers des références spécialisées (Goubier et al., 2008a, 2011 ; Kalray, 2011). Il s'agit juste d'en illustrer les grandes lignes au travers d'un exemple simple : le calcul du laplacien d'une image à l'aide d'une décomposition en filtres séparables de l'opérateur laplacien d'une gaussienne (Cocquerez & Philipp, 1995). Cet opérateur consiste simplement à réaliser une convolution monodimensionnelle sur toutes les lignes de l'image avec,

d'une part, le masque

$$g^{(1)} = (-1\ -6\ -17\ -17\ 18\ 46\ 18\ -17\ -17\ -6\ -1)$$

et, d'autre part, le masque

$$g^{(2)} = (0\ 1\ 5\ 17\ 36\ 46\ 36\ 17\ 5\ 1\ 0),$$

puis à réaliser une convolution monodimensionnelle sur toutes les colonnes des deux images résultantes avec les masques $g^{(2)}$ pour la première et $g^{(1)}$ pour la seconde. La somme de ces deux dernières images donne le laplacien de l'image d'entrée dont les passages par zéros en indiquent les contours. Plus précisément, on commence par calculer

$$R^{(1)}_{i,j} = \sum_{k=1}^{11} I_{i,j+k-1} g^{(1)}_{11-k+1} \tag{4.1}$$

et

$$R^{(2)}_{i,j} = \sum_{k=1}^{11} I_{i,j+k-1} g^{(2)}_{11-k+1}, \tag{4.2}$$

où I dénote l'image d'entrée, puis

$$\Delta_{i,j} = \sum_{k=1}^{11} R^{(1)}_{i+k-1,j} g^{(2)}_{11-k+1} + \sum_{k=1}^{11} R^{(2)}_{i+k-1,j} g^{(1)}_{11-k+1}. \tag{4.3}$$

La figure 4.1 donne un exemple de réseau de processus permettant de réaliser cette fonction. Le réseau est décrit de gauche à droite. La tâche R représente une tâche de lecture de l'image d'entrée, la tâche S qui lui succède est un opérateur de $\mathsf{split}(W, H)$ dont la fonction est de répartir les données sur son canal d'entrée (en round-robin W données par W données) sur ses H canaux de sortie. Les tâches L réalisent les deux premières convolutions monodimensionnelles sur une ligne d'image (elles ont donc une entrée et deux sorties) i.e., les équations (4.1) et (4.2) à i fixé. Les deux tâches J qui leur succèdent sont des opérateurs de $\mathsf{join}(W, H)$ qui ont pour fonction d'insérer les données des H canaux d'entrées (en round-robin W données par W données) sur le canal de sortie. Les tâches S suivantes sont des opérateurs de $\mathsf{split}(1, W)$ qui vont permettre de décomposer les deux images de résultats intermédiaires en colonnes afin que les tâches C procèdent au traitement des colonnes (deux convolutions et une sommation) i.e., l'équation (4.3) à j fixé, et fournissent des résultats qui sont recomposés par la tâche J qui leur succède (opérateur de $\mathsf{join}(1, W)$) en une image résultat qui est écrite par la tâche W.

L'encadré 1 donne un exemple de code source ΣC qui implémente simultanément les deux premières convolutions monodimensionnelles sur une ligne

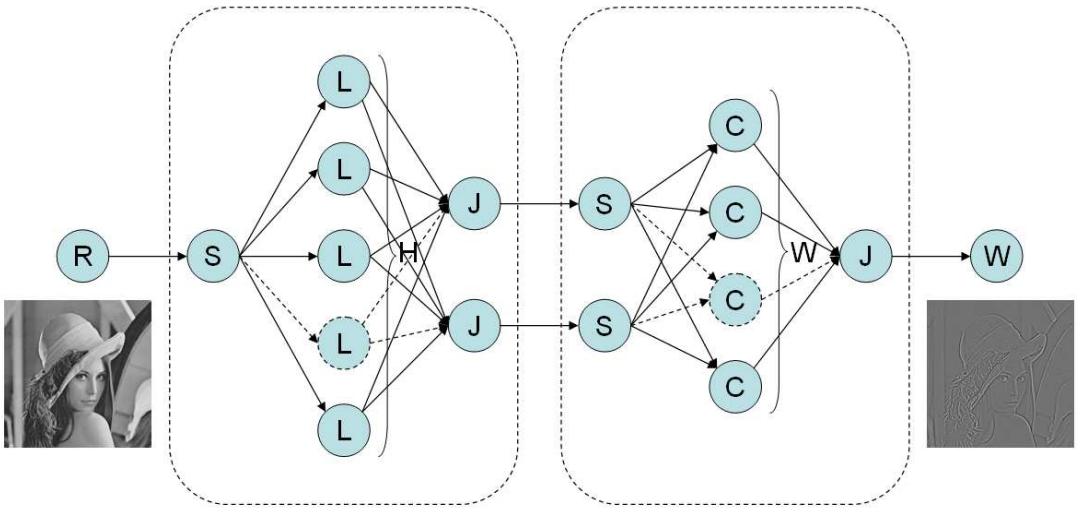


FIGURE 4.1 – Réseau de processus flot de données pour le calcul du laplacien d'une image.

d'image. Il s'agit d'un agent nommé `LineFilter` qui possède un paramètre d'instanciation `W` (la largeur de l'image) de type int ainsi qu'une interface indiquant qu'il est muni d'un port d'entrée (in) appelé `line` et sur lequel sont véhiculées des données de type int ainsi que de deux ports de sortie (out), `output1` et `output2` également typés sur int. L'interface de l'agent donne également la spécification (spec) de sa machine à état, cette dernière étant réduite à une transition consommant (respectivement produisant) `W` données sur le port `line` (respectivement sur les ports `output1` et `output2`). L'agent `LineFilter` possède deux tableaux de constantes (const) partagées (shared) entières `g1` et `g2` qui représentent les masques de convolution. Ces deux tableaux sont partagés au sens où contrairement aux données d'instances (l'agent `LineFilter` n'en a pas) accessibles en lecture/écriture ou aux constantes dépendantes de paramètres d'instanciation, elles n'ont pas besoin d'être dupliquées. Enfin, l'agent `LineFilter` possède une fonction de traitement start (dont la sémantique est d'être invoquée indéfiniment) dont l'effet sur les ports d'entrée/sortie de l'agent (spécifié par sa signature exchange) se résume à consommer (respectivement produire) `W` données sur le port `input` (respectivement sur les ports `output1` et `output2`), ces données étant accessibles depuis le code de traitement sous la forme d'un tableau d'entiers `a` (respectivement des tableaux d'entiers `b` et `c`). Une des caractéristiques sémantiques du langage est qu'il y a une équivalence pointeur stricte pour ces tableaux.

L'encadré 2 illustre comment combiner plusieurs instances de l'agent `LineFilter` au sein d'une entité composite (subgraph) afin de traiter l'intégralité d'une image pour ce qui est la première étape du calcul du laplacien. Cette entité possède un paramètre d'instanciation `W` (la largeur de l'image) de type int ainsi qu'une interface indiquant qu'elle est munie d'un port d'entrée (in) appelé `imag`

```
agent LineFilter(int W)
{
  interface
  {
    in<int> line;
    out<int> output1, output2;
    spec {{line[W]; output1[W]; output2[W]}};
  }
  shared const int
    g1[11] = { -1, -6, -17, -17, 18, 46, 18, -17, -17, -6, -1},
    g2[11] = {0, 1, 5, 17, 36, 46, 36, 17, 5, 1, 0};
  void start() exchange(line a[W], output1 b[W], output2 c[W])
  {
    int i,j;
    for(i=0;i<W;i++)
    {
      b[i] = 0; c[i] = 0;
      if(i<W-11)
        for(j=0;j<11;j++)
        {
          b[i] += g1[j] * a[i+j];
          c[i] += g2[j] * a[i+j];
        }
    }
  }
}
```

Encadré 1: Source ΣC d'un agent de traitement d'une ligne d'image.

et sur lequel sont véhiculées des données de type int ainsi que de deux ports de sortie (out), output1 et output2 aussi typés sur int. L'interface de l'entité donne également la spécification (spec) de sa machine à état, cette dernière étant réduite à une transition consommant (respectivement produisant) W*H (soit l'intégralité d'une image) données sur le port line (respectivement sur les ports output1 et output2). En tant qu'entité composite, LinePass n'a pas de fonction de traitement, il possède par contre une section map dont la vocation est d'instancier (new) et d'interconnecter (connect) ses composants (qui peuvent eux-mêmes être composites), certains étant des agents "built-ins" (les split, les join et autres dup, select et merge[3]).

```
subgraph LinePass(int W,int H)
{
  interface
  {
    in<int> imag;
    out<int> output1,output2;
    spec {{imag[W*H]; output1[W*H];output2[W*H]}};
  }
  map
  {
    int i;
    agent mysplit=new split<int>(H,W);
    agent myjoin1=new join<int>(H,W);
    agent myjoin2=new join<int>(H,W);
    for(i=0;i<H;i++)
    {
      agent filter=new LineFilter(W);
      connect(mysplit.output[i],filter.line);
      connect(filter.output1,myjoin1.input[i]);
      connect(filter.output2,myjoin2.input[i]);
    }
    connect(imag,mysplit.input);
    connect(myjoin1.output,output1);
    connect(myjoin2.output,output2);
  }
}
```

Encadré 2: Source ΣC d'un composant calculant la première passe du calcul du laplacien d'une image.

3. Le select étant doté d'une sémantique régularisée, proche de celle du dup (Goubier et al., 2011), ce qui permet d'éviter l'écueil de l'indécidabilité évoqué à la section 4.2.1.

Ce petit exemple suffit à illustrer quelques unes des caractéristiques déjà évoquées du langage. En particulier, nous avons vu qu'il est facile d'exprimer un degré de parallélisme important (ici pas moins de 640 tâches pour la première passe du traitement et 480 pour la seconde, dans le cas d'une image 640×480 typique d'une caméra de surveillance lambda), de fait trop important car le nombre de chaînes parallèles requis pour tenir un débit de 30 images par seconde (là encore typique) sur un tel traitement est en fait beaucoup plus faible (d'un facteur entre 10 et 100).

4.3 Cadencement par un temps logique

Dans cette section, nous présentons un mécanisme de cadencement basé sur un temps logique vectoriel permettant une exécution fonctionnellement déterministe, performante et maîtrisée d'applications flot de données sur une architecture parallèle.

La mise en œuvre d'un tel cadencement se base sur la construction d'un poset de vecteurs d'horloge isomorphe à un ordre partiel calculé hors ligne qui encode les dépendances de données (ou autres) entre les occurrences des tâches de l'application. D'une part, un tel procédé de cadencement permet un fonctionnement système performant pouvant, qui plus est, être accéléré moyennant la présence de quelques briques matérielles spécifiques (par exemple un opérateur de comparaison entre deux vecteurs temps). D'autre part, il définit un cadre suffisamment riche pour permettre à une chaîne de compilation d'encoder les prérequis d'exécution des tâches (en particulier les disponibilités de données et d'espace de production dans les tampons associés aux canaux du réseau de processus) dans un graphe de dépendances et de contraindre, par génération de code, le fonctionnement système à s'y conformer. Ceci, afin de garantir un fonctionnement correct, dénué de blocages et dans des bornes de dimensionnement calculées hors ligne.

Ce modèle d'exécution fait l'objet de deux brevets, l'un portant sur le procédé d'exécution (Sirdey & David, 2010b) et l'autre portant sur un circuit d'accélération matérielle associé (Sirdey & David, 2010a). Cf. également (Sirdey & David, 2008).

4.3.1 Rappels sur la théorie des ensembles ordonnés

Afin de formaliser la notion de temps logique, il convient de rappeler quelques éléments de théorie des ensembles ordonnés (Davey & Priestley, 2002).

Soit X un ensemble. Un ordre partiel sur X est une relation, notée $\sqsubseteq$, réflexive ($x \sqsubseteq x$, $\forall x \in X$), antisymétrique ($x \sqsubseteq y \wedge y \sqsubseteq x \Leftrightarrow x = y$, $\forall x, y \in X$) et transitive ($x \sqsubseteq y \wedge y \sqsubseteq z \Rightarrow x \sqsubseteq z$, $\forall x, y, z \in X$). Le couple $(X, \sqsubseteq)$ définit un ensemble partiellement ordonné (poset). De même, on définit la relation $\sqsubset$ telle

que $x \sqsubset y \Leftrightarrow x \sqsubseteq y \wedge x \neq y$, $\forall x, y \in X$. Enfin, lorsque $x \not\sqsubseteq y$ et $y \not\sqsubseteq x$, on dit que x et y sont incomparables, ce qui se note $x \| y$.

Un poset $(X, \sqsubseteq)$ définit une chaîne si $\forall x, y \in X$ soit $x \sqsubseteq y$, soit $y \sqsubseteq x$. Autrement dit, une chaîne est un poset sans couple d'éléments distincts incomparables. Inversement, un poset $(X, \sqsubseteq)$ définit une antichaîne lorsque $x \sqsubseteq y$ si et seulement si $x = y$.

Une extension linéaire d'un poset $(X, \sqsubseteq)$ est une chaîne $(X, \sqsubseteq')$ telle que $x \sqsubseteq y \Rightarrow x \sqsubseteq' y$, $\forall x, y \in X$. L'intersection $(X, \sqsubseteq_1) \cap (X, \sqsubseteq_2) = (X, \sqsubseteq_3)$ de deux posets $(X, \sqsubseteq_1)$ et $(X, \sqsubseteq_2)$ est telle que, $\forall x, y \in X$, $x \sqsubseteq_3 y$ si et seulement si $x \sqsubseteq_1 y$ et $x \sqsubseteq_2 y$. On dit qu'un ensemble $\{(X, \sqsubseteq_1), ..., (X, \sqsubseteq_n)\}$ d'extensions linéaires du poset $(X, \sqsubseteq)$ le réalisent si $\bigcap_{i=1}^{n} (X, \sqsubseteq_i) = (X, \sqsubseteq)$.

La dimension d'un poset P, notée $\dim(P)$, est définie comme le plus petit entier n tel qu'il existe un ensemble de n extensions linéaires de P qui le réalisent. Précisons également qu'étant donné un poset P et un entier $K > 2$, la question "$\dim(P) \leq K$?" est NP-complète [4].

4.3.2 Encodage vectoriel d'un ordre partiel d'exécution

Nous rappelons ici un mécanisme bien connu (Raynal, 1991 ; Raynal & Singhal, 1996) qui permet à chaque site d'un système réparti asynchrone de se construire une approximation d'un temps global correcte au sens où elle permet de conclure sur la dépendance ou l'indépendance causale entre deux événements.

Le principe est simple.

Chaque site i d'un système réparti à n sites est doté d'un vecteur horloge $h^{(i)}$ de dimension n, initialisé à 0. La composante $h_j^{(i)}$ de ce vecteur représente la connaissance qu'a le site i du nombre d'événements produits par le site j. Cette connaissance est acquise au fur et à mesure que le site i reçoit des messages. La progression du temps est alors dictée par les règles suivantes :

1. lorsque le site i produit un événement (interne, émission, réception), il incrémente $h_i^{(i)}$, son compteur d'événements, de 1 ;
2. à tout message est associé la valeur du vecteur horloge de l'émetteur, h, au moment de l'émission, à la réception d'un tel message le récepteur, i, incrémente $h_i^{(i)}$ et ajuste son vecteur horloge :

$$\forall j, h_j^{(i)} := \max(h_j, h_j^{(i)}).$$

Ces règles garantissent, d'une part, que $h_j^{(i)}$ croît de manière monotone, $\forall i, j$, et, d'autre part, que $h_j^{(i)} \leq h_j^{(j)}$, $\forall i, j$.

Si l'on dote l'ensemble des vecteurs horloge de la relation :

$$h \sqsubseteq h' \Leftrightarrow \forall i, h_i \leq h'_i,$$

4. Il s'agit d'ailleurs de l'un des problèmes ouverts du traité bien connu de Garey & Johnson (1979) : Partial Order Dimension (Yannakakis, 1982).

alors le poset des vecteurs horloge est isomorphe à la relation de causalité i.e., si a et b sont deux événements respectivement estampillés par les vecteurs h_a et h_b alors

$$a \rightarrow b \Leftrightarrow h_a \sqsubset h_b$$

et

$$a \text{ indépendant de } b \Leftrightarrow h_a \| h_b.$$

Dans les termes de la section 4.3.1, chacune des n composantes des vecteurs horloge définissent une extension linéaire de la relation de causalité et ces n extensions linéaires la réalisent.

En utilisant un principe analogue, il devient élémentaire de construire un ensemble de n extensions linéaires qui réalisent un ordre partiel entre les occurrences des n tâches d'une application parallèle donné par un graphe de dépendances sans circuit, borné ou non : le graphe est parcouru selon l'ordre topologique et chaque arc du graphe est assimilé à un message dûment estampillé.

D'une manière équivalente, cela signifie que l'on construit une représentation d'un graphe de dépendances entre les occurrences de n tâches à l'aide de vecteurs de dimension n. Une dimension de n n'est pas forcément minimum, nous avons néanmoins rappelé dans la section 4.3.1 que minimiser la dimension est NP-difficile et il a été montré (Charron-Bost, 1991) qu'il est parfois impossible de faire mieux. Néanmoins, ce pire cas n'est pas représentatif et il est généralement possible de faire beaucoup mieux : un calcul réparti typique mettant en jeu jusqu'à 300 processus ne dépasse généralement pas la dimension 10 (Ward, 1999). Sachant qu'en pratique on peut toujours relâcher la contrainte d'isomorphisme et contraindre la dimension à ne pas dépasser une borne donnée en réduisant (avec précaution) le parallélisme.

Pour se fixer les idées, la figure 4.2 illustre l'ordre partiel d'exécution et la numérotation vectorielle associée pour une application élémentaire composée de trois tâches à consommation et production unitaires, connectées en pipeline et communiquant à l'aide de tampons de taille 3. Les composantes des vecteurs temps ont été repliées en utilisant un ordre partiel modulo M sur $\{0, \ldots, M-1\}$ [5] afin de borner le nombre de bits utilisé pour les composantes des vecteurs d'horloge logique.

4.3.3 Illustration du fonctionnement système

Soit T l'ensemble des tâches de l'application. À chaque tâche $t \in T$ on associe :

1. un vecteur de temps logique, $d^{(t)}$, de dimension $n \leq |T|$ (cette dimension étant la même pour toutes les tâches) qui représente sa date logique de disponibilité, de même, on suppose donnée $d_0^{(t)}$, la date logique de disponibilité initiale de la tache t ;

5. Soit T et M deux entiers tels que $M > 2T$, on dote l'ensemble $\{0, \ldots, M-1\}$ de la relation $x \sqsubset y =$ vrai si et seulement si $x < y \wedge y - x \leq T \vee x > y \wedge M - x + y \leq T$.

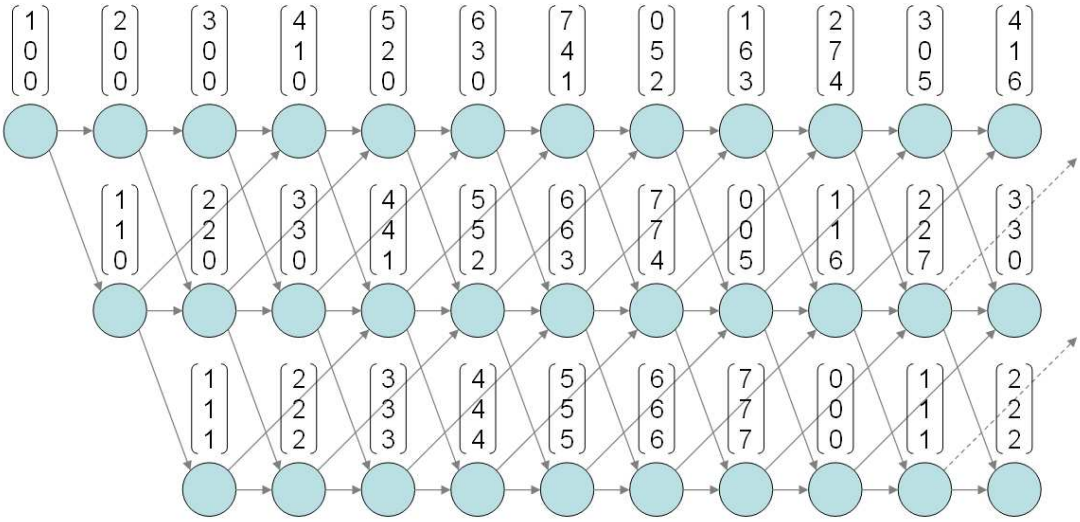


FIGURE 4.2 – Ordre partiel d'exécution et numérotation vectorielle (repliée) associée pour un pipeline à trois étages.

2. un entier, $k^{(t)}$, qui représente un compteur de dépendances ;
3. une règle d'incrémentation du temps logique, $\delta^{(t)} : \mathbb{Z}^n \longrightarrow \mathbb{Z}^n$, qui permet, à la fin de l'exécution de la tâche t, de calculer la nouvelle date logique de disponibilité pour la tâche.

Une tâche dont le compteur de dépendances est égal à 0 peut être exécutée. Au démarrage du système les actions suivantes sont réalisées :

1. pour chaque tâche $t \in T$ faire $d^{(t)} := d_0^{(t)}$ et $k^{(t)} := 0$;
2. pour chaque tâche $t \in T$ faire, pour chaque tâche $t' \in T \setminus \{t\}$ faire si $d^{(t')} \sqsubset d^{(t)}$ alors $k^{(t)} := k^{(t)} + 1$;

Dans la mesure où les vecteurs de temps logique encodent un ordre partiel, il y a nécessairement une tâche dont le compteur de dépendances est égal à 0 et l'exécution peut commencer (d'une manière équivalente, il y a toujours un sommet sans prédécesseur dans un graphe sans circuit, borné ou non).

Fonctionnellement, lorsque l'occurrence d'une tâche t se termine, les actions suivantes sont réalisées :

1. pour chaque tâche $t' \in T \setminus \{t\}$ faire si $d^{(t)} \sqsubset d^{(t')}$ alors $k^{(t')} := k^{(t')} - 1$;
2. faire $d^{(t)} := \delta^{(t)}(d^{(t)})$;
3. pour chaque tâche $t' \in T \setminus \{t\}$ faire si $d^{(t)} \sqsubset d^{(t')}$ alors $k^{(t')} := k^{(t')} + 1$;
4. pour chaque tâche $t' \in T \setminus \{t\}$ faire si $d^{(t')} \sqsubset d^{(t)}$ alors $k^{(t)} := k^{(t)} + 1$.

En termes de mise en œuvre, il convient de souligner, d'une part, qu'il y a matière à parallélisme dans les actions ci-dessus et, d'autre part, qu'il est possible de ne travailler que sur des sous-ensembles de tâches (calculés hors ligne) qui

excluent, par exemple, les tâches t' qui sont telles que $d^{(t)}$ et $d^{(t')}$ sont toujours incomparables ou encore en se limitant arbitrairement à un horizon borné.

4.4 Processus de compilation

Cette section donne un bref aperçu de l'architecture logicielle d'une chaîne de compilation pour le langage ΣC, dont l'objet est in fine de faire le lien entre une application écrite dans ce langage et le modèle d'exécution présenté à la section précédente pour aboutir à la construction d'un binaire exécutable.

4.4.1 Parseur et génération de code

La première passe de compilation d'une application ΣC commence classiquement par une étape de preprocessing (au sens du préprocesseur C) suivie des étapes d'analyse lexicale, syntaxique et sémantique (par exemple en termes de typage) que l'on trouve dans tous les compilateurs. La fonction de cette première passe consiste ensuite à générer des codes C à partir des sources ΣC. Ces codes sont répartis en deux catégories : les codes d'instanciation du réseau de processus, issus de la compilation des sections map que nous avons introduites à la section 4.2.2 et qui ont vocation à être exécutés hors-ligne dans le cadre du processus de compilation ainsi que les codes de traitement, issus de la compilation des fonctions de traitement des agents qui, eux, ont vocation à être embarqués.

Il est important de noter que ces codes sont générés une fois pour toutes et qu'ils ne sont plus destinés à être modifiés autrement que par substitution, c'est-à-dire à ne pas être manipulés par autre chose qu'un préprocesseur C.

4.4.2 Compilation du parallélisme

La deuxième passe de compilation a pour vocation première l'exécution des codes d'instanciation générés par la première passe. Comme nous l'avons vu à la section 4.2.2 aucune restriction particulière ne s'applique au code présent dans les sections map (les topologies d'interconnexions peuvent donc être quelconques, les valeurs des paramètres d'instanciation des agents peuvent être le résultat de calculs arbitrairement complexes, etc.). Une approche pragmatique est donc de lui faire correspondre un code C faisant appel à des API adaptées qui est ensuite compilé et exécuté sur la station de compilation afin de construire une structure de données représentant le réseau de processus (graphe du réseau de processus et machines à états des instances d'agents), d'une part, et d'autre part de générer les données propres aux instances des agents (valeurs des paramètres d'instanciation, valeurs d'initialisation pour les variables d'état, etc.).

Cette passe de compilation est aussi responsable d'un certain nombre de manipulations sur le réseau de processus. En particulier, il lui incombe de procéder

à une première réduction du parallélisme (par réduction des schémas d'accès, par exemple le nombre de sorties d'un split, et fusion de tâches) afin de mettre ce dernier en adéquation avec les ressources du système et, également, de composer les schémas d'accès aux données exprimés à l'aide des agents "built-ins" (split, join, dup, etc.) afin, autant que faire se peut, de permettre leurs implantations à l'aide de tampons mémoires partagés. C'est aussi à cette étape, que l'on peut procéder à une vérification de cohérence hiérarchique des spécifications (il s'agit, pour chaque agent composite, de vérifier que la composition des machines à états de ces composants implémente bien sa machine à états) ainsi qu'à une vérification de l'absence d'interblocage accompagnée d'un prédimensionnement sûr des tampons associés aux canaux tel que présenté au chapitre 6.

Dès la fin de cette phase, il est également possible de procéder à une première exécution du réseau de processus à l'aide d'un moteur d'exécution littérale basé sur des primitives Posix (par exemple des processus indépendants communiquant par pipes, par segments de mémoire partagée ou par sockets TCP). Un tel environnement d'exécution permet assez rapidement de procéder à une mise au point fonctionnelle du code applicatif, indépendamment de l'architecture cible.

4.4.3 Affectation de ressources

La troisième passe de compilation est en charge du lien avec la structure de l'architecture cible en procédant à l'affectation de ses ressources (au sens large).

Il s'agit donc, dans un premier temps, de procéder à un dimensionnement des tampons de communication tenant compte des caractéristiques temporelles des tâches (les temps d'exécution) et des objectifs de débit de l'application (contraintes non fonctionnelles). Pour ce faire, nous avons développé une approche à l'ordre 2 qui vise à trouver des tailles de tampon permettant d'atténuer les effets des variances des temps d'exécution sur les débits moyens afin de ramener ces derniers, lorsque cela est possible évidemment, au dessus de valeurs seuils critiques calculées à partir des contraintes de l'application.

Dans un second temps, afin de faire le lien avec le modèle d'exécution présenté à la section 4.3, il convient de construire un ordre partiel non borné, replié (donc représentable de manière finie) sur les occurrences des tâches (Galea & Sirdey, 2010). Pour ce faire, on commence par calculer (par résolution d'un système linéaire), un vecteur de répétition canonique qui donne, pour chaque tâche, le nombre d'occurrences permettant de ramener le réseau d'un état initial vers le même état. Ce vecteur définit un cycle de fonctionnement qui peut, par définition, être répété indéfiniment et ce, également par définition, en mémoire bornée. Guidé par ce cycle de fonctionnement, on procède ensuite à une première exécution symbolique du réseau (au sens où l'on exécute les machines à états des tâches et où seules nous intéressent les quantités produites et consommées sur les canaux) afin de construire un ordre partiel reflétant les dépendances de données entre les occurrences des tâches, puis à une seconde exécution symbolique locale à chaque

couple producteur/consommateur afin d'ajouter à cet ordre partiel les dépendances (repliées) permettant de garantir le respect des bornes de dimensionnement des tampons.

Enfin, il s'agit de procéder à l'étape de placement/routage qui est présentée en détails au chapitre 5 et dont l'objet est schématiquement de grouper, sous les contraintes de capacité des clusters de l'architecture, les tâches qui communiquent ensemble puis d'affecter ces groupes de tâches aux clusters de l'architecture en tenant compte d'un critère de distance et, enfin, de calculer les chemins d'acheminement des données au travers du réseau sur puce. Bien évidemment, cette étape requiert la résolution de plusieurs problèmes difficiles d'optimisation discrète et nous renvoyons le lecteur au chapitre 5 de ce mémoire pour les détails, cf. également Sirdey & David (2009).

Pour donner des résultats de qualité, cette passe d'affectation des ressources nécessite une connaissance assez précise des caractéristiques de l'application, en particulier temporelles. De telles données sont généralement obtenues en intégrant à la chaîne une boucle itérative de rétroaction par le biais de laquelle des mesures réalisées à différents niveaux de simulation (simulation fonctionnelle, simulation TLM voire exécution sur cible) sont réinjectées dans les instances des problèmes d'optimisation à résoudre. Certains paramètres peuvent également être évalués, généralement de manière conservative, par analyse statique.

4.4.4 Génération de runtime et construction des binaires

L'objet de la quatrième et dernière passe de compilation d'une application ΣC est double. Il s'agit d'une part de procéder à la génération des données de paramétrisation du logiciel système (tables d'incréments vectoriels pour le modèle d'exécution, structures de données associées aux tampons de communication inter-tâche, données de configuration du réseau sur puce, etc.). Cf. Dubrulle & Sirdey (2009). D'autre part, il s'agit, en s'appuyant sur un backend de compilation C, de procéder à la construction d'un binaire chargeable sur la cible par le biais d'une édition des liens multipasse (Sirdey et al., 2009b) permettant d'assurer par construction que le seul moyen de communication inter-tâche se réduit bien aux canaux de communication. En particulier, un certain nombre de symboles doivent être dupliqués, par tâche, afin d'éviter la création de canaux cachés non conformes à la sémantique du langage ΣC.

La main peut ensuite être passée au logiciel système dont le rôle est d'animer l'exécution de l'application sur la structure.

Chapitre 5

Placement et routage de réseaux de processus

5.1 Introduction

Dans ce chapitre, nous présentons une approche des problèmes de partitionnement, de placement et de routage de réseaux de processus (y compris de grande taille) adaptée à des architectures parallèles clusterisées du type de celle évoquée au chapitre 4.

Ces travaux ont été réalisés en collaboration avec Pascal Aubry, Vincent David et François Galéa ainsi qu'avec Jacques Carlier, Dritan Nace et Oana Stan pour les aspects stochastiques (section 5.3). Également, les algorithmes présentés dans ce chapitre, exception faite de l'algorithme de partitionnement stochastique de la section 5.3, ont tous fait partie de la première vague de transfert industriel de la technologie de compilation du langage ΣC (cf. chapitre 4).

En première approche, nous avons décomposé le problème en trois phases. Nous reviendrons sur les conséquences de ce choix à la fin de ce chapitre.

5.2 Partitionnement de réseaux de processus

5.2.1 Énoncé et complexité du problème

Soit $G = (V, A)$ un graphe orienté dont les sommets (respectivement les arcs) sont les tâches (respectivement les canaux) d'un réseau de processus à instancier sur une architecture parallèle à N nœuds. Soit R un ensemble de ressources (essentiellement des tailles d'empreinte mémoire et des taux d'occupation de ressources de calcul). Sont également donnés une fonction de taille $s : V \longrightarrow \mathbb{R}^{+|R|}$, une fonction d'affinité $q : A \longrightarrow \mathbb{R}^{+}$ et un vecteur de capacité nœud $C \in \mathbb{R}^{+|R|}$ (pour simplifier on suppose ici, modulo une légère perte de généralité, que les nœuds sont homogènes).

Pour alléger, on utilisera la notation $Q_{vw} = q((v,w))$ si $(v,w) \in A$, 0 sinon (pour tout $(v,w) \in V^2$). Également, on posera $S_{vr} = s(v)_r$, $r \in R$.

Le problème de partitionnement que nous devons résoudre consiste à trouver une affectation des sommets aux nœuds, $f : V \longrightarrow N$, qui vérifie les contraintes de capacité,

$$\sum_{v \in V : f(v) = n} S_{vr} \leq C_r, \forall n \in N, \forall r \in R, \tag{5.1}$$

et qui minimise la fonction économique :

$$\sum_{(v,w) \in A : f(v) \neq f(w)} Q_{vw}.$$

Dans sa variante mono-ressource le problème ci-dessus est connu sous le nom de Node Capacitated Graph Partitioning Problem dans la littérature (Ferreira et al., 1996). Il s'agit sans surprise d'un problème NP-difficile au sens fort.

À noter, également, que le problème qui ne consiste qu'à trouver une affectation quelconque des sommets aux nœuds qui satisfait les contraintes de capacité est déjà un problème de décision NP-complet (apparenté aux problèmes de bin-packing et de sac à dos multiple).

5.2.2 État de l'art et positionnement

Ces problèmes de partitionnement ont déjà été bien étudiés. Ceci bien que les modèles soient rarement tout à fait équivalents dans leur façon de contraindre à ne pas grouper tous les sommets du graphe dans une unique partition (Hendrickson & Kolda, 2000 ; Bichot & Siarry, 2010).

Sur le plan de la résolution approchée on trouve des algorithmes par améliorations successives tels l'heuristique de Kernighan & Lin (1970), le recuit simulé (Johnson et al., 1989) et des algorithmes par construction progressive (David et al., 1991). Cf. également le traité récent de Bichot & Siarry (2010). Ces algorithmes permettent d'obtenir des solutions pour des instances de grandes tailles, sans toutefois, par définition, fournir de garantie d'optimalité.

Sur le plan de la résolution exacte, le problème a essentiellement été étudié sous l'angle de l'approche polyèdrale (Ferreira et al., 1996, 1998). Les meilleurs algorithmes connus, de type branch-and-cut, peuvent résoudre en quelques heures des instances avec des graphes de l'ordre de 300 sommets et 500 arcs.

Ces résultats, tout à fait honorables, ne sont pas suffisants dans le cadre de notre contexte applicatif où l'on doit prévoir d'avoir à partitionner des graphes à quelques milliers de sommets (disons jusqu'à de l'ordre de 4000) en quelques dizaines de partition (disons entre 16 et 64).

Notons, enfin, des travaux intéressants sur l'utilisation de la programmation semidéfinie pour obtenir des bornes inférieures d'excellente qualité (Lisser & Rendl, 2003).

Comme nous venons de le voir, la taille prévisible des instances à résoudre dans le cadre d'une chaîne de compilation flot de données les rend inaccessibles aux meilleures méthodes de résolution exactes (qui doivent par ailleurs être adaptées au contexte multiressource).

Il convient donc de se tourner vers des méthodes de résolution approchées.

La meilleure d'entre elles, de réputation, est l'heuristique de Kernighan-Lin, elle nécessite néanmoins un effort important d'adaptation car elle est initialement formulée pour le problème de bipartitionnement uniforme avec poids unitaires et les approches de généralisation au cas pondéré conduisent à des graphes de taille pseudopolynomiale.

Une approche par recuit simulé est séduisante, d'autant qu'il existe des schémas généraux de fixation des paramètres permettant de tenir compte d'une distance à l'optimum cible (Sirdey et al., 2009a), mais la conception d'une fonction de voisinage préservant la connexité forte du graphe qu'elle engendre, une condition nécessaire de convergence que l'on s'efforce généralement de respecter lorsque l'on conçoit un tel algorithme, sans perturber la fonction économique avec des termes de pénalités, s'avère délicate en raison de la structure du problème. Ceci sans compter les temps de calcul qui sont généralement élevés, pour ne pas dire prohibitifs en tout cas en début du cycle de développement [1], sur les instances de grandes tailles.

En conséquence, nous nous sommes tourné vers un algorithme par construction progressive. Cet algorithme, de nature gloutonne, est fondé sur la notion d'affinité relative introduite par David et al. (1991) qui s'avère expérimentalement très pertinente pour ce type de problèmes. L'utilisation algorithmique de cette notion est néanmoins très différente et adaptée au présent contexte, en particulier afin de tenir compte de la facette bin-packing du problème ainsi que de garantir que l'algorithme termine toujours avec un partitionnement réalisable.

5.2.3 Notion d'affinité relative

Soit S et T deux sous-ensembles disjoints de V.

L'affinité de S envers T est donnée par

$$\alpha(S,T) = \sum_{(v,w)\in\delta(S,T)} Q_{vw}.$$

Trivialement, on a $\alpha(S,T) = \alpha(T,S)$.

L'affinité totale de S (et de manière analogue pour T) est donnée par

$$\beta(S) = \alpha(S,\bar{S}).$$

1. Pour se fixer les idées, une implémentation séquentielle du recuit simulé en voisinage 2-OPT requiert un temps de calcul de l'ordre de 3 heures pour résoudre (de manière approchée) une instance du problème du voyageur de commerce de l'ordre de 4000 villes (instance `fl3795` de TSPLib). Or, dans le cas du TSP, la mise à jour de la fonction économique se fait en $O(1)$.

Enfin, l'affinité relative de S envers T est définie par

$$\gamma(S,T) = \frac{1}{2}\alpha(S,T)\left(\frac{1}{\beta(S)} + \frac{1}{\beta(T)}\right)$$

où $\frac{\alpha(S,T)}{\beta(S)}$ (par exemple) représente la contribution de l'ensemble des arcs incidents à S et à T à l'affinité totale de S.

L'intérêt d'utiliser cette notion d'affinité relative, plutôt que l'affinité, comme base pour une méthode par construction progressive est que cela permet d'éviter de réaliser trop tôt des choix déterminants, choix qui ne pourraient être remis en question par nature même de l'algorithme.

Illustrons ce principe sur un petit graphe non orienté à quatre sommets A, B, C et D, A étant relié à B par une arête de valeur 2, B à C par une arête de valeur 3 et C à D par une arête de coût 2, que l'on souhaite décomposer en deux partitions de capacité 2. Un partitionnement glouton basé sur l'affinité commencerait par apparier les sommets B et C donnant ainsi le partitionnement $\{A, D\}$ et $\{B, C\}$ de coût 4. Un partitionnement glouton basé sur l'affinité relative conduirait d'abord à apparier les sommets A et B (ou de manière équivalente C et D) dans la mesure où $\gamma(\{A\},\{B\}) = \gamma(\{C\},\{D\}) = 0.7$ et où $\gamma(\{B\},\{C\}) = 0.6$, donnant ainsi le partitionnement $\{A, B\}$ et $\{C, D\}$ de coût 3.

5.2.4 Un algorithme par construction progressive

Cette section fournit la spécification d'un algorithme de résolution approchée par construction progressive pour le Multi-resource Node Capacitated Graph Partitioning Problem.

Soit W l'ensemble des sommets non encore affectés à un nœud (initialement $W = V$).

À chaque itération de l'algorithme, on procède alors comme suit :

1. trouver, s'il en existe une, l'affectation admissible (v^*, n^*) ($v^* \in W$, $n^* \in N$) d'affinité relative,

$$\gamma_1 = \gamma(\{v^*\}, \{v \in V \setminus W : f(v) = n^*\}),$$

   maximum ;

2. trouver, s'il en existe une, la fusion admissible (n_1^*, n_2^*) ($n_1^* \in N$, $n_2^* \in N$) d'affinité relative,

$$\gamma_2 = \gamma(\{v \in V \setminus W : f(v) = n_1^*\}, \{v \in V \setminus W : f(v) = n_2^*\}),$$

   maximum ;

3. si $\gamma_1 \geq \gamma_2$ alors affecter v^* à n^*, sinon fusionner n_1^* et n_2^*.

L'algorithme s'arrête lorsqu'il n'a rien pu faire ce qui signifie soit que W est vide (auquel cas un partitionnement complet a été trouvé) soit qu'il n'y a plus ni affectation ni fusion admissible (auquel cas seul un partitionnement partiel a été trouvé).

Une affectation d'un sommet v à un nœud n est admissible si elle ne viole pas de contrainte de capacité c'est-à-dire si, pour tout $r \in R$,

$$S_{vr} + \sum_{w \in V \setminus W : f(w) = n} S_{wr} \leq C_r. \tag{5.2}$$

De même, une fusion entre un nœud n et un nœud m est admissible si, pour tout $r \in R$,

$$\sum_{v \in V \setminus W : f(v) = n} S_{vr} + \sum_{v \in V \setminus W : f(v) = m} S_{vr} \leq C_r. \tag{5.3}$$

À noter que le test dans la dernière étape de l'itération, $\gamma_1 \geq \gamma 2$, favorise les affectations par rapport aux fusions. Ceci est empiriquement justifié.

De même, afin de tenir compte de la facette bin-packing du problème, lorsque deux décisions sont équivalentes sur le plan de l'affinité relative, on donne la priorité aux affectations des sommets de plus grande taille sur les nœuds les moins chargés (cas de l'affectation) et aux fusions des nœuds les plus chargés entre eux (cas de la fusion).

Pour ce faire, il convient de généraliser ces notions au cas multi-ressource (Sirdey et al., 2003).

On dit alors qu'un sommet v est plus petit qu'un sommet w si

$$\max_{r \in R} \frac{S_{vr}}{C_r} < \max_{r \in R} \frac{S_{wr}}{C_r}$$

De même, on dit qu'un nœud n est plus chargé qu'un nœud m si

$$\max_{r \in R} \frac{1}{C_r} \left(C_r - \sum_{v \in V \setminus W : f(v) = n} S_{vr} \right) < \max_{r \in R} \frac{1}{C_r} \left(C_r - \sum_{v \in V \setminus W : f(v) = m} S_{vr} \right)$$

Dans le cas monoressource, ces définitions coïncident trivialement avec les définitions usuelles.

5.2.5 Diversification par randomisation

Sur certaines instances, il est bien connu qu'un algorithme par construction progressive peut se retrouver piégé par une mauvaise décision qu'il est incapable de remettre en cause ou de compenser. Paradoxalement ou pas, c'est souvent sur des instances de taille relativement modeste que ce phénomène se manifeste (Sirdey et al., 2010). Une solution pour remédier à ce problème consiste à exécuter

plusieurs fois une version randomisée de l'algorithme (on parle d'heuristique à démarrages multiples) et à ne conserver que la meilleure des solutions engendrées.

Dans le cas du présent algorithme, la prise en compte du critère de départage des égaux que nous avons vu à la section précédente conduit à prétrier la liste des sommets du graphe par taille décroissante (au sens multiressource). Une stratégie de randomisation simple consiste alors à exécuter une fois l'algorithme sur cet ordre, puis à recommencer plusieurs fois en réordonnant la liste des sommets aléatoirement (à l'image des algorithmes de liste utilisés en ordonnancement).

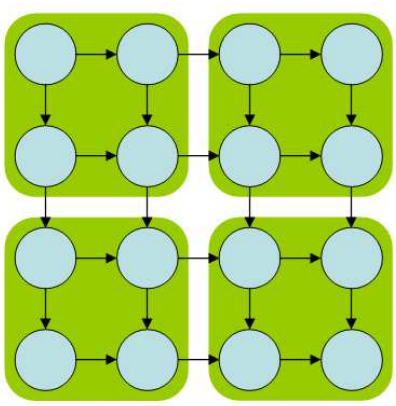
Une autre stratégie de randomisation est celle de Hart & Shogan (1987), souvent utilisée dans les algorithmes de type GRASP (Feo & Resende, 1995 ; Sirdey et al., 2010), qui consiste à choisir aléatoirement la décision prise à chaque étape de l'algorithme parmi les k meilleures décisions admissibles. Ce n'est néanmoins pas la solution que nous avons retenue en première approche.

5.2.6 Résultats expérimentaux

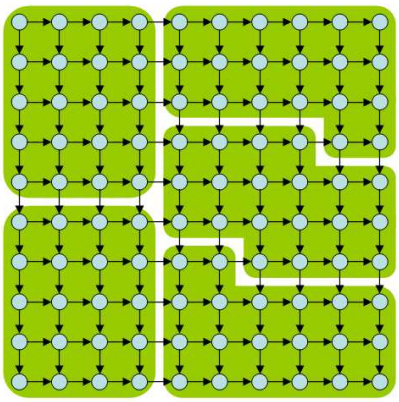
Les exemples ci-après ont été obtenus à l'aide d'une implémentation de l'algorithme présenté à la section précédente et ont vocation à illustrer les résultats obtenus sur des exemples simples. La figure 5.1(a) indique le partitionnement obtenu pour une grille 4x4 (16 sommets) sur 4 nœuds de capacité 4, dans le cas monoressource avec tailles et affinités unitaires. La valeur de la solution est 8 (optimale). La figure 5.1(b) indique le partitionnement obtenu pour une grille 10x10 (100 sommets) sur 5 nœuds de capacité 20, dans le cas monoressource avec tailles et affinités unitaires. La valeur de la solution est 28 (optimale). La figure 5.1(c) indique le partitionnement obtenu pour un arbre binaire de profondeur 5 (31 sommets) sur 4 nœuds de capacité 8, dans le cas monoressource avec tailles et affinités unitaires. La valeur de la solution est 4 (optimale).

Pour se donner une idée des temps de calcul mis en jeu sur des instances de taille importante, reflétant ce qu'une chaîne de compilation pourrait avoir à traiter, nous avons fait tourner une implémentation soignée (mais séquentielle) de l'algorithme sur les cas suivants. Pour une grille 23x23 (529 sommets) sur 16 nœuds de capacité 40, on obtient une partition de valeur 150 (1 itération) en 0.18 secs sur une station Linux lambda (cf. figure 5.2). Pour une grille 46x46 (2116 sommets) sur 64 nœuds de capacité 40, on obtient une partition de valeur 691 (1 itération) en 5.36 secs sur une station Linux lambda. Pour une grille 100x100 (10000 sommets) sur 200 nœuds de capacité 50, instances dont la taille est largement supérieure au pire cas qu'une chaîne de compilation flot de données devrait avoir à traiter, on obtient une partition de valeur 3101 (1 itération) en 3 mins 36 secs sur la même station Linux lambda.

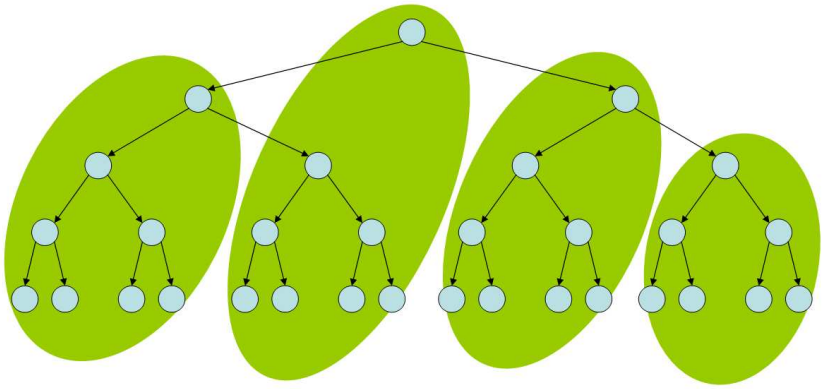
Nous avons évalué la présente heuristique sur le jeu d'instances de bipartitionnement de Johnson et al. (1989), les meilleures valeurs connues provenant de l'article de Boulle (2004), plus récent. Les résultats obtenus sont indiqués dans le tableau 5.1 (un "*" indique que la meilleure valeur connue est aussi connue



(a) Grille 4×4.



(b) Grille 10×10.



(c) Arbre binaire.

FIGURE 5.1 – Exemple de partitionnement de petits graphes (cf. texte).

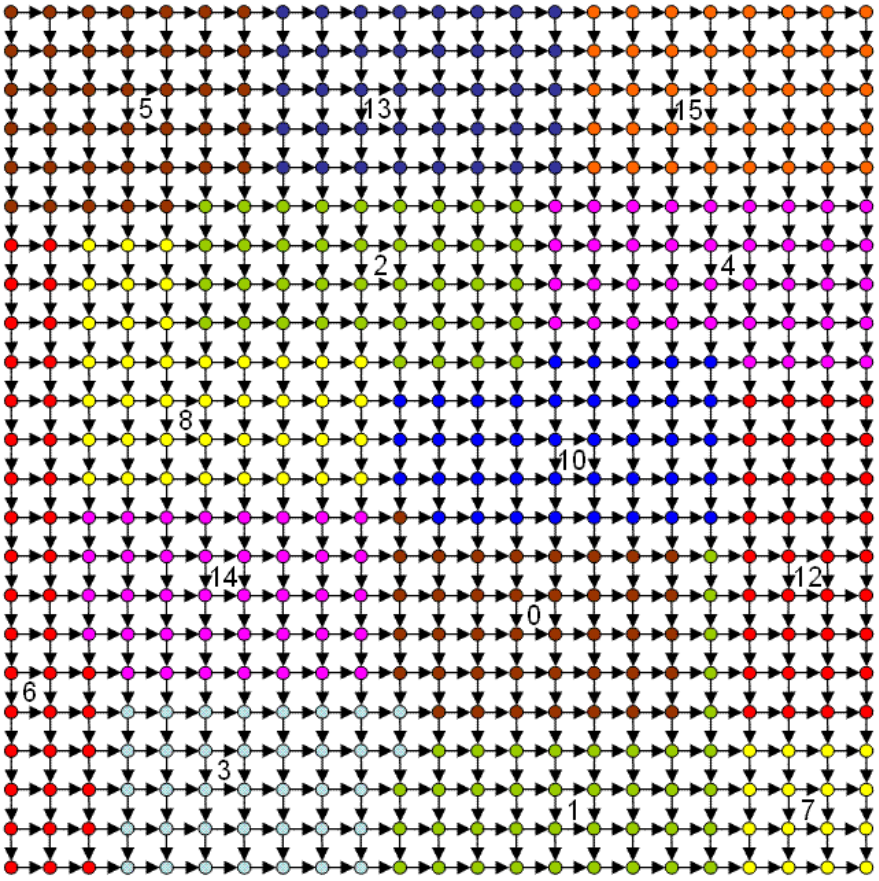


FIGURE 5.2 – Exemple de partitionnement d'une grille 23×23 sur 16 nœuds de capacité 40, 2 nœuds restent vides.

pour être optimale) et illustre bien, d'une part, que les solutions obtenues à l'aide de l'heuristique sont toujours proches de l'optimum (ou de la meilleure solution connue) et, d'autre part, que l'approche itérative par démarrage multiple (10 itérations) est significativement bénéfique sur une part non négligeable des instances.

Inst.	# sommets	Meil. connu	Part. init.	Avec multist.	Sol. aléa.
Gsub.500	500	206	249	236	585
G1000.0025	1000	95	147	118	614
G1000.005	1000	445	527	509	1184
G1000.01	1000	1362	1491	1461	2528
G1000.02	1000	3382	3576	3526	5024
G124.02	124	13*	17	15	73
G124.04	124	63*	74	68	151
G124.08	124	178	192	183	333
G124.16	124	449	479	471	629
G250.01	250	29*	40	36	173
G250.02	250	114	130	127	306
G250.04	250	357	397	378	629
G250.08	250	828	874	855	1260
G500.005	500	49*	70	61	309
G500.01	500	218	270	253	635
G500.02	500	626	680	669	1170
G500.04	500	1744	1826	1825	2535
U1000.05	1000	1*	20	6	1158
U1000.10	1000	39*	115	69	2364
U1000.20	1000	222	320	299	4678
U1000.40	1000	737	866	866	8931
U500.05	500	2*	20	12	653
U500.10	500	26*	72	68	1217
U500.20	500	178*	236	196	2309
U500.40	500	412	422	412	4438

TABLE 5.1 – Résultats expérimentaux sur les instances de Johnson et al. (1989).

5.3 Généralisation au partitionnement stochastique

Dans cette section, nous généralisons l'algorithme de la section précédente au cas où certains des poids des tâches, en particulier, les taux d'occupation de ressources de calcul, qui sont fonctions des temps d'exécution des noyaux de

calcul associés aux tâches, sont des variables aléatoires. Nous commençons par introduire une approche pragmatique générale de l'optimisation sous contraintes probabilistes, puis nous l'appliquons à notre problème de partitionnement afin d'obtenir une robustesse de réalisabilité au regard de l'indéterminisme inhérent à ces temps d'exécution.

5.3.1 Sur le caractère incertain des temps d'exécution

Dans nos problèmes d'affectation de ressources, l'une des principales sources d'incertitude réside dans l'indéterminisme des temps d'exécution des traitements qui correspondent à des noyaux de calcul de granularité moyenne. Cet indéterminisme est attribuable d'une part à certaines caractéristiques des architectures de processeur (présence de mémoires caches [2], d'arbitres d'accès, etc.) mais aussi, et de manière intrinsèque, à la présence de structures de branchement conditionnel et de boucles dépendantes des données d'entrée.

En l'occurrence, s'il est raisonnable de faire l'hypothèse que les lois de probabilité des temps d'exécution de noyaux de calcul embarqués sont à support borné [3], autrement dit que l'on embarque pas de boucles possiblement infinies [4], force est de constater qu'elles sont intrinsèquement multimodales (par exemple la loi du temps d'exécution du noyau "pour $i = 1$ à n si x alors S_1 sinon S_2", avec n variable entre 1 et N et où S_1 et S_2 sont deux séquences linéaires d'instructions, possède $2N$ modes). Ces lois se modélisent donc mal à l'aide du bestiaire des lois de probabilités usuelles, loi de Gauss et loi uniforme en tête, qui sont monomodes.

Enfin, lorsque de tels noyaux de calcul se trouvent intégrés dans un réseau de processus, se pose le problème des dépendances. Typiquement, on imagine faci-

2. L'effet des caches pouvant être atténué à l'aide de techniques de segmentation adaptées, cf. David & Sirdey (2010).

3. Une propriété qui n'est pas anodine puisqu'elle permet entre autres de garantir la finitude des moments et, en conséquence, l'utilisation de techniques statistiques basées sur des inégalités de type de Bienaymé-Tchebychev ou sur l'estimation du paramètre de position d'une loi de Weibull (cf. note 7 page 48), en tant que loi asymptotique de la valeur extrême de type III pour les variables aléatoires à support borné (Coles, 2001 ; de Haan & Ferreira, 2006), pour estimer des temps d'exécution pire cas. Entre parenthèses, nous pensons que la théorie de la valeur extrême n'a pas non plus reçu l'attention qu'elle mérite dans le domaine de l'optimisation combinatoire, en particulier lorsqu'elle se trouve associée à des algorithmes randomisés de résolution approchée : une expérience élémentaire révèle par exemple qu'en exécutant 3523 fois un recuit simulé sur l'instance `ch150` (par exemple) de TSPLib on obtient, à l'aide des outils de cette théorie, une estimation pour la valeur d'une solution optimale de 6520 (l'optimum réel étant de 6528) alors que la meilleure solution trouvée sur l'ensemble des exécutions est de 6703. La figure 5.3 illustre l'adéquation entre l'histogramme de la variable aléatoire $\max_{j\in\{10i,\ldots,10i+9\}} \mathrm{sa}_i$ ($i = 0$ à 3520), utilisée pour l'estimation, et la loi de Weibull. Bien entendu, exécuter un grand nombre de fois un recuit devient vite prohibitif en termes de temps de calcul et il faudrait considérer d'autres heuristiques plus rapides.

4. Bien entendu, en raison de l'indécidabilité du problème de l'arrêt, cette propriété ne peut être vérifiée qu'à l'aide de conditions suffisantes de terminaison.

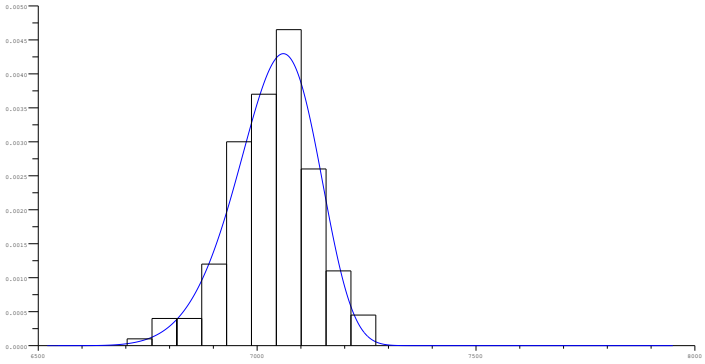


FIGURE 5.3 – Voir note 3, page 78.

lement que les temps d'exécution des différents étages d'un pipeline de suivi de cibles, par exemple, exhibent un certain de degré de dépendance sur le nombre de cibles effectivement traitées. Il convient donc d'assumer la multidimensionalité et de considérer des vecteurs aléatoires dont les lois jointes sont mieux modélisées par des unions d'hyperectangles disjoints que par des lois de Gauss multidimensionnelles (sans toutefois que les paramètres de ce premier modèle soient particulièrement aisés à déterminer, que ce soit par analyse statique, par analyse des données ou par combinaison de ces deux classes de techniques).

Lorsqu'il s'agit de prendre en compte ces types d'incertitudes dans nos problèmes d'optimisation, l'idéal est donc de se tourner vers des techniques non paramétriques fondées sur l'exploitation directe[5] de données expérimentales. L'obtention de telles données se réalise naturellement par le biais des boucles de compilation rétroactives telles qu'évoquées au chapitre 4. Également, toujours idéalement, ces méthodes doivent pouvoir s'intégrer sans trop de déstructuration dans des algorithmes de résolution, souvent déjà existants, pour le cas déterministe. Y compris lorsqu'il s'agit d'heuristiques pour les instances de grande taille.

5.3.2 L'approche par scénarios revisitée

Dans cette section, nous présentons une approche simple, pragmatique et non paramétrique du problème général de l'optimisation sous contraintes probabilistes. Nous illustrons notre propos à l'aide de la programmation linéaire sans toutefois en affecter la portée applicative générale.

5. Par opposition à des approches qui procéderaient de manière indirecte en commençant par valider un modèle à l'aide d'un test statistique d'adéquation puis à estimer les paramètres de ce modèle (e.g. le vecteur des espérances et la matrice des variances-covariances d'une loi de Gauss multidimensionnelle). À un moment, il faut bien faire le lien avec des valeurs expérimentales.

On commence par considérer le problème

$$\begin{cases} \text{Minimiser } c^T x, \\ \text{s. l. c.} \\ P(Ax \leq b) \geq 1 - \varepsilon, \end{cases} \tag{5.4}$$

où A est une matrice aléatoire $m \times n$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$ (ou $\mathbb{Z}^n$ dans le cas d'un problème discret) et $0 \leq \varepsilon \leq 1$. Le vecteur x étant l'inconnu.

On suppose donné un échantillon de réalisations de la matrice A : $\tilde{A}_1, \ldots, \tilde{A}_N$.

On pose

$$\chi_i = \begin{cases} 1 & \text{si } \tilde{A}_i x \leq b, \\ 0 & \text{sinon.} \end{cases}$$

Pour N suffisamment grand, en première approche, le programme mathématique (5.4) peut donc être approximé par le programme linéaire mixte suivant :

$$\begin{cases} \text{Minimiser } c^T x, \\ \text{s. l. c.} \\ \tilde{A}_i x \leq b + (1 - \chi_i)L, \qquad i = 1, \ldots, N, \\ \displaystyle\sum_{i=1}^{N} \chi_i \geq (1 - \varepsilon)N, & (5.5) \\ \chi_i \in \{0, 1\}, \qquad i = 1, \ldots, N, & (5.6) \end{cases}$$

où L est une constante de grande taille dépendante du problème [6].

La validité asymptotique de cette approximation ne dépend pas d'hypothèses particulières sur la loi jointe des coefficients de la matrice A, en particulier en termes d'indépendance entre les variables aléatoires associées à ces coefficients. On utilise simplement, étant donnés x et b, le prédicat $Ax \leq b$ comme un projecteur de la matrice aléatoire A sur un schéma de Bernoulli dont on cherche à estimer le paramètre.

Remarquons que l'on peut également poser le problème comme un problème d'optimisation bicritère (fonction économique et probabilité de réalisabilité) et chercher à approximer le front de Pareto avec les techniques de l'arsenal de l'optimisation multiobjectif.

Formellement, le programme ci-dessus possède la même structure qu'un programme issu de l'approche dite par scénarios, cf. Gaivoronski et al. (2011) pour une instanciation récente sur le problème du sac à dos quadratique. Dans cette approche on suppose donnée et connue une loi de probabilité discrète, à support fini, de la forme

$$\sum_{\mathcal{A} \in \Omega} p_{\mathcal{A}} \delta_{\mathcal{A}},$$

6. Par exemple, pour une contrainte de sac à dos de la forme $\sum_{i=1}^{n} w_i x_i \leq C$, avec $w_i > 0$, $\forall i$, L vaudrait simplement $\sum_{i=1}^{n} w_i$.

avec, bien entendu, $\sum_{\mathcal{A}\in\Omega} p_{\mathcal{A}} = 1$, où Ω est l'ensemble des événements et où δ_ξ dénote la distribution de Dirac centrée au point $\xi \in \mathbb{R}^{m\times n}$. D'où une paraphrase du programme (5.6) de la forme,

$$\begin{cases} \text{Minimiser } c^T x, \\ \text{s. l. c.} \\ \mathcal{A}x \leq b + (1-\chi_{\mathcal{A}})L, & \mathcal{A}\in\Omega, \\ \sum_{\mathcal{A}\in\Omega} p_{\mathcal{A}}\chi_{\mathcal{A}} \geq (1-\varepsilon), \\ \chi_{\mathcal{A}} \in \{0,1\}, & \mathcal{A}\in\Omega. \end{cases}$$

Il va sans dire que, en l'absence d'une théorie sous-jacente les justifiant, la validité de cette approche repose sur des hypothèses ad hoc et difficilement vérifiables. Dans la présente approche, au contraire, la philosophie est de justifier le modèle par la théorie de la statistique inférentielle et donc d'assumer pleinement le rôle que jouent les données expérimentales. Ceci nous permet, comme nous allons le voir dans la section suivante, de faire un lien direct avec la théorie des tests d'hypothèses statistiques afin de résoudre le problème initial (5.4), indépendamment de la loi jointe des coefficients de la matrice A, avec un niveau de confiance spécifié sur la tenue de la contrainte en probabilité. Ceci va également nous permettre de nous débarrasser du problème de la validité asymptotique et de donner un sens précis à la résolution du problème avec un échantillon de taille finie.

5.3.3 Lien avec la théorie des tests d'hypothèses

Étant donnés x et b, la variable aléatoire χ qui correspond au nombre de fois que le prédicat $Ax \leq b$ est satisfait sur un échantillon de taille N suit, par construction, une loi binomiale de paramètres N et p_0.

On peut alors chercher un seuil $k(N, 1-\varepsilon, \alpha)$ (k pour faire court) tel que

$$P(\chi \geq k | p_0 = 1-\varepsilon; N) \leq \alpha$$

où α est une (petite) probabilité d'erreur (typiquement 0.05 ou 0.01). Intuitivement, cela revient à fixer un seuil suffisamment haut pour que si on observe un nombre de violations supérieur au seuil alors il y a une probabilité suffisamment faible que ce nombre ait été engendré par la loi $\mathcal{B}(N, 1-\varepsilon)$ pour que l'on puisse conclure que, avec un niveau de confiance au moins $1-\alpha$, que $p_0 \geq 1-\varepsilon$. La table 5.2 illustre quelques valeurs pour le seuil k.

On note également, que l'on peut déterminer si la taille de l'échantillon est suffisante ou pas pour conclure au niveau de confiance requis. En particulier si

$$P(\chi = N-1 | p_0 = 1-\varepsilon; N) \geq \alpha$$

alors on peut affirmer que la taille de l'échantillon est insuffisante (ce qui est le cas pour $N = 10$ et $N = 20$ dans la table 5.2). Par exemple, pour conclure

N	$0.90N$	$k(N, 0.90, 0.10)$
10	9	-
20	18	-
30	27	29
40	36	38
50	45	48
100	90	94

TABLE 5.2 – Exemples de valeurs de $k(N, 0.90, 0.10)$ pour quelques valeurs de N. Commande Scilab (exemple) : `1-cdfbin("PQ",94,100,0.9,1-0.9)`.

qu'une contrainte en probabilité avec $\varepsilon = 0.05$ est satisfaite avec un niveau de confiance de 0.95 il faut au minimum un échantillon de taille 59 et le seuil k sera à 58 avec $P(\chi = 58|p_0 = 0.95; 59) \approx 0.048$. Toujours à titre d'exemple, la taille d'échantillon minimum pour $\varepsilon = 0.01$ et $1 - \alpha = 0.99$ est de 459.

Lorsque $k(N, 1 - \varepsilon, \alpha)$ est bien défini, on peut alors remplacer la contrainte (5.5) par la contrainte

$$\sum_{i=1}^{N} \chi_i \geq k(N, 1 - \varepsilon, \alpha).$$

En conséquence, si l'on résout le programme linéaire

$$\begin{cases} \text{Minimiser } c^T x, \\ \text{s. l. c.} \\ \tilde{A}_i x \leq b + (1 - \chi_i)L, \qquad i = 1, \ldots, N, \\ \displaystyle\sum_{i=1}^{N} \chi_i \geq k(N, 1 - \varepsilon, \alpha), \end{cases}$$

alors on peut affirmer que l'*on a résolu le programme mathématique* (5.4) *avec un niveau de confiance d'au moins* $1 - \alpha$.

Par souci didactique, nous avons illustré la présente approche sur la programmation linéaire. Néanmoins, la programmation linéaire (au sens large) n'est pas son domaine d'application privilégié car elle induit une augmentation importante du nombre de contraintes et que l'on passe, pour ce qui est de la programmation linéaire continue, à un modèle mixte. Comme nous allons le voir dans la section suivante, cette approche prend tout son sens pratique lorsqu'il s'agit d'intégrer la prise en compte d'incertitudes dans des algorithmes de résolution approchée ad hoc.

5.3.4 Extension de l'algorithme de la section 5.2

L'approche que nous avons illustrée sur la programmation linéaire dans la section précédente, s'adapte également simplement (et, cette fois, efficacement)

dans des contextes heuristiques : il suffit de compter les violations.

Dans le cadre du problème de partitionnement de graphe de la section 5.2, nous allons maintenant supposer que les poids des tâches, les S_{vr}, sont des variables aléatoires. La contrainte en probabilité se dérive alors de l'équation (5.1) et s'exprime simplement

$$P\left(\bigwedge_{n\in N}\bigwedge_{r\in R}\sum_{v\in V:f(v)=n} S_{vr} \leq C_r\right) \geq 1-\varepsilon.$$

Pour $k = 1$ à N, soit $\tilde{S}_{vr}^{(k)}$ la k-ième réalisation des poids des tâches.

Il convient donc de redéfinir les conditions d'admissibilité pour l'affectation d'une tâche à une partition (équation 5.2) et pour la fusion de deux partitions (équation 5.3) afin de s'assurer qu'à chaque étape de l'algorithme de la section 5.2.4, la contrainte en probabilité est satisfaite avec le niveau de confiance souhaité.

Dès lors, l'affectation d'un sommet v à un nœud n est admissible si la quantité

$$\sum_{k=1}^{N}\chi\left(\left\{\exists n'\neq n, \exists r : \sum_{w:f(w)=n'}\tilde{S}_{wr}^{(k)} > C_r\right\} \vee \left\{\exists r : \tilde{S}_{vr}^{(k)} + \sum_{w:f(w)=n'}\tilde{S}_{wr}^{(k)} > C_r\right\}\right), \tag{5.7}$$

où $\chi(P) = 1$ si et seulement si le prédicat P est vrai, est inférieure ou égale à $N-k(N, 1-\varepsilon, \alpha)$. À l'aide d'une structure de données ad hoc, un tableau booléen de taille N indiquant, pour le partitionnement partiel courant, si l'échantillon k induit déjà une violation, on peut simplifier l'équation (5.7) en

$$\sum_{k=1}^{N}\chi\left(t[k] \vee \left\{\exists r : \tilde{S}_{vr}^{(k)} + \sum_{w:f(w)=n'}\tilde{S}_{wr}^{(k)} > C_r\right\}\right),$$

et donc réaliser ce calcul avec un facteur d'augmentation de complexité limité à $O(N)$. L'entrée $t[k]$ étant elle même mise à jour lors de l'affectation effective de v à n dans le partitionnement partiel courant par

$$t[k] := t[k] \vee \left\{\exists r : \tilde{S}_{vr}^{(k)} + \sum_{w:f(w)=n'}\tilde{S}_{wr}^{(k)} > C_r\right\}.$$

D'une manière analogue, on peut généraliser le critère d'admissibilité d'une fusion entre un nœud n et un nœud m avec un facteur d'augmentation de la complexité linéaire en N.

Ceci illustre que notre approche par projection sur un test d'hypothèse statistique s'intègre sans difficultés majeures (que ce soit en termes de génie logiciel

ou de performances) dans le cadre d'un algorithme de résolution approchée existant. Ceci permet ainsi d'introduire, de manière non paramétrique et statistiquement maîtrisée, la prise en compte d'incertitudes avec un coût de mise en œuvre également maîtrisé.

Les tables 5.3, 5.4 et 5.5 fournissent quelques résultats expérimentaux. Ces résultats ont été obtenus par Oana Stan à l'aide d'une adaptation de l'algorithme de la section 5.2 qu'elle a programmée et utilisée pour résoudre le problème de partitionnement sur les grilles 4×4 (16 sommets de poids unitaires sur 4 nœuds de capacité 4 dans le cas déterministe), 10×10 (100 sommets de poids unitaires sur 5 nœuds de capacité 20 dans le cas déterministe) et 23×23 (529 sommets de poids unitaires sur 14 nœuds de capacité 40 dans le cas déterministe), de la section 5.2.6, pour différentes valeurs des paramètres N, ε et $1 - \alpha$, et où les vecteurs aléatoires de poids des tâches sont issus d'une loi jointe bimodale, uniforme sur ses modes. Le premier mode correspond à l'hypercube

$$[0.8, 0.9]^{|T|},$$

et le second à

$$[1.1, 1.2]^{|T|}.$$

Les deux modes étant sélectionnés de manière équiprobable.

Type	# nœuds	coût	CPU	C	coût	CPU
Grille 4×4	6	14	≈ 0	4.71	12	≈ 0
Grille 10×10	6	38	0.02 s	23.3	29	0.01 s
Grille 23×23	16	182	1.12 s	44.1	173	0.99 s

TABLE 5.3 – Partitionnement stochastique avec $N = 100$, $\varepsilon = 0.05$, $1-\alpha = 0.95$.

Type	# nœuds	coût	CPU	C	coût	CPU
Grille 4×4	6	14	≈ 0	4.712	12	≈ 0
Grille 10×10	6	37	0.16 s	23.273	37	0.13 s
Grille 23×23	16	182	11.23 s	44.13	172	9.65 s

TABLE 5.4 – Partitionnement stochastique avec $N = 1000$, $\varepsilon = 0.05$, $1-\alpha = 0.95$.

Type	# nœuds	coût	CPU	C	coût	CPU
Grille 4×4	6	14	≈ 0	4.74	10	≈ 0
Grille 10×10	6	37	0.15 s	23.36	37	0.13 s
Grille 23×23	16	182	10.75 s	44.183	193	9.67 s

TABLE 5.5 – Partitionnement stochastique avec $N = 1000$, $\varepsilon = 0.01$, $1-\alpha = 0.99$.

Dans chacune des tables nous avons donné le nombre de nœuds à partir duquel la contrainte en probabilité est satisfaite (colonne "# nœuds"), le coût de la solution obtenue (première colonne "coût") et le temps moyen pour une exécution de l'algorithme (première colonne "CPU"), sachant que 10 exécutions ont été réalisées en multistart. Dans un second temps, à nombre de nœuds constant, nous avons indiqué la capacité nœud à partir de laquelle la contrainte en probabilité est satisfaite (colonne "C") ainsi que, de nouveau, le coût de la solution et le temps d'exécution unitaire moyen (secondes colonnes "coût" et "CPU", respectivement). Ces résultats sont à mettre en perspective avec ceux obtenus dans le cas déterministe donnés à la section 5.2, cas qui correspond ici au cas moyen. À noter que les solutions obtenues dans le cas moyen ne sont réalisables qu'avec une probabilité de 0.5 (légèrement supérieure dans le cas de la grille 23×23).

5.4 Placement des partitions

5.4.1 Énoncé et complexité du problème

Soit N l'ensemble des partitions obtenues à l'aide de l'algorithme de la section 5.2 (ou de sa généralisation de la section 5.3). On note $H = (N, B)$ le graphe réduit dont les sommets sont les partitions et tel que deux partitions n et m sont connectées par un arc (n, m) s'il existe dans G au moins un arc reliant un sommet de n à un sommet de m. On a $|N| = |K|$, K étant l'ensemble des clusters de l'architecture.

Est également donnée une matrice de distance intercluster D. Ceci suppose que le plan de routage est donné a priori, au moins sur le plan des distances.

Le problème d'affectation que nous avons à résoudre consiste alors à trouver une bijection $g : N \longrightarrow K$ qui minimise

$$\sum_{(n,m)\in B} D_{g(n)g(m)}.$$

Le problème ci-dessus est un QAP (Quadratic Assignment Problem) général. Il s'agit d'un problème NP-difficile au sens fort bien connu.

5.4.2 Bref état de l'art et positionnement

Le QAP est un problème "star" qui, en tant que tel, a déjà été bien étudié sur le plan de la résolution exacte et de la résolution approchée (Loiola et al., 2007). Ce problème fait néanmoins partie des problèmes combinatoires parmi les plus difficiles et les meilleurs méthodes de résolution exacte sont limitées à des instances ayant de l'ordre de 30 nœuds [7].

7. Une difficulté qui s'explique vraisemblablement, au moins partiellement, par le fait que le QAP appartient à une classe de problèmes combinatoires pour lesquels, asymptotiquement

Sur le plan de la résolution approchée, le problème a été adressé à l'aide du bestiaire des métaheuristiques. Il existe par contre, à notre connaissance, très peu d'algorithmes constructifs référencés dans la littérature.

Dans le cadre de notre contexte applicatif, les instances à résoudre auront entre 16 et 64 nœuds (et au-delà dans le cas d'une application à placer sur plusieurs puces). Si les plus petites instances ne présentent pas de difficultés particulières [8], tel n'est pas le cas, comme nous l'avons vu à la section précédente, pour les instances de taille 64 qui sont hors de portée des méthodes de résolution exactes actuelles.

Sur ce problème, une approche par recuit simulé est séduisante. D'une part, comme nous l'avons déjà dit, il existe des schémas généraux de fixation des paramètres permettant de tenir compte d'une distance à l'optimum cible (Sirdey et al., 2009a). D'autre part, la conception d'une fonction de voisinage garantissant la connexité forte du graphe qu'elle engendre est triviale pour ce problème : 2-OPT fait l'affaire.

Nul besoin de souligner que nous ne cherchons pas ici à "battre un quelconque record du monde" sur le QAP mais bien à définir un algorithme de résolution adapté à un contexte applicatif en matière de qualité et de stabilité des solutions obtenues ainsi qu'en matière de génie logiciel, en tenant compte des contraintes (temps d'exécution notamment) et des particularités (spectre des tailles d'instances) de l'application.

5.4.3 Un algorithme de recuit simulé

Soit Ω l'ensemble des solutions d'un problème combinatoire de minimisation. On dit qu'une solution $\omega \in \Omega$ est (α, β)-acceptable si

$$P(c(\omega) \leq e_1 + \beta(e_P - e_1)) \geq \alpha$$

où e_1 et e_P dénotent respectivement la valeur d'une meilleure et d'une pire solution. Il s'agit là du principe de l'approximation différentielle introduit par Demange & Paschos (1996).

On peut alors montrer (Sirdey et al., 2009a) que, pour tout $e_1 \leq z_1 < z_P \leq e_P$, les solutions issues de la loi stationnaire de l'algorithme du recuit simulé à la température

$$T_f(z_1, z_P) = \frac{\beta(z_P - z_1)}{\log|\Omega| - \log(1-\alpha)}$$

sont (α, β)-acceptables.

---

en la taille du problème, toutes les solutions sont quasi-optimales (Burkard & Fincke, 1985).

8. Un algorithme de branch-and-bound même relativement simple et séquentiel vient à bout d'une instance du QAP de taille 16 en de l'ordre d'une centaine de secondes dans le pire cas expérimental, sans compter qu'un facteur d'accélération linéaire peut être obtenu à l'aide de schémas de parallélisation simples (Maurin & Sirdey, 2011). En même temps il n'y a "que" $2^{44.25}$ possibilités...

Le principe consiste alors à utiliser un schéma de décroissance de T pour converger de proche en proche vers cette loi stationnaire. Cela donne un algorithme dont le principe est analogue au schéma général défini dans Sirdey et al. (2009a) et que nous ne détaillerons par conséquent pas ici.

Pour le QAP, le calcul de la mise à jour de la fonction économique peut être implémenté en $O(|N|)$, on fait $|N|$ itérations par paliers et le nombre de paliers est en $O(U|N|\log|N|)$ (ibid.). D'où une complexité pseudopolynomiale en

$$O(U|N|^3 \log|N|).$$

5.4.4 Résultats expérimentaux

Nous illustrons les résultats ainsi obtenus en partant du partitionnement d'une grille 23x23 (529 sommets) sur 16 nœuds de capacité 40. Le partitionnement a été donné à la figure 5.2. Dans ce partitionnement, 2 nœuds sont vides. La solution est de valeur 150 (cf. section 5.2.6).

On cherche à placer les nœuds sur un réseau sur puce à 16 clusters interconnectés selon une topologie en tore bidimensionnel 4×4. Sous l'hypothèse d'un routage de type plus court chemin la matrice de distance peut alors être obtenue à l'aide de l'algorithme bien connu de Floyd-Warshall. D'où la matrice donnée dans l'encadré 3 où les distances sont reportées en nombre de hops (nombre d'arcs -1, donc).

$$\begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 1 & 2 & 1 & 1 & 2 & 3 & 2 & 0 & 1 & 2 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 2 & 2 & 1 & 2 & 3 & 1 & 0 & 1 & 2 \\
1 & 0 & 0 & 0 & 2 & 1 & 0 & 1 & 3 & 2 & 1 & 2 & 2 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 2 & 1 & 0 & 2 & 3 & 2 & 1 & 1 & 2 & 1 & 0 \\
0 & 1 & 2 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 2 & 1 & 1 & 2 & 3 & 2 \\
1 & 0 & 1 & 2 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 2 & 2 & 1 & 2 & 3 \\
2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 2 & 1 & 0 & 1 & 3 & 2 & 1 & 2 \\
1 & 2 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 2 & 1 & 0 & 2 & 3 & 2 & 1 \\
1 & 2 & 3 & 2 & 0 & 1 & 2 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 2 & 1 \\
2 & 1 & 2 & 3 & 1 & 0 & 1 & 2 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 2 \\
3 & 2 & 1 & 2 & 2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 2 & 1 & 0 & 1 \\
2 & 3 & 2 & 1 & 1 & 2 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 2 & 1 & 0 \\
0 & 1 & 2 & 1 & 1 & 2 & 3 & 2 & 0 & 1 & 2 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 2 & 2 & 1 & 2 & 3 & 1 & 0 & 1 & 2 & 0 & 0 & 0 & 1 \\
2 & 1 & 0 & 1 & 3 & 2 & 1 & 2 & 2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 2 & 1 & 0 & 2 & 3 & 2 & 1 & 1 & 2 & 1 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}$$

Encadré 3: Matrice de distances sur un tore 2D 4×4, cf. texte.

Dans ces conditions, l'algorithme de la section 5.4.3 fournit une solution de coût 10 (10 arcs du graphe réduit, sur 33, ne sont pas directs mais à 1 hop) et

cette solution est en réalité optimale comme nous avons pu le vérifier à l'aide d'un algorithme de branch-and-bound.

Enfin, toujours pour se fixer les idées, le tableau ci-dessous indique les temps de calcul observés sur une station Linux lambda pour le placement des graphes réduits associés aux grilles de la section 5.2.6.

Type	# nœuds	temps CPU
Grille 23×23	16	0.09 secs
Grille 46×46	64	3.7 secs
Grille 100×100	200	3 m 37 s

TABLE 5.6 – Illustration des temps de calcul de l'algorithme de la section 5.4.3.

5.5 Calcul des chemins de routage

Une fois les tâches placées sur les clusters de l'architecture, il convient de calculer les chemins d'acheminement des données.

5.5.1 Modèle de multiflot

Soit F l'ensemble des flots inter-clusters. Soit $G = (V, A)$ le graphe orienté qui représente le réseau sur puce. Étant donné un flot $f \in F$, on note $s(f)$, $d(f)$ et $w(f)$, respectivement, la source, le puits et le débit du flot f. Cette dernière quantité correspondant à la somme des débits entre les tâches affectées à $s(f)$ et celles affectées à $d(f)$. Enfin, étant donné un arc $a \in A$, soit C_a la capacité du brin du réseau associé.

De manière assez naturelle, le problème du calcul des chemins d'acheminement se met sous la forme d'un modèle de multiflot tel qu'énoncé à l'encadré 4, où x_{fa} est la quantité du flot f acheminée sur l'arc a et où, pour la fonction économique (5.8),

$$\gamma_{fa} = \begin{cases} 0 & \text{s'il existe un plus court chemin de } s(f) \text{ à } d(f) \text{ passant par } a, \\ 1 & \text{sinon.} \end{cases}$$

Une solution de coût 0 indique alors que tout le trafic a pu être acheminé via des plus courts chemins, comme le stipulait la matrice de distance donnée a priori à l'étape de placement de la section 5.4. Une solution de coût strictement positif permet de quantifier la déviation. Alternativement, il est possible de procéder par contrainte i.e., de remplacer la fonction économique (5.8) par les contraintes

$$x_{fa} \leq C_a(1 - \gamma_{fa}),\ \forall f \in F, \forall a \in A,$$

mais, ce faisant, on court bien évidemment le risque d'avoir des problèmes de réalisabilité.

$$
\begin{cases}
\text{Minimiser } \displaystyle\sum_{a\in A}\sum_{f\in F}\gamma_{fa}x_{fa}, & (5.8)\\
\text{s. l. c.}\\
\displaystyle\sum_{a\in\delta^+(s(f))} x_{fa} = w(f), & \forall f\in F,\\
\displaystyle\sum_{a\in\delta^-(d(f))} x_{fa} = w(f), & \forall f\in F,\\
\displaystyle\sum_{a\in\delta^-(v)} x_{fa} = \sum_{a\in\delta^+(v)} x_{fa}, & \forall f\in F, \forall a\in A\setminus\{s(f),d(f)\},\\
0\leq x_{fa}\leq C_a & \forall f\in F, \forall a\in A.
\end{cases}
$$

Encadré 4: Modèle de multiflot pour le calcul des chemins d'acheminement.

Posé en ces termes, il s'agit donc d'un problème de multiflot tout à fait classique que l'on peut résoudre à l'aide d'un algorithme pour la programmation linéaire.

5.5.2 Contraintes de chemins uniques

Afin de simplifier les protocoles de communication (notamment pour éviter les problèmes de restauration de la causalité à la réception) il est souhaitable, au moins dans les premières versions d'un système, de se limiter à un seul chemin d'acheminement par flot. Pour se faire, on ajoute les variables bivalentes, $\forall f\in F$, $\forall a\in A$,

$$\chi_{fa} = \begin{cases} 1 & \text{si } a \text{ peut écouler une part non nulle du flot } f,\\ 0 & \text{sinon.}\end{cases}$$

On ajoute alors les contraintes, $\forall f\in F$, $\forall v\in V$,

$$\sum_{a\in\delta^-(v)}\chi_{fa}\leq 1$$

ainsi que

$$\sum_{a\in\delta^+(v)}\chi_{fa}\leq 1.$$

Bien entendu, on obtient alors un programme linéaire mixte. Ce problème est NP-difficile au sens fort par restriction triviale au problème Directed Edge Disjoint Paths (Korte & Vygen, 2000) [9].

9. Également, par restriction un peu moins triviale au problème 3-partition.

En termes de résolution pratique, cet état de fait n'est pas problématique. En effet, nos instances pratiques sont de taille suffisamment petite (au plus 240 flots pour 64 liens) pour pouvoir être résolues en quelques secondes avec les solveurs sur étagère que sont CPLEX et COIN-CBC. Mentionnons également que les tests que nous avons réalisés avec GLPK n'ont par contre pas été convaincants.

5.6 Discussion

Exception faite des algorithmes de la section 5.3 sur le partitionnement stochastique, nous l'avons déjà dit, les algorithmes présentés dans ce chapitre ont fait l'objet d'un premier transfert industriel. Comme nous l'avons illustré, ces algorithmes ont l'avantage de permettre d'obtenir rapidement des résultats de qualité, ce qui les rend incontournables en début du cycle de développement d'une application parallèle où le programmeur doit avoir à sa disposition une boucle de test rapide de son application. Ils ont également une complexité de mise en œuvre modérée, ce qui a permis de les prototyper et de les transférer rapidement.

Le principal défaut de cette approche par décomposition est qu'en remplaçant la résolution d'un problème d'affectation complexe par la résolution séquentielle de trois problèmes moins complexes (bien que tous NP-difficiles), nous avons déstructuré le problème initial. En particulier, il peut se poser des problèmes de réalisabilité sur les problèmes en aval en raison de la relaxation implicite de certaines contraintes en amont. Typiquement, par exemple, il est possible que l'ensemble des partitions engendré à la première étape ne soit pas routable simplement parce que le débit entre deux de ces partitions excède $\max_{a \in A} C_a$.

Il convient alors de résoudre le problème global d'affectation et de routage. Par contre, la définition d'un modèle mathématique unique est malaisée et il est plus approprié de considérer un binôme problème maître (placement) et esclave (routage). Par exemple, une approche consiste à augmenter les tests d'admissibilité pour l'affectation d'une tâche à une partition (équation 5.2) et pour la fusion de deux partitions (équation 5.3) d'une vérification de réalisabilité du routage revenant à s'assurer que le système linéaire d'inégalités mixtes associé au programme de la section 5.5.2 est cohérent. Ceci suppose donc, à chaque étape de l'algorithme de la section 5.2.4, un calcul de placement du partitionnement partiel (algorithme de la section 5.4.3) et la résolution du système d'inégalités mixte à l'aide de CPLEX ou de COIN-CBC. Même si l'on peut tirer partie des capacités de warm starting de ces solveurs, il est à prévoir que cela induise une augmentation importante, mais non prohibitive, des temps de calculs. On estime qu'une exécution complète de l'algorithme de la section 5.2.4 ainsi étendu passerait de quelques secondes à quelques dizaines de minutes et qu'il faudrait avoir recours au parallélisme pour les aspects multistart (section 5.2.5), ce dernier point ne présentant pas de difficultés particulières. De tels temps de compilation restent tout à fait acceptables pour le domaine de l'embarqué, en fin de cycle de

développement. À noter, également, que cette approche s'intègre sans difficulté avec l'approche pour le partitionnement stochastique de la section 5.3 (dès lors que l'on se limite à prendre en compte une incertitude sur les poids et non sur la fonction économique).

Chapitre 6

Vérification de réseaux de processus

6.1 Introduction

Comme nous l'avons vu au chapitre 4, l'avènement des architectures multicœurs et massivement multicœurs a conduit à un regain d'intérêt pour les modèles flot de données, modèles dans lesquels on exprime une application de calcul intensif de manière parallèle, sous la forme de réseaux de tâches concurrentes communiquant par le biais de canaux FIFO unidirectionnels, et uniquement par ce biais. Comme nous l'avons également vu, l'intérêt pratique de ces modèles est de libérer le programmeur de l'explicitation des synchronisations inter-tâches, l'une si ce n'est la principale des difficultés de la programmation parallèle, ainsi que de permettre l'expression naturelle d'un parallélisme suffisant pour adresser des architectures à plusieurs centaines de cœurs, au moins dans le domaine du traitement du signal et de l'image.

Dans ce chapitre, nous nous attachons à fournir un modèle des états d'un réseau de processus flot de données (DPN), au sens le plus général de Lee & Parks (1995) à l'aide de la programmation linéaire (en nombres entiers). Ce modèle nous permet principalement d'obtenir des conditions suffisantes de vivacité (dans le sens défini à la section 6.3) et d'exécution sans deadlock en mémoire bornée (cf. section 6.4), certaines de ces conditions étant polynomiales. Également, cette approche peut être étendue afin d'obtenir une méthode de dimensionnement sûre des tampons de communication. Il convient de souligner que le modèle DPN ne doit pas être confondu avec les modèles synchrone (SDF) de Lee & Messerschmitt (1987) et cyclostatique (CSDF) de Bilsen et al. (1996) qui sont beaucoup plus connus et restrictifs. Par essence, le modèle DPN a la même puissance que celui des réseaux de processus de Kahn et, en tant que tel, certaines propriétés telle que l'exécution en mémoire bornée sont indécidables (alors que ce n'est pas le cas dans les deux modèles précédemment mentionnés ainsi que dans le cas de nombre

de leurs variantes).

Ces travaux ont été réalisés en collaboration avec Pascal Aubry et ont fait l'objet d'une première publication dans les actes du workshop d'informatique théorique *Interactions and Concurrency Experience* (Sirdey & Aubry, 2010), cf. également (Sirdey, 2008b ; Aubry, 2009).

6.2 Modélisation des états d'un réseau

6.2.1 Notations et hypothèses

Soit T et F les ensembles de tâches et de canaux, respectivement.

À chaque tâche $t \in T$, on associe un multigraphe d'états-transitions $G_t = (\{v_t^{(0)}\} \cup V_t, \{\tau_t^{(0)}\} \cup A_t)$ (i.e., les arcs parallèles et les boucles sont autorisés), où $v_t^{(0)}$ dénote l'état initial de la tâche t et où $\tau_t^{(0)}$ dénote la transition initiale de cette même tâche, cette transition étant unique et systématiquement exécutée. Aussi, étant donnée $t \in T$, $P_t \subseteq F$ (respectivement $C_t \subseteq F$) dénote l'ensemble des canaux dans lesquels t produit (respectivement consomme) des données. À noter que l'on a $P_t \cap C_t = \emptyset$. Également, à noter que pour tout $t, t' \in T$, $t \neq t'$, on a $P_t \cap P_{t'} = \emptyset$ ainsi que $C_t \cap C_{t'} = \emptyset$.

Soit $t \in T$, $\tau \in A_t$ et $f \in P_t$ (respectivement $f \in C_t$), $\mathsf{qp}_{\tau f}$ (respectivement $\mathsf{qc}_{\tau f}$) indique la quantité de données produite (respectivement consommée) sur le canal f, par la tâche t, lorsque la transition τ est exécutée. Une contrainte supplémentaire est que, $\forall t \in T$, $\forall \tau \in A_t$,

$$\sum_{f \in P_t} \mathsf{qp}_{\tau f} + \sum_{f \in C_t} \mathsf{qc}_{\tau f} > 0. \tag{6.1}$$

Ce faisant, l'existence de transitions sans effet sur les canaux d'entrée/sortie est exclue.

Étant donné $f \in F$, p_f et c_f dénotent respectivement les tâches productrice et consommatrice sur le canal f. Également, d_f dénote la capacité du tampon associé au canal f (selon le problème, d_f peut être soit une donnée soit une inconnue, cf. ci-après).

Dans la suite de ce chapitre, on fait l'hypothèse supplémentaire, sans perte de généralité, que le graphe représentant le réseau de processus est (simplement) connexe.

6.2.2 Variables et contraintes linéaires

Afin de représenter les états du réseau, pour tout $\tau \in \bigcup_{t \in T} \{\tau_t^{(0)}\} \cup A_t$, on introduit une variable $n_\tau \in \mathbb{Z}^+$ qui indique le nombre de fois que la transition τ a été exécutée pour mettre le réseau dans un état donné. De manière à ce que les n_τ représentent un état admissible du système, un certains nombre de contraintes (que nous allons mettre sous forme linéaire) doivent être satisfaites.

Contraintes d'initialisation. Soit $t \in T$, la transition initiale $\tau_t^{(0)}$ doit avoir été exécutée une et une seule fois d'où, $n_{\tau_t^{(0)}} = 1$.

Contraintes de conservation. Soient $t \in T$ et $v \in V_t$. On doit alors avoir la contrainte suivante[1] :

$$\sum_{\tau \in \omega^-(v)} n_\tau - 1 \leq \sum_{\tau \in \omega^+(v)} n_\tau \leq \sum_{\tau \in \omega^-(v)} n_\tau.$$

Une telle contrainte reflète le fait que dans un état admissible du système, on doit être entré dans le sommet v autant de fois que l'on en est sorti, à un près. Lorsque

$$\sum_{\tau \in \omega^+(v)} n_\tau = \sum_{\tau \in \omega^-(v)} n_\tau - 1,$$

alors l'état du système décrit par les n_τ est tel que la tâche t est dans l'état v.

Pour simplifier l'écriture on pose

$$\gamma_v = \sum_{\tau \in \omega^-(v)} n_\tau - \sum_{\tau \in \omega^+(v)} n_\tau.$$

On a alors $\gamma_v = 0$ lorsque que la tâche t n'est pas dans l'état v et 1 sinon. Remarquons que l'état $v_t^{(0)}$ est dûment exclu, dans la mesure où l'on ne sort qu'une fois de cet état sans jamais y entrer, par définition.

Contraintes d'unicité. De plus, chaque tâche doit être dans un et un seul état. Il suit que pour chaque $t \in T$, on doit avoir

$$\sum_{v \in V_t} \gamma_v = 1.$$

Là encore, $v_t^{(0)}$ est dûment exclu de la somme.

Contraintes de cohérence. Soit $f \in F$, on pose

$$\mathsf{qp}_f = \sum_{\tau \in A_{p_f}} n_\tau \mathsf{qp}_{\tau f}$$

et

$$\mathsf{qc}_f = \sum_{\tau \in A_{c_f}} n_\tau \mathsf{qc}_{\tau f},$$

quantités qui dénotent respectivement les quantités de données produites et consommées jusqu'à présent sur le canal f. On doit alors avoir la contrainte évidente,

$$\mathsf{qp}_f \geq \mathsf{qc}_f.$$

1. Avec les notations usuelles de la théorie des graphes pour les ensembles d'arcs incidents à l'intérieur et à l'extérieur.

Contraintes de capacité. Enfin, pour chaque $f \in F$, on doit également avoir la contrainte

$$\mathsf{qp}_f - \mathsf{qc}_f \leq d_f. \tag{6.2}$$

L'encadré 5 résume le système linéaire d'inégalités en nombres entiers ainsi obtenu. Une solution de ce système définit ce que nous appelons un état admissible du système. Il convient de souligner, néanmoins, que dans la mesure où l'on ne se préoccupe pas de la manière avec laquelle le système pourrait arriver dans un tel état, tous les états admissibles ne sont pas nécessairement réalisables c'est-à-dire accessible depuis l'état initial du système. Posé en ces termes, l'ensemble des solutions de ce système fournit une surapproximation polyèdrale des états du réseau.

$$\begin{cases}
\displaystyle\sum_{\tau\in\omega^+(v)} n_\tau - \sum_{\tau\in\omega^-(v)} n_\tau \leq 0, & \forall t \in T, \forall v \in V_t, \\
\displaystyle\sum_{\tau\in\omega^-(v)} n_\tau - \sum_{\tau\in\omega^+(v)} n_\tau \leq 1, & \forall t \in T, \forall v \in V_t, \\
\gamma_v = \displaystyle\sum_{\tau\in\omega^-(v)} n_\tau - \sum_{\tau\in\omega^+(v)} n_\tau, & \forall t \in T, \forall v \in V_t, \\
\displaystyle\sum_{v\in V_t} \gamma_v = 1, & \forall t \in T, \\
\mathsf{qp}_f = \displaystyle\sum_{\tau\in A_{p_f}} n_\tau \mathsf{qp}_{\tau f}, & \forall f \in F, \\
\mathsf{qc}_f = \displaystyle\sum_{\tau\in A_{c_f}} n_\tau \mathsf{qc}_{\tau f}, & \forall f \in F, \\
\mathsf{qp}_f - \mathsf{qc}_f \geq 0, & \forall f \in F, \\
\mathsf{qp}_f - \mathsf{qc}_f \leq d_f, & \forall f \in F, \\
n_{\tau_t^{(0)}} = 1, & \forall t \in T, \\
n_\tau \geq 0, & \forall \tau \in \cup_{t\in T} A_t, \\
n_\tau \in \mathbb{Z}, & \forall \tau \in \cup_{t\in T} A_t.
\end{cases}$$

Encadré 5: Système linéaire d'inégalités en nombres entiers dont les solutions sont les états admissibles d'un DPN.

6.3 Modélisation de propriétés systèmes indésirables

6.3.1 Blocage d'une tâche aux sens fort et faible

Dans un état du réseau donné, une tâche $t \in T$ est dite bloquée au sens fort lorsqu'elle se trouve dans un état v dont aucune des transitions de sortie ne peut être exécutée. Considérons le jeu de contraintes suivant, pour chaque $\tau = (v, v') \in A_t$,

$$\begin{cases} \gamma_v \leq 0, & (6.3) \\ \mathsf{qp}_f - \mathsf{qc}_f \leq \mathsf{qc}_{\tau f} - 1, \quad \text{pour chaque } f \in C_t, & (6.4) \\ \mathsf{qc}_f - \mathsf{qp}_f \leq \mathsf{qp}_{\tau f} - d_f - 1, \quad \text{pour chaque } f \in P_t. & (6.5) \end{cases}$$

Alors, pour une tâche t donnée, la propriété de blocage au sens fort signifie que pour chaque $\tau = (v, v') \in A_t$ soit la contrainte (6.3) est satisfaite ou au moins l'une des contraintes de type (6.4) ou au moins une des contraintes de type (6.5) est satisfaite.

Dans un état du système donné, une tâche $t \in T$ est dite bloquée au sens faible lorsqu'elle se trouve dans un état v dont toutes les transitions de sortie ne peuvent être exécutées. Considérons alors l'ensemble de contraintes suivant, pour chaque $t \in T$ et pour chaque $v \in V_t$,

$$\begin{cases} \gamma_v \leq 0, & (6.6) \\ \mathsf{qp}_f - \mathsf{qc}_f \leq \mathsf{qc}_{\tau f} - 1, \quad \text{pour chaq. } \tau = (v, v') \in A_t,\ f \in C_t, & (6.7) \\ \mathsf{qc}_f - \mathsf{qp}_f \leq \mathsf{qp}_{\tau f} - d_f - 1, \quad \text{pour chaq. } \tau = (v, v') \in A_t,\ f \in P_t. & (6.8) \end{cases}$$

Alors, pour une tâche t donnée, la propriété de blocage au sens faible signifie que pour chaque $v \in V_t$ soit la contrainte (6.6) ou au moins l'une des contraintes de type (6.7) ou au moins l'une des contraintes de type (6.8) est satisfaite.

La propriété de blocage au sens fort est adaptée aux tâches non déterministes (typiquement un select, cf. chapitre 4) alors que la propriété de blocage au sens faible, quant à elle, est adaptée aux tâches déterministes : il est suffisant de s'assurer que lorsqu'une tâche est dans un état donné toutes les transitions en sortie de cet état sont exécutables afin de garantir que la transition que la tâche *doit* faire (et qui est fonction de son état interne qui échappe au modèle) est possible.

Lorsque le système est dans un état tel que toutes les tâches sont bloquées (au sens fort ou faible selon les cas), on parle d'état de deadlock. Un réseau qui admet de tels états est non vivace.

6.3.2 Conditions suffisantes de vivacité

Bien que les définitions des propriétés de blocage aux sens fort et faible mettent en jeu des contraintes disjonctives, ces dernières peuvent être linéarisées à l'aide de techniques standards (Nemhauser & Wolsey, 1999 ; Billionnet, 2007).

Il suit qu'étant donné un DPN et un vecteur de dimensionnement $d \in \mathbb{Z}^{|F|}$ nous avons montré comment formuler un système linéaire d'inégalités en nombres entiers,

$$\{x \in \mathbb{Z}^n : Ax \leq b(d)\} \tag{6.9}$$

dont l'incohérence, i.e. le fait que $\{x : Ax \leq b(d), x \in \mathbb{Z}^n\} = \emptyset$, suffit à établir la vivacité du réseau. A fortiori, l'incohérence de la relaxation continue de ce système, $\{x : Ax \leq b(d), x \in \mathbb{R}^n\} = \emptyset$, fournit également une condition suffisante plus faible certes, mais polynomiale.

Pour se fixer les idées, notre définition de la vivacité pour un DPN est l'analogue de la propriété de pseudo-vivacité [2] (Diaz, 2001) de la théorie des réseaux de Petri qui stipule que pour tout marquage accessible depuis le marquage initial, il existe au moins une transition franchissable. Cette propriété est $PSPACE$-complète dans le cas général et l'on ne connait que deux cas particuliers assez restrictifs : les réseaux à choix libre et les réseaux sans conflits pour lesquels elle est respectivement NP-complète et polynomiale (Esparza & Nielsen, 1994). Cette propriété est bien évidemment vérifiable, en principe, par examination du graphe d'accessibilité, lorsque celui-ci est fini. Dans la mesure où un DPN dimensionné possède un nombre fini d'états, nous pourrions, en principe, procéder par examination du graphe des états du système. Au lieu de cela, nous cherchons in fine à remplacer une énumération par une autre, certes au prix d'une relaxation de l'accessibilité, en comptant sur le fait que, par définition des algorithmes sous-jacents à un solveur de PLNE, une partie de l'énumération sera implicite [3].

Enfin, précisons qu'il est possible de construire un système d'addition de vecteurs à états (SAVE), donc un réseau de Petri, à partir de notre modèle pour les DPN mais qu'un tel SAVE possède un nombre exponentiel d'états dans le cas général. Les états de ce SAVE correspondent en effet aux sommets du produit synchronisé des graphes d'états-transitions des tâches.

2. Brams (1983) parle de réseau sans blocage et la littérature anglo-saxonne de "deadlock freedom" (Esparza & Nielsen, 1994). Précisons également que la littérature anglo-saxonne utilise parfois le terme "deadlock" (verrou ou siphon) pour qualifier un ensemble de places qui, lorsqu'elles ne sont pas initialement marquées, ne pourront jamais l'être e.g., Minoux & Barkaoui (1990) ; Murata (1989). Une notion connexe donc, mais autre.

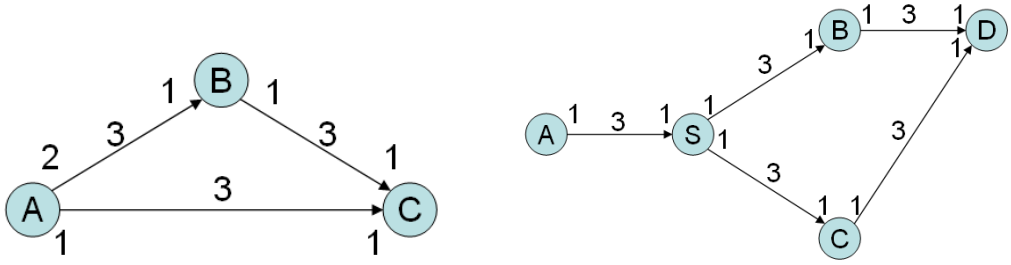
3. La philosophie est ici analogue à celle des techniques par énumération implicite des chemins (IPET), fondées sur la programmation linéaire en nombres entiers, pour les problèmes de majoration de temps de calcul pire cas (WCET) par analyse statique (Li & Malik, 1995).

6.3.3 Exemples élémentaires

Pour se fixer les idées, considérons l'exemple élémentaire de la figure 6.1(a) où chaque occurrence de la tâche A produit deux données sur le canal (A, B) et une donnée sur le canal (A, C), où chaque occurrence de la tâche B consomme une donnée sur le canal (A, B) et produit une donnée sur le canal (B, C) et où chaque occurrence de la tâche C consomme une donnée sur le canal (B, C) et une donnée sur le canal (A, C), tous les tampons étant dimensionnés à 3. Ces trois tâches sont donc cycliques et leurs machines à états sont limitées à une seule transition. Si l'on fait abstraction du dimensionnement, on se convainc aisément que ce réseau ne peut s'exécuter en mémoire bornée en raison de l'accumulation systématique sur le canal (A, B) d'une donnée non consommée pour chaque occurrence de la tâche C[4]. Dans le cas de ce réseau, CPLEX nous indique immédiatement que l'état spécifié par

$$n_{\tau(A)} = 5,\ n_{\tau(B)} = 8 \text{ et } n_{\tau(C)} = 5$$

est une solution du système (6.9) et donc, en conséquence, qu'il s'agit d'un état de deadlock potentiel. Ce qui s'avère être le cas. Si, par contre, on modifie la quantité produite par les occurrences de la tâche A sur le canal (A, B) à 1 alors le système devient incohérent, ce que CPLEX indique immédiatement, établissant ainsi l'absence de deadlock sur le réseau élémentaire résultant.



(a) (b)

FIGURE 6.1 – Cf. texte.

Considérons maintenant le réseau de la figure 6.1(b). Les conventions sont identiques à celles de l'exemple précédent. Par contre, la tâche S représente un switch (cf. chapitre 4) : chaque occurrence de cette tâche consomme une donnée sur le canal (A, S) et produit une donnée soit sur le canal (S, B) (premier choix)

4. Comme cet exemple rentre dans le cadre du modèle SDF de Lee & Messerschmitt (1987) on peut s'en convaincre en cherchant une solution non nulle pour le système linéaire formé des équations $n_{p_f}\mathsf{qp}_f - n_{c_f}\mathsf{qc}_f = 0$, $\forall f \in F$ (Galea & Sirdey, 2010) où les n_t donnent les nombres d'occurrences pour chacune des tâches pour ramener le réseau depuis un état initial prédéterminé (généralement l'état à vide) vers ce même état (dans le modèle SDF, les tâches n'ont qu'une transition et l'on peut donc simplifier les notations en identifiant les transitions aux tâches).

soit sur le canal (S, C) (second choix), de manière indéterminée (au niveau du modèle). En d'autres termes, les tâches A, B, C et D sont cycliques et représentées par des machines à états à une transition, alors que celle de la tâche S comporte deux transitions, le choix entre les deux se faisant de manière non déterministe. Abstraction faite du dimensionnement, il est clair que ce réseau ne peut s'exécuter systématiquement en mémoire bornée. En effet, la tâche S peut "choisir" indéfiniment de produire sur le canal (S, C) conduisant ainsi à une accumulation non bornée de données sur le canal (C, D). Dans le cas de ce réseau, CPLEX nous indique immédiatement que l'état spécifié par

$$n_{\tau^{(A)}} = 9,\ n_{\tau^{(S\to C)}} = 6,\ n_{\tau^{(B)}} = 0,\ n_{\tau^{(D)}} = 0 \text{ et } n_{\tau^{(C)}} = 3$$

est une solution du système (6.9), où $\tau^{(S\to C)}$ dénote la transition de S qui produit dans le canal (S, C). D'où un état de deadlock. En effet, A ne peut être exécutée car le tampon (A, S) est saturé, S (on suppose que S "veut" prendre son second choix) ne peut être exécutée car le tampon (S, C) est saturé et, enfin, C ne peut être exécutée car le tampon (C, D) est saturé. Ce qui se passe sur la branche associée au premier choix de S n'est pas pertinent au sens où si la tâche S "veut" faire son second choix alors le réseau est bel et bien bloqué.

D'autres exemples plus élaborés sont donnés dans Aubry (2009).

6.4 Exécution en mémoire bornée d'un DPN

6.4.1 Monotonie par rapport au dimensionnement

Comme l'ont montré Lee & Parks (1995), et comme nous l'avons déjà mentionné au chapitre 4, le formalisme DPN est équivalent au formalisme des réseaux de processus de Kahn (KPN). Il suit que les DPN possèdent la même propriété de déterminisme que les KPN : pour un jeu d'entrées fixé, les données qui circulent sur les canaux ne sont pas dépendantes de la trace d'exécution du réseau. Cette propriété fort pratique permet, pour un jeu d'entrées fixé, de dériver des propriétés générales sur le réseau à partir de propriétés exhibées par des traces d'exécution singulières.

Le formalisme des KPN se base sur des lectures bloquantes et des écritures non bloquantes sur les canaux, dans certains cas cela peut induire une exigence de mémoire infinie sur certains canaux. Néanmoins, un KPN K soumis à une contrainte de capacité sur ses canaux peut facilement être converti en un autre KPN $K(d)$ ($d \in \mathbb{Z}^{+n}$) : il suffit pour cela d'associer à chacun des canaux du réseau un canal de rétroaction et de modifier les processus de manière à ce qu'avant d'écrire dans un canal donné, un processus lise dans le canal de rétroaction associé, les canaux de rétroaction étant initialement munis d'une quantité de données non consommées égale à la capacité du canal correspondant. Bien entendu, les

propriétés du KPN non contraint ne sont généralement pas préservées par cette transformation et, en particulier, $K(d)$ peut ne pas être vivace alors que K l'est.

Par essence, la propriété de vivacité définie à la section précédente est suffisante pour garantir que, pour tout jeu d'entrées, une quantité de données non bornée circule sur au moins un des canaux du réseau. Partons de l'hypothèse que KPN $K(d)$ est vivace, alors une conséquence directe de la propriété de déterminisme des KPN est que tout KPN $K(d')$ avec $d' \geq d$ (i.e., $\forall f \in F$, $d'_f \geq d_f$) est aussi vivace. En effet, la vivacité de $K(d)$ implique que, pour tout jeu d'entrées α, toute trace d'exécution $\omega(\alpha)$ de $K(d)$ est telle qu'une quantité non bornée de données circule sur au moins un des canaux et $\omega(\alpha)$ est aussi une trace d'exécution valide pour $K(d')$. En conséquence par la propriété de déterminisme déjà évoquée, toutes les traces d'exécution de $K(d')$ sur le jeu d'entrées α sont aussi telles qu'une quantité non bornée de données circule sur au moins un des canaux du réseau. Il suit que $K(d')$ est vivace.

Donc, étant donné un DPN, si l'on peut trouver $d \in \mathbb{Z}^{|F|}$ tel que $\{x : Ax \leq b(d), x \in \mathbb{Z}^n\} = \emptyset$ ou $\{x : Ax \leq b(d), x \in \mathbb{R}^n\} = \emptyset$, alors pour tout $d' \in \mathbb{Z}^{|F|}$ tel que $d' \geq d$, le DPN en question est vivace.

6.4.2 Exécution bornable en mémoire

Repartons du système en nombres entiers (6.9), posons $d_f = z$ ($\forall f \in F$) et considérons le programme linéaire en nombres entiers suivant :

$$\begin{cases} z_{\text{IP}} = \text{Maximiser } z \\ \text{s. l. c.} \\ A'y \leq b', \\ y \in \mathbb{Z}^{n+1}, \end{cases} \tag{6.10}$$

où le vecteur y est obtenu par concaténation du vecteur x et du scalaire z. Ce programme se dérive directement du système (6.9) en remplaçant d_f par z dans les contraintes (6.2) ainsi que dans les contraintes (6.5) ou (6.8) (selon lesquelles des deux s'appliquent) et en déplaçant z dans le membre gauche.

En conséquence de la propriété de monotonie établie à la section 6.4.1, une solution du programme ci-dessus fourni *la plus grande valeur de z telle que le réseau n'est pas vivace* et, par conséquent, pour tout vecteur de dimensionnement $d \in \mathbb{Z}^{|F|}$ tel que $\forall f \in F$, $d_f \geq z_{\text{IP}}$ le réseau est garanti vivace. Trois cas peuvent alors se présenter. Cas 1 : le programme n'admet pas de solution, un cas dégénéré qui ne se présente que dans le cas d'un réseau sans canaux (puisque, par la contrainte (6.1), il n'y a pas de transition sans effet). On ignore donc ce cas. Cas 2 : $z_{\text{IP}} < \infty$, ce qui est suffisant pour établir que le réseau est vivace et bornable en mémoire. Cas 3 : $z_{\text{IP}} = \infty$ cas dans lequel on ne peut conclure quant à ces deux propriétés. De plus, quand $z_{\text{IP}} < \infty$, poser $d_f = z_{\text{IP}} + 1$ ($\forall f \in F$) donne un dimensionnement (présumé petit) des canaux garantissant la vivacité.

De nouveau, il est possible de considérer la relaxation continue z_{LP} du programme (6.10). En particulier, lorsque l'on cherche uniquement à déterminer si $z_{\text{IP}} < \infty$, c'est-à-dire si le réseau peut être exécuté en mémoire bornée, alors il est nécessaire et suffisant de déterminer si $z_{\text{LP}} < \infty$ car $z_{\text{IP}} < \infty \Leftrightarrow z_{\text{LP}} < \infty$. En effet, un résultat bien connu de la théorie des polyèdres (Nemhauser & Wolsey, 1999) indique que si l'enveloppe entière P_I d'un polyèdre rationnel P (i.e., l'enveloppe convexe des vecteurs à valeurs entières de P) est non vide alors $\max\{cx : x \in P\}$ est borné si et seulement si $\max\{cx : x \in P_I\}$ l'est aussi. Dans la mesure où (trivialement) $\emptyset \neq \{y : A'y \leq b', y \in \mathbb{Z}^{n+1}\}$, ce résultat s'applique au programme (6.10) et à sa relaxation. Il suit que $z_{\text{LP}} < \infty$ fournit une condition suffisante polynomiale permettant d'établir l'exécution sans deadlock, bornable en mémoire, d'un DPN qui est équivalente à $z_{\text{IP}} < \infty$.

À noter que la présente propriété d'exécution bornable en mémoire n'est pas analogue à la propriété de bornitude (boundedness) d'un réseau de Petri. Cette dernière propriété requiert en effet qu'il ne soit en aucun cas possible d'accumuler une quantité non bornée de jetons dans une place, et ce de manière intrinsèque au couple réseau/marquage initial. Au contraire, ici, *aucun dimensionnement n'étant a priori donné*, on cherche s'il existe un dimensionnement fini permettant de contraindre l'exécution en mémoire bornée sans pour autant invalider la vivacité du réseau. En ce sens, la propriété de bornitude de la théorie des réseaux de Petri est une condition suffisante mais non nécessaire pour l'exécution bornable en mémoire dans le sens où nous l'entendons ici.

6.4.3 Remarques sur les aspects algorithmiques

Bien que z_{LP} peut être calculé en temps polynomial, il est à prévoir, lorsque $z_{\text{LP}} < \infty$, que l'écart, $z_{\text{LP}} - z_{\text{IP}}$, puisse être important. À titre d'exemple, pour le réseau de la figure 6.1(a) où toutes les productions et consommations seraient à 3, on obtient $z_{\text{LP}} = 6$ (versus $z_{\text{IP}} = 2$), donc une borne de bon fonctionnement de 7 (versus 3). Remarquons néanmoins que sur cet exemple, les tailles minimums de tampons permettant de faire fonctionner toutes les tâches en parallèle sont de 6, 6 et 9 pour les canaux (A, B), (B, C) et (A, C), respectivement.

Lorsque l'on cherche à obtenir une borne serrée sur la taille des canaux il devient nécessaire de résoudre le programme en nombres entiers (6.10), ou tout au moins de renforcer sa relaxation. Il convient alors de souligner que le polyèdre $P_I = \{y : A'y \leq b', y \in \mathbb{Z}^{n+1}\}$ (rappelons que l'enveloppe entière d'un polyèdre est également un polyèdre) n'est généralement pas un polytope (puisque les n_τ ne sont pas nécessairement bornés). En conséquence, dans le cas général, un algorithme qui procéderait par énumération des points entiers à l'intérieur du polyèdre, comme c'est le cas de la majeure partie des solveurs sur étagère, peut ne pas terminer. Pour garantir la terminaison, il conviendrait donc d'utiliser une approche par plans coupants, à l'image de l'algorithme des coupes de Gomory qui fournirait dans ce cas une garantie de terminaison (après un temps prohibi-

tivement long néanmoins).

6.5 Discussion

Nous avons donc présenté, d'une part, une condition suffisante de vivacité pour un DPN dimensionné et, d'autre part, une méthode suffisante pour trouver un dimensionnement fini et sûr des tampons, en temps polynomial, lorsqu'un tel dimensionnement existe (au sens où il existe des réseaux bornables en mémoire que notre méthode considérera ne pas avoir cette propriété mais la réciproque est fausse). Ces deux approches peuvent être combinées afin de donner une méthode de dimensionnement : on commence tout d'abord par résoudre la relaxation linéaire du programme (6.10), afin d'obtenir une borne de bon fonctionnement $d^{\max} = \lceil z_{\mathrm{LP}} \rceil$ (ou $z_{\mathrm{LP}} + 1$ lorsque z_{LP} est entier) sur la taille des canaux du DPN (si $d^{\max} = \infty$ alors stop), puis, sur la base d'un ordre lexicographique sur les canaux, on procède à la recherche par dichotomie d'une dimension critique canal par canal. Plus précisément, soit F_0 l'ensemble des canaux déjà dimensionnés et soit f_0 le prochain (dans l'ordre lexicographique) canal que l'on cherche à dimensionner, on procède par dichotomie sur $d_0 \in \{1, \ldots, d^{\max}\}$ par résolutions successives de la relaxation linéaire du système (6.9), avec

$$d_f = \begin{cases} d_f & \text{si } f \in F_0, \\ d_0 & \text{si } f = f_0, \\ d^{\max} & \text{sinon.} \end{cases}$$

Dans une logique d'algorithme à démarrages multiples, on pourra exécuter cet algorithme sur plusieurs ordres lexicographiques et conserver le dimensionnement le plus adapté au sens d'une fonction économique, par exemple l'empreinte mémoire totale,

$$\sum_{f \in F} d_f.$$

Il n'aura pas échappé au lecteur que ces travaux sont destinés à des modèles de calcul plus généraux que celui sous-jacent au langage ΣC. Ceci dans la mesure où ce dernier, comme nous l'avons vu au chapitre 4, est bien moins général que celui des DPN. En l'occurrence, les outils d'analyse formelle présentés dans ce chapitre ont initialement été pensés pour le modèle de calcul sous-jacent à un autre langage que nous avons également contribuer à concevoir, le langage τC (Goubier et al., 2008b), qui n'est autre que le modèle DPN. Ce langage, qui avait initialement été proposé dans le cadre du projet Systématic Téraops, est en quelque sorte le dual du langage ΣC : plus puissant sur le plan de l'expressivité mais plus délicat à maîtriser sur le plan formel et à compiler efficacement.

Chapitre 7

Ordonnancement pour le préchargement des données

7.1 Introduction

Ce chapitre résume succinctement les travaux réalisés dans le cadre de la thèse de Sergiu Carpov (Carpov, 2011) que nous avons co-encadrée avec Jacques Carlier et Dritan Nace. Dans cette thèse, Sergiu s'est attaché à modéliser et à résoudre un certain nombre de problèmes d'ordonnancement d'actions de préchargement de données depuis un niveau d'une hiérarchie mémoire vers un niveau plus bas à accès plus rapide (typiquement, depuis une mémoire externe vers la mémoire interne des clusters d'une architecture du type celle évoquée au chapitre 4), ainsi que des tâches de calcul associées à ces données. Nous avons également donné une dimension spéculative à certains de ces problèmes afin de tenir compte de structures de branchement conditionnel, en particulier dans les réseaux de processus flots de données. Le préchargement est l'action de chargement des données avant d'en avoir effectivement besoin. Le préchargement devient spéculatif si cette action est réalisée avant de savoir si l'on aura effectivement besoin des données.

Sergiu a soutenu sa thèse le 14 octobre 2011. Depuis le 20 octobre 2011, il est ingénieur-chercheur au CEA.

7.2 Étude d'une structure de branchement élémentaire

Afin de commencer par des problèmes relativement épurés, nous avons étudié les stratégies de préchargement optimales pour une structure de branchement conditionnel flot de données à n sorties (un switch au sens du chapitre 4) relativement à plusieurs fonctions économiques en particulier l'espérance mathématique du temps d'exécution et le temps pire cas.

Ces travaux ont fait l'objet d'un article dans les actes de l'*International Symposium on Combinatorial Optimization* (Carpov et al., 2010b).

7.2.1 Modèle

On considère donc ici une structure de branchement élémentaire à n sorties telle que représentée à la figure 7.1. Une telle structure possède un canal d'entrée, un canal de contrôle et n canaux de sortie. Lorsqu'une donnée se présente sur le canal d'entrée, elle est (fonctionnellement) reproduite sur l'unique canal de sortie spécifié par le canal de contrôle et, ce faisant, la tâche de traitement connectée à ce canal est activée.

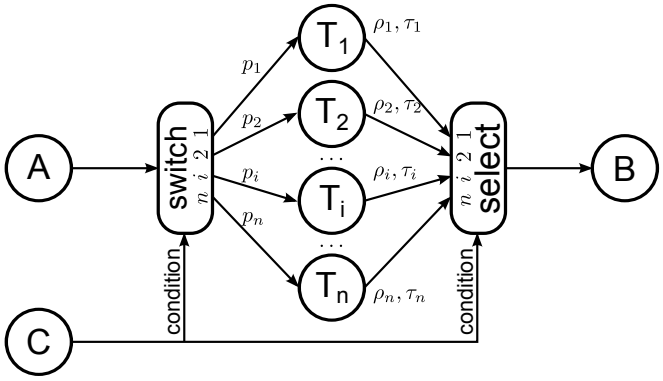


FIGURE 7.1 – Structure de branchement conditionnel à n sorties.

7.2.2 Minimisation de l'espérance du temps d'exécution

On note ρ_i le temps de chargement des données desquelles dépendent l'exécution de la tâche T_i et τ_i le temps de d'exécution de cette tâche. Soit p_i la probabilité de sélection de la i-ème branche [1] et soit T le temps de préchargement. Pour ce qui est du critère d'espérance mathématique du temps d'exécution, nous avons montré qu'une stratégie de préchargement optimale s'obtient par simple résolution d'un problème de sac à dos continu et consiste à précharger les branches dans l'ordre décroissant des p_i. Ce résultat, bien qu'élémentaire, n'est pas dépourvu d'intérêt. En effet, il nous donne, pour ce premier critère, une stratégie de préchargement optimale, non préemptive (les données des branches préchargées le sont intégralement, exception faite possiblement de la dernière) et monotone par rapport au temps de préchargement T (qui est en général une variable aléatoire).

1. Supposer connues ces probabilités n'est pas déraisonnable dans un contexte de compilation itérative (cf. chapitre 4).

7.2.3 Minimisation du temps d'exécution pire cas

Dans le cas du temps d'exécution pire cas, nous nous sommes également restreint, en raison de leurs bonnes propriétés, à des stratégies non préemptives et monotones [2]. Nous avons donc chercher à trouver un ordre σ sur les branches qui minimise

$$\int_{\mathcal{D}} f_\sigma(T)dT$$

où $\mathcal{D}$ est le support du temps de préchargement et où $f_\sigma(T)$ donne le temps d'exécution pire cas induit par l'ordre σ, lorsque le temps de préchargement est T. Posé en ces termes, ce problème revient à minimiser l'espérance mathématique du pire temps d'exécution sous l'hypothèse que le temps de préchargement est uniformément réparti sur $\mathcal{D}$. En supposant que les branches sont ordonnées par $\tau_i + \rho_i$ décroissant et en utilisant les propriétés de monotonie des fonctions f_σ, nous avons pu établir une règle de dominance permettant de ramener le problème à la recherche d'un plus court chemin dans un graphe particulier à $n+1$ sommets dont la structure est illustrée à la figure 7.2 pour une structure de branchement à 4 sorties.

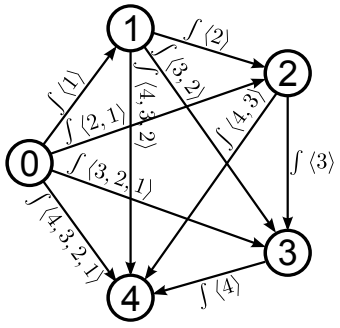


FIGURE 7.2 – Voir texte.

7.3 Problèmes de flowshop hybride

Afin de faire le lien entre ces travaux et le modèle d'exécution cadencé par un ordre partiel présenté à la section 4.3, nous avons introduit un modèle de flowshop hybride prenant en compte des contraintes de précédence.

Ces travaux font l'objet d'un article à paraître dans le journal *Computers & Operations Research* (Carpov et al., 2012).

2. Par contre, on se convainc aisément que de telles stratégies ne sont pas nécessairement optimales pour le temps d'exécution pire cas.

7.3.1 Modèle

Notre modèle de flowshop hybride consiste en la donnée d'un ensemble de n travaux décomposés en deux opérations. Pour le travail i, la première étape consiste à réaliser la première opération (de durée a_i) sur une unique machine puis la seconde étape consiste à réaliser la seconde opération (de durée b_i), de manière non préemptive, sur l'une quelconque de m machines parallèles identiques. Il n'y a pas de contraintes d'ordonnancement sur les opérations de la première étape. Il y a par contre un ensemble de contraintes de précédence, spécifiées par un graphe orienté sans circuit, entre les opérations de la seconde étape. Pour revenir au modèle d'exécution de la section 4.3, les travaux sont les occurrences des tâches qui sont divisées en deux opérations (chargement des données externes puis calcul proprement dit), l'unique machine de la première étape représente un moteur DMA et les m machines de la seconde étape, les processeurs de calcul.

Nous nous sommes intéressés à la minimisation du temps total d'exécution (makespan) dans le cas de deux variantes de ce modèle, selon que l'on autorise (variante avec attente) ou pas (variante sans attente) la possibilité d'un temps mort entre la fin de l'exécution de la première opération d'une tâche et le début de l'exécution de la seconde.

7.3.2 Bornes inférieures

Pour ce problème, nous avons introduit plusieurs bornes inférieures globales, dans l'optique de pouvoir les coupler à des algorithmes de résolution approchée afin d'obtenir des évaluations par excès de distances à l'optimum. Une première borne triviale est donnée par

$$\max\left(\min_{i=1,\ldots,n} a_i + \frac{1}{m}\sum_{i=1}^{n} b_i, \sum_{i=1}^{n} a_i + \min_{i=1,\ldots,n} b_i\right).$$

Dans un premier temps, Nous avons amélioré cette borne en adaptant deux bornes proposées par Gupta (1988) pour le problème de flowshop hybrique sans contraintes de précédence, justement en tirant parti de ces contraintes. Le principe de la première de ces bornes, GLB_1^1, repose sur une minoration du temps mort cumulé (nécessairement strictement positif) sur les m machines parallèles pendant l'exécution des $m+1$ premiers travaux [3], suivant l'ordre topologique sur le graphe de dépendances, induit par la sérialisation des opérations sur la première machine (cf. figure 7.3(a)). Le principe de la seconde de ces bornes, GLB_1^2, repose sur une minoration de la plus grande durée résiduelle de travail (nécessairement strictement positive) induite sur les m machines parallèles par l'exécution des m derniers travaux, suivant l'ordre topologique sur le graphe de dépendances, une fois toutes les premières opérations réalisées sur la première machine (cf. figure

3. On suppose $n >> m$.

7.3(b)). Une première borne inférieure globale, GLB_1, s'obtient alors en prenant le maximum de ces deux bornes.

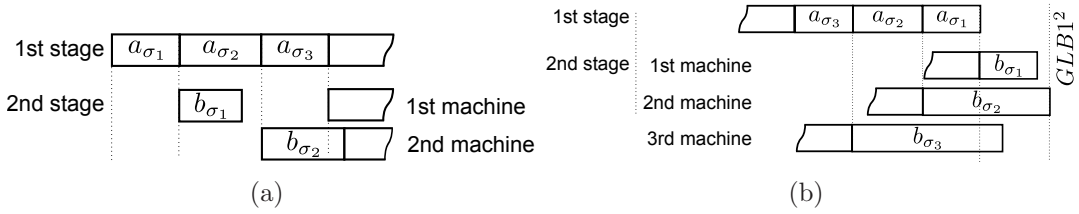


FIGURE 7.3 – Cf. texte.

Dans la veine des travaux de Carlier & Pinson (1994), nous avons défini une autre borne basée sur les dates de disponibilité (heads) et les durées de latence (tails) des opérations. Soit $r_i^{(1)}$ (respectivement $r_i^{(2)}$) la date de disponibilité de la première (respectivement de la seconde) opération du i-ème travail et soit $q_i^{(1)}$ (respectivement $q_i^{(2)}$) la durée de latence pour la première (respectivement pour la seconde) opération du i-ème travail alors on a la borne,

$$\mathcal{L} = \max_{i=1,\ldots,n} (r_i^{(2)} + b_i + q_i^{(2)}).$$

Sur la base d'un ensemble de contraintes de cohérence imposées sur les $r_i^{(x)}$ et sur les $q_i^{(x)}$ (contraintes de précédence intra-travaux, contraintes de précédence inter-travaux, contraintes imposées par des relaxations, etc.) nous avons construit une borne, GLB_2, consistant à trouver, par dichotomie, la plus grande valeur $\mathcal{L}$ compatible avec ces contraintes. Nous avons également commencé à généraliser ces travaux sur l'ajustement des dates de disponibilité et des durées de latence dans le cas stochastique (Carpov et al., 2011a).

Dans nos expérimentations (Carpov et al., 2012), sur les deux variantes susmentionnées, GLB_2 domine GLB_1 dans 60 à 70% des cas et GLB_1 domine GLB_2 dans environ 10% des cas. Les deux bornes ont donc un intérêt pratique.

7.3.3 Algorithme de résolution

En complément des bornes inférieures globales présentées à la section précédente, et afin d'obtenir des solutions, nous avons conçu un algorithme de liste randomisé. Cet algorithme se base sur deux règles de priorité. La première correspond à la règle CP/MISF (Critical Path/Most Immediate Successor First) de Kasahara & Narita (1984) la seconde est une variante de la règle ETF (Earliest Time First) de Hwang et al. (1989).

Le schéma de randomisation de cet algorithme est une adaptation de celui proposé par Hart & Shogan (1987) pour les algorithmes gloutons et couramment utilisé dans les algorithmes de type GRASP (Sirdey et al., 2010). Soit R l'ensemble

des jobs prêts et soit $p_{\min} = \min_{j \in R} p_j$ et $p_{\max} = \max_{j \in R} p_j$, alors au lieu de choisir systématiquement le job $\text{argmax}_{j \in R} p_j$, on tire le prochain job uniformément dans l'ensemble

$$\{j \in R : p_j \in [p_{\max} - \alpha(p_{\max} - p_{\min}), p_{\max}]\}$$

avec $\alpha \in [0, 1]$. Lorsque $\alpha = 0$ on a un algorithme de liste (avec départage des égaux aléatoire), lorsque $\alpha = 1$ on a un algorithme aléatoire. Bien entendu, il faut choisir une valeur pour le paramètre α. Dans la veine des stratégies de GRASP dites réactives (Prais & Ribeiro, 2000), nous ajustons la valeur du paramètre α en fonction de statistiques sur la qualité des solutions engendrées. Par contre, sans effets expérimentaux significatifs en défaveur de cette approche, nous avons simplement procédé en deux phases. Une première phase collecte des statistiques sur les solutions sur un ensemble fini, $\mathcal{A}$, de valeurs de α (typiquement l'intervalle $[0, 1]$ discrétisé avec un pas constant) afin de construire une loi de probabilité de la forme

$$\pi_{\alpha_0} = \frac{S_{\max} - S_{\alpha_0}}{\sum_{\alpha' \in \mathcal{A}} (S_{\max} - S_{\alpha'})} \tag{7.1}$$

où $S_{\max}$ est la valeur de la meilleure solution obtenue durant la première phase (toutes valeurs de α confondues) et où S_{α_0} est la meilleure solution obtenue, pendant cette même phase, avec $\alpha = \alpha_0$. La loi de probabilité (7.1) est alors utilisée, dans une seconde phase, pour répartir un budget d'itérations entre les différentes valeurs de α selon la règle

$$N_{\alpha_0} = \pi_{\alpha_0} N$$

où N est le nombre total d'itérations de la seconde phase, et où N_{α_0} est le nombre de ces itérations qui seront réalisées avec $\alpha = \alpha_0$ (la moitié étant réalisées avec la première règle de priorité, l'autre moitié avec la seconde).

Dans nos expérimentations (Carpov et al., 2012), nous pu mettre en évidence que cet algorithme, associé aux bornes de la section 7.3.2, fournissait des solutions prouvées être à moins de (de l'ordre de) 2% de l'optimum dans le cas de la variante avec attente et à moins de (de l'ordre de) 5% de l'optimum dans le cas sans attente. Ce travail et ces résultats sont tout à fait en phase avec notre philosophie de résolution pratique des problèmes combinatoires difficiles : associer une heuristique adaptée à une borne inférieure globale de bonne qualité afin de majorer aussi peu conservativement que possible la distance à l'optimum [4]. Dans ce contexte, l'utilisation de la borne ne se résume pas à une passe de validation expérimentale, cette dernière fait partie intégrante de l'algorithme de résolution.

4. C'est déjà ce que nous avions fait dans notre thèse de doctorat, en associant une borne polyèdrale et un algorithme de recuit (Sirdey, 2007c), cf. chapitre 2.

7.4 Gestion de la mémoire pour un DAG

Dans le cadre de la thèse de Sergiu, certes un peu en marge du sujet principal mais en continuation de ses travaux de stage de Master Recherche, nous avons également travaillé sur un problème d'estimation du degré de parallélisme d'une application, supposée fortement parallèle et modélisée par un graphe orienté sans circuit (DAG) dont les sommets représentent des calculs et les arcs des dépendances, induit par une contrainte de bande passante mémoire. Cette problématique a donné lieu à des problèmes combinatoires NP-difficiles originaux pour lesquels nous avons exhibé des cas particuliers polynomiaux et conçu des algorithmes de résolution exacte (branch-and-bound) et approchée (heuristiques ad hoc)

Ces travaux ont fait l'objet d'un article dans les actes de la 17ème *International Computing and Combinatorics Conference* (Carpov et al., 2011b) ainsi que d'une présentation au *workshop on Combinatorial Optimization for Embedded System Design* de la 7ème *International Conference on Integration of Artificial Intelligence and Operations Research techniques in Constraint Programming* (Carpov et al., 2010a).

7.5 Discussion

Il convient de souligner que les travaux de thèse de Sergiu ont vocation à être valorisés dans le cadre de la chaîne de compilation du langage ΣC (cf. chapitre 4). En particulier, soulignons que le problème de flowshop hybride étudié (section 7.3) représente fidèlement le modèle d'exécution à cadencement par un temps logique vectoriel exposé à la section 4.3. Il trouvera donc une pertinence pratique directe lorsqu'il s'agira de permettre l'exécution d'applications dont l'empreinte mémoire est trop importante pour la plateforme, en particulier pour ce qui est des données supports des tâches (code, constantes, etc.), un cas qui est loin d'être théorique (nous pensons notamment à certains algorithmes de reconnaissance des formes, à la radio cognitive ainsi qu'aux standards d'encodage vidéo de prochaine génération). Dans ce dernier contexte, il en est également de même pour les travaux résumés à la section 7.2 sur une structure de branchement élémentaire. Ceci dans la mesure où même si l'on a doté, dans le langage ΣC, ces structures d'une sémantique régularisée pour ce qui est des quantités produites et consommées dans les canaux, les tâches restent conditionnellement exécutées et leurs données supports n'ont donc pas à être systématiquement chargées. Il reste néanmoins à définir un cadre pratique permettant d'articuler ces deux approches.

Travaux en cours et perspectives

En termes de perspectives, il est tout d'abord pertinent d'indiquer que le domaine de l'embarqué nous semble être une source intarissable de problèmes combinatoires variés et intéressants.

Ce mémoire ne pouvait pas être exhaustif. À titre d'exemple, nous avons récemment travaillé avec Thomas Megel sur l'optimisation du nombre de préemptions et de migrations de tâches induit dans les ordonnancements temps réel préemptifs sur multiprocesseur symétrique. Cette approche combine une partie hors-ligne basée sur la résolution d'un programme linéaire en nombres entiers adéquat [5] et une partie en-ligne basée sur une technique de commutation de tâches originale. Ces travaux ont fait l'objet d'un article dans les actes du *31st IEEE Real-Time Systems Symposium* (Megel et al., 2010). Dans la même veine, avec Kods Trabelsi et Mathieu Jan, chercheurs au sein de mon laboratoire d'appartenance, nous travaillons actuellement sur l'optimisation de la consommation induite dans de tels ordonnancements préemptifs. À l'image des travaux précédemment cités ci-dessus, notre approche comprend une partie hors-ligne qui consiste à utiliser la solution d'un programme linéaire en nombres entiers adéquat pour piloter finement des fonctionnalités processeur de type variation de fréquence et autre mode IDLE. Ces travaux sont en cours de publication.

De même, les modèles et les algorithmes de résolution des problèmes de placement/routage présentés au chapitre 5 devront être étendus d'une part pour tenir compte d'architectures "multimulticœurs" à plusieurs niveaux de hiérarchie (cartes multipuces, clusters de cartes multipuces) et ainsi généraliser nos outils d'optimisation au contexte du supercalcul embarqué voire du supercalcul tout court. Ceci aura des conséquences en termes de complexité des modèles d'une part et en termes de taille des instances. De la même manière, ces mêmes modèles et algorithmes de résolution vont aussi devoir être étendus pour tenir compte des problèmes de rendement lithographique et, plus généralement, de tolérance aux pannes par génération hors-ligne de plans de reconfiguration (migrations de tâches, reroutages) associés à des scénarios de défaillances (niveau cluster voire plus bas). Il est d'ailleurs à prévoir que nos travaux de thèse, résumés au chapitre 2, trouvent, dans ce contexte, de nouvelles applications. Des premiers travaux

5. NP-difficile par restriction à 3-partition mais raisonnablement traitable pour les tailles d'instances pratiques à l'aide de solveurs sur étagère, CPLEX notamment.

en ce sens ont été réalisés en collaboration avec Thomas Megel et sont en cours de publication. D'autres travaux sont à prévoir dans le domaine de la compilation parallèle en matière de dimensionnement des tampons associés aux canaux, d'optimisation du cadencement, de prise en compte d'aspects stochastiques (cf. paragraphe suivant), etc. Avec à la clef, des perspectives d'industrialisation quasi-immédiates.

Un autre axe de perspectives concerne la programmation stochastique et l'optimisation robuste. En effet, comme nous l'avons déjà évoqué à la section 5.3, les problèmes rencontrés dans le domaine de l'embarqué ont une caractéristique commune : la présence de données intrinsèquement incertaines, tels, par exemple, des temps d'exécution ou des latences réseau, dans leurs énoncés. Les techniques d'optimisation robustes ou sous contraintes probabilistes trouvent là un champ d'application privilégié que nous avons commencé à explorer, dans un premier temps, via le problème de partitionnement stochastique de graphe présenté dans ce mémoire. Ce chantier est également adressé dans le cadre de la thèse d'Oana Stan qui a débutée en novembre 2010. Il s'agit d'une thèse dont le spectre applicatif est assez large et dont l'objet est d'aborder une large gamme des problèmes d'optimisation posés dans le contexte de la compilation pour l'embarqué sous l'angle de la programmation stochastique afin de proposer des modèles adéquats, des méthodes de résolution opérationnelles ainsi que d'étudier l'apport effectif de ces techniques au domaine : meilleure maîtrise de la fiabilité (et non de la sûreté) de systèmes temps réel critiques, meilleure maîtrise du dimensionnement de systèmes temps réel non critiques, estimation du degré de robustesse d'un système, etc. Les travaux réalisés dans ce contexte ont déjà fait l'objet d'un premier article de positionnement dans les actes de l'École d'Été Temps Réel 2011 (Stan et al., 2011).

Cette exigence de robustesse, ainsi que l'avènement des stations de travail multicœurs, rend incontournable le développement d'algorithmes parallèles pour la résolution exacte ou approchée et *opérationnelle* de ces problèmes. Dans le cadre du stage de Benoît Maurin, nous avons pu commencer à tester quelques idées de parallélisation de codes d'optimisation, que ce soit en résolution approchée (recuit simulé parallèle avec spéculation sur les rejets, notamment), en résolution exacte (branch-and-bound) ou en prise en compte d'aspects stochastique (en cherchant à atténuer l'augmentation de la complexité des modèles par recours au parallélisme), en allant jusqu'à l'implémentation et l'exécution sur un calculateur à 48 cœurs avec mémoire unifiée (Maurin & Sirdey, 2011). Cet axe de recherche fera vraisemblablement l'objet d'une thèse qui démarrera fin 2012.

Depuis fin 2010, je renoue (officiellement) avec la cryptologie et la sécurité des systèmes d'informations[6]. Je m'intéresse en particulier aux systèmes de chif-

6. Domaine dans lequel j'ai été particulièrement actif en début de carrière par des travaux industriels sur les algorithmes d'authentification, de génération de clefs et d'aléas pour les standards GSM et UMTS puis, autour de 2007, par la conception d'infrastructures sécuritaires permettant de connecter une BTS à son BSC via un réseau paquet non sécurisé (notamment

frement homomorphiques, aux circuits de Yao et, plus généralement, aux cryptosystèmes permettant de réaliser des calculs sur les chiffrés ayant un sens dans le domaine des clairs ainsi qu'aux problèmes posés par la mise en œuvre efficace de schémas de ce type sur des cas applicatifs concrets (e.g., pattern matching dans des textes chiffrés, requêtes "à la SQL" dans des bases de données chiffrées, inspection de paquets à critères masqués, etc.). Ces activités de recherche monteront en puissance durant l'année 2011 notamment par l'encadrement du stage de Master Recherche puis de la thèse de Simon Fau, consacrés à ces techniques, cette dernière devant commencer à l'automne 2011. Également, en décembre 2010, j'ébauche une infrastructure système complète et pratique d'authentification et, surtout, de sécurisation de l'exécution adaptée à la couche d'Infrastructure-as-a-Service (IaaS) du modèle de Cloud Computing (Sirdey, 2010). Sur le plan de la confidentialité, cette infrastructure est robuste contre toute compromission logicielle, y compris lorsque cette compromission touche le niveau de l'hypervision, le plus bas et le plus préjudiciable. Ces travaux servent de base structurante à un projet interne CEA visant à prototyper un support d'exécution traitant des flux chiffrés d'instructions et de données (Sirdey, 2011) et dont je me suis vu confié la responsabilité.

Autant dire, en guise de conclusion, qu'entre parallélisme, compilation, optimisation et cryptologie, ces prochaines années ne devraient pas être moins passionnantes que les précédentes.

dans le contexte du chapitre 3).

Bibliographie

3GPP. « Digital cellular telecommunications system (Phase 2+) — Radio subsystem synchronization ». Rapport technique ETSI TS 145 010 V4.0.0 (2001-04), European Telecommunications Standards Institute, 2001.

R. K. Ahuja, T. L. Magnanti et J. B. Orlin. *Network flows. Theory, algorithms and applications*. Prentice Hall, 1993.

P. Aubry. « Application de la programmation linéaire à la vérification de propriétés de sûreté dans le modèle des flots de données général ». Rapport technique DTSI/SARC/09-289, Commissariat à l'Énergie Atomique, 2009.

J. Aweya, D. Y. Montuno, M. Ouellette et K. Felske. « Clock recovery based on packet inter-arrival time averaging ». *Computer Communications*, 29:1696–1709, 2006.

M. L. Balinski et H. P. Young. *Fair representation: meeting the ideal of one man, one vote*. Brookings Institution Press, 2001.

J. Bi, Q. Wu et Z. Li. « On estimating clock skew for one-way measurements ». *Computer Communications*, 29:1213–1225, 2006.

C.-E. Bichot et P. Siarry, éditeurs. *Partitionnement de graphes. Optimisation et applications*. Lavoisier, 2010.

A. Billionnet. *Optimisation discrète. De la modélisation à la résolution par des logiciels de programmation mathématique*. Dunod, 2007.

G. Bilsen, M. Engels, R. Lauwereins et J. Paperstraete. « Cyclo-static dataflow ». *IEEE Transactions on Signal Processing*, 44(2):397–408, 1996.

M. Boulle. « Compact mathematical formulation for graph partitioning ». *Optimization and Engineering*, 5:315–333, 2004.

G. W. Brams. *Réseaux de Petri : théorie et pratique. Tome 1 : théorie et analyse*. Masson, 1983.

J. T. Buck. *Scheduling dynamic dataflow graphs with bounded memory using the token flow model*. Thèse de doctorat, Université de Berkeley, 1993.

R. E. Burkard et U. Fincke. « Probabilistic asymptotic properties of some combinatorial optimization problems ». *Discrete Applied Mathematics*, 12: 21–29, 1985.

J. Carlier. *Problèmes d'ordonnancement à contraintes de ressources : algorithmes et complexité*. Thèse d'état, 1984.

J. Carlier et E. Pinson. « Adjustements of heads and tails for the job-shop problem ». *European Journal of Operational Research*, 78:146–161, 1994.

S. Carpov. *Ordonnancement pour la gestion de la mémoire et du préchargement dans les microprocesseurs multicœurs pour l'embarqué*. Thèse de doctorat, Université de Technologie de Compiègne, 2011.

S. Carpov, J. Carlier, D. Nace et R. Sirdey. « Probabilistic parameters of conditional task graphs ». Dans *Proceedings of the 14th IEEE International Conference on Network-Based Information Systems*, pages 376–381, 2011a.

S. Carpov, J. Carlier, D. Nace et R. Sirdey. « Task ordering and memory management problem for degree of parallelism estimation ». Dans *Lecture Notes in Computer Science*, tome 6842, pages 592–603, 2011b.

S. Carpov, J. Carlier, D. Nace et R. Sirdey. « Two-stage hybrid flow shop with precedence constraints and parallel machines at second stage ». *Computers & Operations Research*, 39:736–745, 2012.

S. Carpov, R. Sirdey, J. Carlier et D. Nace. « Memory bandwidth-constrained parallelism dimensioning for embedded many-core microprocessors ». Dans *CPAIOR10 workshop on Combinatorial Optimization for Embedded System Design*, 2010a.

S. Carpov, R. Sirdey, J. Carlier et D. Nace. « Speculative data prefetching for branching structures in dataflow programs ». *Electronic Notes on Discrete Mathematics*, 36:119–126, 2010b.

B. Charron-Bost. « Concerning the size of logical clocks in distributed systems ». *Information Processing Letters*, 39(1):11–16, 1991.

J.-P. Cocquerez et S. Philipp, éditeurs. *Analyse d'image : filtrage et segmentation*. Masson, 1995.

S. Coles. *An introduction to statistical modeling of extreme values*. Springer, 2001.

B. J. Copeland, éditeur. *Colossus. The secrets of Bletchley Park codebreaking computer*. Oxford University Press, 2006.

B. A. Davey et H. A. Priestley. *Introduction to lattices and order*. Cambridge University Press, 2002.

V. David, C. Fraboul, J.-Y. Rousselot et P. Siron. « Étude et réalisation d'une architecture modulaire et reconfigurable : projet MODULOR ». Rapport technique 1/3364/DERI, ONERA, 1991.

V. David et R. Sirdey. « Mémoire cache segmentée ». Brevet, n° de dépot 1001494, 2010.

L. de Haan et A. Ferreira. *Extreme value theory. An introduction*. Springer, 2006.

P. Dell'Olmo, H. Kellerer, M. G. Speranza et Z. Tuza. « A $\frac{13}{12}$ approximation algorithm for bin packing with extendable bins ». *Information Processing Letters*, 65:229–233, 1998.

P. Dell'Olmo et M. G. Speranza. « Approximation algorithms for partitioning small items in unequal bins to minimize the total size ». *Discrete Applied Mathematics*, 94:181–191, 1999.

M. Demange et V. T. Paschos. « On an approximation measure founded on the links between optimization and polynomial approximation theory ». *Theoretical Computer Science*, 158:117–141, 1996.

M. Diaz, éditeur. *Les réseaux de Petri. Modèles fondamentaux*. Hermès, 2001.

P. Dubrulle et R. Sirdey. « Génération d'un runtime de cadencement par un temps logique vectoriel ». Rapport technique DTSI/SARC/09-138, Commissariat à l'Énergie Atomique, 2009.

A. Duda, G. Harrus, Y. Haddad et G. Bernard. « Estimating global time in distributed systems ». Dans *Proceedings of the 7th IEEE International Conference on Distributed Computing Systems*, pages 299–306, 1987.

M. E. Dyers. « Linear time algorithms for two- and three-variable linear programs ». *SIAM Journal of Computing*, 13:31–45, 1983.

J. Esparza et M. Nielsen. « Decidability issues for Petri nets: a survey ». *Journal of Information Processing and Cybernetics*, 30:143–160, 1994.

T. A. Feo et M. G. C. Resende. « A probabilistic heuristic for a computationally difficult set covering problem ». *Operations Research Letters*, 8:67–71, 1989.

T. A. Feo et M. G. C. Resende. « Greedy randomized adaptive search procedure ». *Journal of Global Optimization*, 6:109–133, 1995.

C. E. Ferreira, A. Martin, C. C. de Souza, R. Weismantel et L. A. Wolsey. « Formulation and valid inequalities for the node capacitated graph partitioning problem ». *Mathematical Programming*, 74:247–266, 1996.

C. E. Ferreira, A. Martin, C. C. de Souza, R. Weismantel et L. A. Wolsey. « The node capacitated graph paritioning problem: a computational study ». *Mathematical Programming*, 81:229–256, 1998.

A. A. Gaivoronski, A. Lisser, R. Lopez et H. Xu. « Knapsack problem with probability constraints ». *Journal of Global Optimization*, 49:397–413, 2011.

F. Galea et R. Sirdey. « Méthode de cadencement d'applications flot de données cyclostatiques ». Rapport technique DACLE/10-070, Commissariat à l'Énergie Atomique, 2010.

G. R. Gao, R. Govindarajan et P. Panangaden. « Well-behaved dataflow programs for DSP computation ». Dans *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 561–564, 1992.

M. R. Garey et D. S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness*. W. H. Freeman and Company, 1979.

A. Girault, B. Lee et E. A. Lee. « Hierarchical finite state machines with multiple concurrency models ». *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(6):742–760, 1999.

T. Goubier, F. Blanc, S. Louise, R. Sirdey et V. David. « Définition du langage de programmation ΣC ». Rapport technique DTSI/SARC/08-466, Commissariat à l'Énergie Atomique, 2008a.

T. Goubier, R. Sirdey et V. David. « Définition du langage de programmation τC ». Rapport technique DTSI/SARC/08-505, Commissariat à l'Énergie Atomique, 2008b.

T. Goubier, R. Sirdey, S. Louise et V. David. « ΣC: a programming model and langage for embedded manycores ». Dans *Lecture Notes in Computer Science*, tome 7016, pages 385–394, 2011.

J. Gupta. « Two stage, hybrid flow-shop scheduling problem ». *Journal of the Operational Research Society*, 39:359–364, 1988.

J. P. Hart et A. W. Shogan. « Semi-greedy heuristics: an empirical study ». *Operations Research Letters*, 6:107–114, 1987.

B. Hendrickson et T. G. Kolda. « Graph partitionning models for parallel computing ». *Parallel Computing*, 26:1519–1534, 2000.

J.-J. Hwang, Y.-C. Chow, F. Anger et C.-Y. Lee. « Scheduling precedence graphs in systems with interprocessor communication times ». *SIAM Journal on Computing*, 18:244–257, 1989.

T. Ibaraki. « The power of dominance relations in branch-and-bound algorithms ». *Journal of the ACM*, 24:264–279, 1977.

IEEE. « Draft standard for a precision clock synchronization protocol for networked measurement and control systems ». Rapport technique IEEE P1588/D1-I 2007-04-15, Institute of Electrical and Electronics Engineer, 2007.

ITU-T. « Timing requirements of slave clocks suitable for use as node clocks in synchronization networks ». Rapport technique ITU-T Recommendation G.812 (06/2004), International Telecommunication Union, 2004.

D. S. Johnson, C. R. Aragon, L. A. McGeoch et C. Schevon. « Optimization by simulated annealing. An experimental evaluation part I: graph partitioning ». *Operations Research*, 37:865–892, 1989.

G. Kahn. « The semantics of a simple language for parallel programming ». Dans *Proceedings of the IFIP Congress*, pages 471–475, 1974.

Kalray. « Introduction to the MPPA technology. A technical overview ». Rapport technique KETD-58, Kalray S. A., 2011.

H. Kasahara et S. Narita. « Practical multiprocessor scheduling algorithms for efficient parallel processing ». *IEEE Transactions on Computers*, 33:1023–1029, 1984.

H. Kellerer, U. Pferschy et D. Pisinger. *Knapsack problems*. Springer, 2004.

H. Kerivin et R. Sirdey. « Polyhedral combinatorics of a resource-constrained ordering problem part II: on the process move program polytope ». Rapport technique PE/BSC/INF/017913 V01/EN, service d'architecture BSC, Nortel GSM Access R&D, France, 2006.

W. Kernighan et S. Lin. « An efficient procedure for partitioning graphs ». *Bell Systems Technical Journal*, 49:291–307, 1970.

H. Khlifi et J.-C. Grégroire. « Low complexity offline and online clock skew estimation and removal ». *Computer Networks*, 50:1872–1884, 2006.

O. Klopfenstein. « Rerouting tunnels for MPLS network resource optimization ». *European Journal of Operational Research*, 188:293–312, 2008.

D. E. Knuth. *Sorting and searching*, tome 3 de *The Art of Computer Programming*. Addison-Wesley, 1998.

B. Korte et J. Vygen. *Combinatorial optimization. Theory and algorithms*. Springer, 2000.

X. Lagrange, P. Godlewski et S. Tabbane. *Réseaux GSM. Des principes à la norme*. Hermès Science Publications, 2000.

E. A. Lee. « Recurrences, iteration and conditionals in statically scheduled block diagram languages ». Dans *VLSI Signal Processing III*, pages 330–340, 1988.

E. A. Lee et D. G. Messerschmitt. « Synchronous data flow ». *Proceedings of the IEEE*, 75(9):1235–1245, 1987.

E. A. Lee et T. M. Parks. « Dataflow process networks ». *Proceedings of the IEEE*, 83(5):773–799, 1995.

Y.-T. S. Li et S. Malik. « Performance analysis of embedded software using implicit path enumeration ». Dans *ACM SIGPLAN Workshop on Languages, Compilers and Tools for Real-Time Systems*, pages 88–98, 1995.

A. Lisser et F. Rendl. « Telecommunication clustering using linear and semidefinite programming ». *Mathematical Programming*, 95:91–101, 2003.

E. M. Loiola, N. M. N. de Abreu, P. O. Boaventura-Netto, P. Hahn et T. Querido. « A survey for the quadratic assignment problem ». *European Journal of Operational Research*, 176:657–690, 2007.

F. Maurice et R. Sirdey. « Method of synchronisation within a base station system ». Brevet WO/2010/083930, 2010.

B. Maurin et R. Sirdey. « Etude de schémas de parallélisation d'algorithmes pour problèmes combinatoires ». Rapport technique DACLE/11-430, Commissariat à l'Énergie Atomique, 2011.

T. Megel, R. Sirdey et V. David. « Minimizing task preemptions and migrations in multiprocessor optimal real-time schedules ». Dans *Proceedings of the 31st IEEE Real-Time Systems Symposium*, pages 37–46, 2010.

N. Megiddo. « Linear-time algorithms for linear programs in $\mathbb{R}^3$ and related problems ». *SIAM Journal of Computing*, 12:759–776, 1983.

M. Minoux. *Programmation mathématique. Théorie et algorithmes (tome 2)*. Dunod, 1983.

M. Minoux et K. Barkaoui. « Deadlocks and traps in Petri nets as Horn-satisfiability solutions and some related polynomially solvable problems ». *Discrete Applied Mathematics*, 29:195–210, 1990.

S. B. Moon, P. Skelly et D. Towsley. « Estimation and removal of clock skew from network delay measurements ». Dans *Proceedings of IEEE INFOCOM*, pages 227–234, 1999.

M. Mouly et M.-B. Pautet. *The GSM System for Mobile Commnunications. A comprehensive overview of the European Digital Cellular Systems*. Telecom Publishing, 1992.

T. Murata. « Petri nets: properties, analysis and applications ». *Proceedings of the IEEE*, 77:541–580, 1989.

C. J. Murray. *The supermen. The story of Seymour Cray and the technical wizards behind the supercomputer*. John Wiley & Sons, 1997.

G. L. Nemhauser et L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1999.

I. Norros. « On the use of fractional brownian motion in the theory of connectionless networks ». *IEEE Journal on Selected Areas in Communications*, 13: 953–962, 1995.

K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran et C. Diot. « Measurement and analysis of single-hop delay on an IP backbone network ». *IEEE Journal on Selected Areas in Communications*, 21:908–921, 2003.

V. Paxson. « On calibrating measurements of packet transit times ». Dans *Joint International Conference on Measurement and Modeling of Computer Systems*, pages 11–21, 1998.

B. Poirier, R. Roy et M. Dagenais. « Accurate offline synchronization of distributed traces using kernel-level events ». *ACM Operating Systems Review*, 44:75–87, 2010.

M. Prais et C. C. Ribeiro. « Reactive GRASP: an application to a matrix decomposition problem in TDMA traffic assignment ». *INFORMS Journal on Computing*, 12:164–176, 2000.

M. Raynal. *La communication et le temps dans les réseaux et les systèmes répartis*. Eyrolles, 1991.

M. Raynal et M. Singhal. « Logical time: capturing causality in distributed systems ». *IEEE Computer*, 29, 1996.

I. E. Richardson. *The H.264 advanced video compression standard*. Wiley, 2010.

G. Saporta. *Probabilités, analyse des données et statistiques*. Éditions Technip, 1990.

R. Sirdey. « A new algorithm for PCM defense on the Agprs interface ». Rapport technique PE/BSC/DD/010799, Service d'architecture BSC, Nortel GSM Access R&D, 2004a.

R. Sirdey. « Specification of the BSCe3 load balancing algorithm ». Rapport technique PE/BSC/DD/009931, Service d'architecture BSC, Nortel GSM Access R&D, 2004b.

R. Sirdey. « An algorithm proposal for the BTS 18000 Intelligent Shutdown problem ». Rapport technique PE/BTS/INF/14934, Service d'architecture BSC, Nortel GSM Access R&D, 2005.

R. Sirdey. « A new algorithm for the TDMA/TRX mapping problem in SPR ». Rapport technique PE/BSC/DD/011889, Service d'architecture BSC, Nortel GSM Access R&D, 2006a.

R. Sirdey. « A polyhedral approach to reroute sequence planning in MP-LS networks ». Rapport technique PE/BSC/INF/20291 V01/EN, service d'architecture BSC, Nortel GSM Access R&D, France, 2006b.

R. Sirdey. « Sudokus et algorithmes de recuit ». *Quadrature*, 62:7–10, 2006c.

R. Sirdey. « Combinatorial optimization problems in wireless switch design ». *4OR*, 5:319–33, 2007a.

R. Sirdey. « Des solutions pour faire bonne figure — Programmation linéaire ». *La Recherche*, 407:82–83, 2007b.

R. Sirdey. *Modèles et algorithmes pour la reconfiguration de systèmes répartis utilisés en téléphonie cellulaire*. Thèse de doctorat, Université de Technologie de Compiègne, 2007c.

R. Sirdey. « Optimiser... en découpant des polyèdres ». *L'Ouvert*, 115:51–61, 2007d.

R. Sirdey. « Sudokus et programmation linéaire ». *Quadrature*, 63:9–13, 2007e.

R. Sirdey. « Vérifier des programmes en prouvant des théorèmes — Programmes ». *La Recherche*, 412:98–99, 2007f.

R. Sirdey. « Models and algorithms for the reconfiguration of distributed wireless switching systems ». *4OR*, 6:195–198, 2008a.

R. Sirdey. « Programmation linéaire et vérification de propriétés de sûreté dans le modèle des flots de données général ». Rapport technique DTSI/SARC/08-337, Commissariat à l'Énergie Atomique, 2008b.

R. Sirdey. « Se mettre d'accord entre ordinateurs — Informatique théorique ». *La Recherche*, 418:82–83, 2008c.

R. Sirdey. « Infrastructure d'authentification, de sécurisation des données et des codes exécutables pour la couche d'Infrastructure-as-a-service (IaaS) du modèle de Cloud Computing ». Rapport technique DACLE/10-625, Commissariat à l'Énergie Atomique, 2010.

R. Sirdey. « Functional specification of an authentication and secured execution infrastructure for the Infrastructure-as-a-Service (IaaS) layer of the Cloud Computing model ». Rapport technique DACLE/11-617, Commissariat à l'Énergie Atomique, 2011.

R. Sirdey et P. Aubry. « A linear programming approach to general dataflow process networks verification and dimensioning ». *Electronic Proceedings in Theoretical Computer Science*, 38:115–119, 2010.

R. Sirdey, J. Carlier, H. Kerivin et D. Nace. « On a resource-constrained scheduling problem with application to distributed systems reconfiguration ». *European Journal of Operational Research*, 183:546–563, 2007.

R. Sirdey, J. Carlier et D. Nace. « Approximate resolution of a resource-constrained scheduling problem ». *Journal of Heuristics*, 15:1–17, 2009a.

R. Sirdey, J. Carlier et D. Nace. « A GRASP for a resource-constrained scheduling problem ». *International Journal of Innovative Computing and Applications*, 2:143–149, 2010.

R. Sirdey et V. David. « Sur un modèle d'exécution haute performance cadencé par un temps logique vectoriel ». Rapport technique DTSI/SARC/08-288, Commissariat à l'Énergie Atomique, 2008.

R. Sirdey et V. David. « Approches heuristiques des problèmes de partitionnement, placement et routage de réseaux de processus sur architectures parallèles clusterisées ». Rapport technique DTSI/SARC/09-470, Commissariat à l'Énergie Atomique, 2009.

R. Sirdey et V. David. « Système d'ordonnancement de l'exécution de taches cadencé par un temps logique vectoriel (circuit d'accélération) ». Brevet, n° de dépot 1003964, 2010a.

R. Sirdey et V. David. « Système d'ordonnancement de l'exécution de taches cadencé par un temps logique vectoriel (procédé d'exécution) ». Brevet, n° de dépot 1003963, 2010b.

R. Sirdey, V. David, P. Dubrulle, T. Goubier et S. Louise. « Une procédure de construction des binaires pour un cluster MPPA sans translation d'adresse ». Rapport technique DTSI/SARC/09-205, Commissariat à l'Énergie Atomique, 2009b.

R. Sirdey, P. Dore et L. Dorys-Charnalet. « Dynamic archipelago allocation HLD ». Rapport technique PE/TCU/DD/010373, Service d'architecture BSC, Nortel GSM Access R&D, 2005.

R. Sirdey et H. Kerivin. « Polyhedral combinatorics of a resource-constrained ordering problem part I: on the partial linear ordering polytope ». Rapport technique PE/BSC/INF/017912 V01/EN, service d'architecture BSC, Nortel GSM Access R&D, France, 2006.

R. Sirdey et H. Kerivin. « A branch-and-cut algorithm for a resource-constrained scheduling problem ». *RAIRO—Operations Research*, 41:235–251, 2007.

R. Sirdey et F. Maurice. « A linear programming approach to highly precise clock synchronization over a packet network ». *4OR*, 6:393–401, 2008.

R. Sirdey, D. Plainfossé et J.-P. Gauthier. « A practical approach to combinatorial optimization problems encountered in the design of a high availability distributed system ». Dans *Proceedings of the International Network Optimization Conference*, pages 532–539, 2003.

O. Stan, R. Sirdey, J. Carlier et D. Nace. « L'apport de l'optimisation sous incertitudes pour les systèmes temps réel ». Dans *École d'Été Temps Réel*, 2011.

J. Wang, M. Zhou et H. Zhou. « Clock synchronization for internet measurements: a clustering algorithm ». *Computer Networks*, 45:731–741, 2004.

P. A. S. Ward. « An offline algorithm for dimension-bound analysis ». Dans *Proceedings of the 1999 IEEE International Conference on Parallel Processing*, pages 128–136, 1999.

M. Yannakakis. « The complexity of the partial order dimension problem ». *SIAM Journal on Algebraic and Discrete Methods*, 3(3):351–358, 1982.

G. Yu. « On the max-min 0-1 knapsack problem with robust optimization applications ». *Operations Research*, 44:407–415, 1996.

L. Zhang, Z. Liu et C. H. Xia. « Clock synchronization algorithms for network measurements ». Dans *Proceedings of IEEE INFOCOM*, pages 160–169, 2002.

Mis en page avec LaTeX, compilation du 5 décembre 2011.
Version finale.