Méthodes numériques pour la solution de systèmes Markoviens à grand espace d'états

Paulo Fernandes

THÈSE

présentée par

Paulo Henrique Lemelle FERNANDES

pour obtenir le titre de DOCTEUR

de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

(arrêté ministériel du 30 Mars 1992)

Spécialité : **Informatique**

Méthodes Numériques pour la Solution de Systèmes Markoviens à Grand Espace d'États

---

Date de soutenance : février 1998

Composition du jury

Rapporteurs :	JEAN-MICHEL FOURNEAU
	SERGE HADDAD
Examinateurs :	RENÉ DAVID
	BRIGITTE PLATEAU (directeur)
	WILLIAM J. STEWART

Thèse préparée au sein du
Laboratoire de Modélisation et Calcul
(Institut de Mathématiques Appliquées de Grenoble)

.

à Marilia Burger, mon autre moi

.

La vie est complexe,
car elle est composée d’un partie réelle
et d’une partie imaginaire.
Par exemple, je crois que j’aurais pu faire mieux,
mais j’ai fait du mieux que j’ai pu.
avez-vous remarqué?

Il y a tant des gens qui m'ont aidé ...

Brigitte Plateau a introduit les recherches dans les réseaux d'automates stochastiques. Nous tous que travaillons dans ce domaine lui devons les grands axes de recherche dans ce domaine. Moi, en particulier, je le dois en plus un encadrement exceptionnel et une motivation hors du commun.

Je remercie à Billy Stewart par sa personnalité très captivante et la très solide connaissance du domaine des chaînes de Markov qu'il n'hésite pas a partager. Son influence dans mes travaux et dans ma conception de la recherche me sont très chères.

Aux membres de mon jury, René David, Jean-Michel Fourneau et Serge Haddad, je remercie leur disponibilité et les commentaires très utiles. Pour l'aide précieuse dans la compréhension et implémentation des méthodes GMRES et d'Arnoldi, je remercie Yousef Saad.

Le travail de programmation dans cette thèse et la confection de ce manuscrit me seraient impossibles sans l'aide de mes amis et collègues de bureau. Je pense surtout à Mathias Doreille, Benhur Stein, Gregory Mounié, Roberta Jungblut-Hessel, João Paulo Kitajima, Gerson Cavalheiro, Eric Morel et Nicolas Maillard. En particulier, le tirage des versions finales de cette thèse a été fait avec l'aide de Alfredo Goldman vel-Lejbman et Pierre-Eric Bernard.

Je veux aussi profiter de cette page pour citer les amis que m'ont permis, avec leur amitié, un séjour formidable ici à Grenoble. Je tiens surtout à citer mes amis Marcelo Pasin, Marcelo Portes de Albuquerque, Lara Botelho de Albuquerque, Ana Paula Felippe, Alexandre Carissimi, Luiz Henrique Alves de Medeiros, Denise Dumke de Medeiros, Elson Manoel Pereira, Stella Maris Meira da Veiga Pereira, Fabiano Luiz dos Santos Garcia, Alessandra Telles Garcia, Marilena Bittar, Remis Balaniuk, Silvio Nabeta, Jaime Sichman, Patricia Palagi, Dominique Verchère, Martha-Rosa Castañeda-Retiz et Pascal Caillaud. Je voudrais aussi vous parler de mes grands amis José Celso Freire Junior et Cleidinéia Cristina Saquetti Seabra Freire, qui ont été les compagnons les plus proches à moi et ma femme pendant cette période.

Je ne pourrais jamais oublier de citer ma famille au Brésil qui m'a toujours réchauffé le coeur et fait le bonheur des compagnies téléphoniques. Je remercie à mon père, Valmir Dasso Fernandes, à ma mère, Nancy Maria Lemelle Fernandes et à ma soeur, Claudia Lemelle Fernandes pour leur soutien inconditionnel.

Finalement, je tiens à dire un mot très spécial à Luiz Gustavo Leão Fernandes, mon cousin et fidèle camarade qui est vraiment le frère que je n'ai jamais eu, car il est fondamental savoir qu'il existe d'autres gens avec la même vision du monde que moi. *Mon frère, voilà la victoire d'un style, le nôtre.*

J'ai sans doute oublié des gens qui ne fallait pas oublier, mais après tout il y a tant des gens que m'ont aidé ...

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Nous n'apprenons rien au lecteur en rappelant que la complexité des systèmes informatiques ne fait qu'augmenter. Les architectures parallèles, les réseaux à haut débit et une pléiade d'autres systèmes à traitement distribué demandent des outils chaque fois plus complexes pour leur planification et gestion. Afin de raccourcir les délais de prototypage et de test, des méthodes de *modélisation* et de *prédiction de performances* sont mises en oeuvre. Nous allons appeler ces deux étapes par le terme générique d'*évaluation de performances*.

Dans cette thèse nous sommes intéressés par l'évaluation de performances des systèmes informatiques parallèles et distribués. En étant modulaires, ces systèmes sont naturellement grands et très complexes à traiter par les méthodes traditionnelles d'évaluation de performances[1]. L'utilisation des réseaux de files d'attente [56] est limitée par les contraintes imposées pour garder le modèle dans le cadre de ceux qui sont *traitables*[2]. Les résultats obtenus avec les méthodes d'approximation [94] peuvent être peu précis et demandent des validations très difficiles. Les simulations [91] sont une alternative possible, quoique leur mise en oeuvre et leur exploitation pour des grands systèmes soient très coûteuses. Ceci nous laisse les méthodes basées sur les chaînes de Markov [100], mais là aussi un grand nombre de difficultés sont rencontrées. Notamment, la taille de l'espace d'états généré est si grande que non seulement le modèle du système n'est pas *traitable*, mais souvent la matrice de transition de la chaîne ne peut même pas être stockée.

C'est à cette problématique, l'évaluation de performances de systèmes complexes décrits par des modèles avec de très grands espaces d'états, que les travaux de cette thèse s'adressent.

Les systèmes parallèles et distribués sont souvent composés par des modules qui ont un comportement presque indépendant, *i.e.*, où des interactions entre les modules sont

[1] Dans la suite de cette introduction nous citons les méthodes traditionnelles d'évaluation de performances (section 1.1). Le lecteur familier de ces méthodes peut sauter de cette section qui ne fait que donner les principales références pour chaque méthode.

[2] Nous allons utiliser l'expression *traitement des modèles* pour faire référence à l'obtention numérique des indices de performances.

plutôt rares. De plus, les interactions se font par des mécanismes comme les synchronisation et échanges de messages. Pour décrire ce type de comportement, un formalisme de modélisation, les réseaux d'automates stochastiques [5, 79], a été proposé. Avec ce formalisme, nous pouvons décrire des modèles équivalents à des chaînes de Markov sans pour autant tomber dans les difficultés habituelles[3]. Notamment, les réseaux d'automates stochastiques réduisent énormément les besoins de stockage de la matrice de transition de la chaîne de Markov équivalente (un format de stockage basé sur une formule tensorielle [26] est utilisé). Malheureusement, ces gains de stockage se traduisent fréquemment par une augmentation prohibitive des coûts de *traitement* du modèle.

L'objectif de cette thèse est la réduction des coûts de traitement lors de l'utilisation des réseaux d'automates stochastiques. Cette réduction des coûts se fait à travers des méthodes numériques efficaces qui tirent parti du format tensoriel de stockage de la matrice de transition.

Dans la section suivante nous faisons une synthèse des méthodes d'évaluation de performances. Ensuite, nous précisons les objectifs de cette thèse par rapport au panorama établi.

1.1 Méthodes d'Évaluation de Performances

L'*évaluation de performances* peut être partagée en deux étapes distinctes que nous allons appeler:

- la *modélisation* - le développement d'une description formelle du système;
- la *résolution* - l'obtention des prédictions de performances du système.

Bien que dans la pratique nous pouvons clairement identifier une association des méthodes d'évaluation de performance à certains formalismes de modélisation, dans cette thèse nous ferons une distinction très nette entre ces deux concepts. Dans la prochaine section nous allons aborder les formalismes de modélisation. La section suivante présente les méthodes de résolution.

1.1.1 Formalismes de Modélisation

Historiquement, les recherches en modélisation et prédiction de performances ont toujours été très liées. Les travaux les plus anciens viennent de l'analyse directe des processus stochastiques [106] et plus généralement des chaînes de Markov [96, 100].

Vers la fin des années 50, avec les travaux de Jackson [49, 50], il est apparu une nouvelle approche basée sur les réseaux de files d'attente [56, 112]. Les recherches dans ce domaine ont connu un formidable essor jusqu'à la fin des années 70, notamment avec les travaux de Little [63], Baskett, Chandy, Muntz et Palacios [9], et Reiser et Lavenberg [88].

[3] Les réseaux d'automates stochastiques sont décrits en détail dans le chapitre 3.

La nécessité de traitement de problèmes plus complexes que ceux couverts par les réseaux de files d'attente classiques a motivé dès les années 70 d'autres approchées en évaluation de performance. Ces approches alternatives ont souvent abouti à des extensions du formalisme de réseaux de files d'attente, comme c'est le cas des travaux sur:

- les méthodes d'approximation de Chandy et Sauer [21] et de Courtois [22];
- les réseaux à capacité limitée de Dallery (*fork-join*) [24, 25];
- les réseaux avec clients négatifs (réseaux généralisés) [41];
- les réseaux avec contrôle de décision [13, 101];
- les réseaux hiérarchiques [18].

L'ensemble de ces travaux a fourni des outils puissants pour une classe relativement restreinte de systèmes d'attente.

D'autres approches ont abandonné le formalisme des réseaux de files d'attente parce qu'il manquait de précision dans la description des comportements complexes. Un grand nombre de ces approches a adopté les réseaux de Petri [15, 75, 89] comme formalisme de base en ajoutant des extensions qui vont de simples temporisations constantes [90, 104], jusqu'à des mécanismes beaucoup plus sophistiqués comme par exemple:

- les réseaux de Petri stochastiques [38];
- les réseaux de Petri stochastiques généralisés [2, 3];
- les réseaux de Petri de haut niveau (réseaux colorés) [45, 46, 52];
- les réseaux de Petri stochastiques généralisés superposés [30].

Ces travaux ont permi l'intégration d'outils d'évaluation de performance à l'ensemble des outils d'analyse structurelle développés pour les réseaux de Petri [89, 76].

Le Calcul des Systèmes de Communication (CCS - *Calculus of Communicating Systems*) [67] a inspiré d'autres approches de modélisation. Ceci est le cas de:

- des algèbres des processus stochastiques [44, 47];
- des réseaux d'automates stochastiques [5, 81].

Ces approches n'ont pas la notion d'*entités* et de *flot* (respectivement *clients* et *routage* dans les réseaux de files d'attente ou *jetons* et *arcs* dans les réseaux de Petri). En revanche, elles offrent une vision compositionnelle de sous-systèmes qui interagissent entre eux (concept de modèles modulaires).

Enfin, des outils algébriques ont été développés sur les graphes afin de décrire le comportement temporel d'événements.

- les algèbres (*max, +*) [6, 7, 40];
- les algèbres exotiques [20, 58, 69].

L'utilisation de ces méthodes offre une vision très distincte des approches précédentes. Les algèbres (*max, +*) et les algèbres exotiques en général ne sont pas basées sur les chaînes de Markov [96, 100, 106]. Les autres formalismes sont en majorité basés sur des hypothèses Markoviennes. Le niveau de détail nécessaire à la description des modèles réduit l'utilisation, au moins pour le moment, de ce formalisme à des cas réels. La figure 1.1 présente un résumé de la classification présentée.

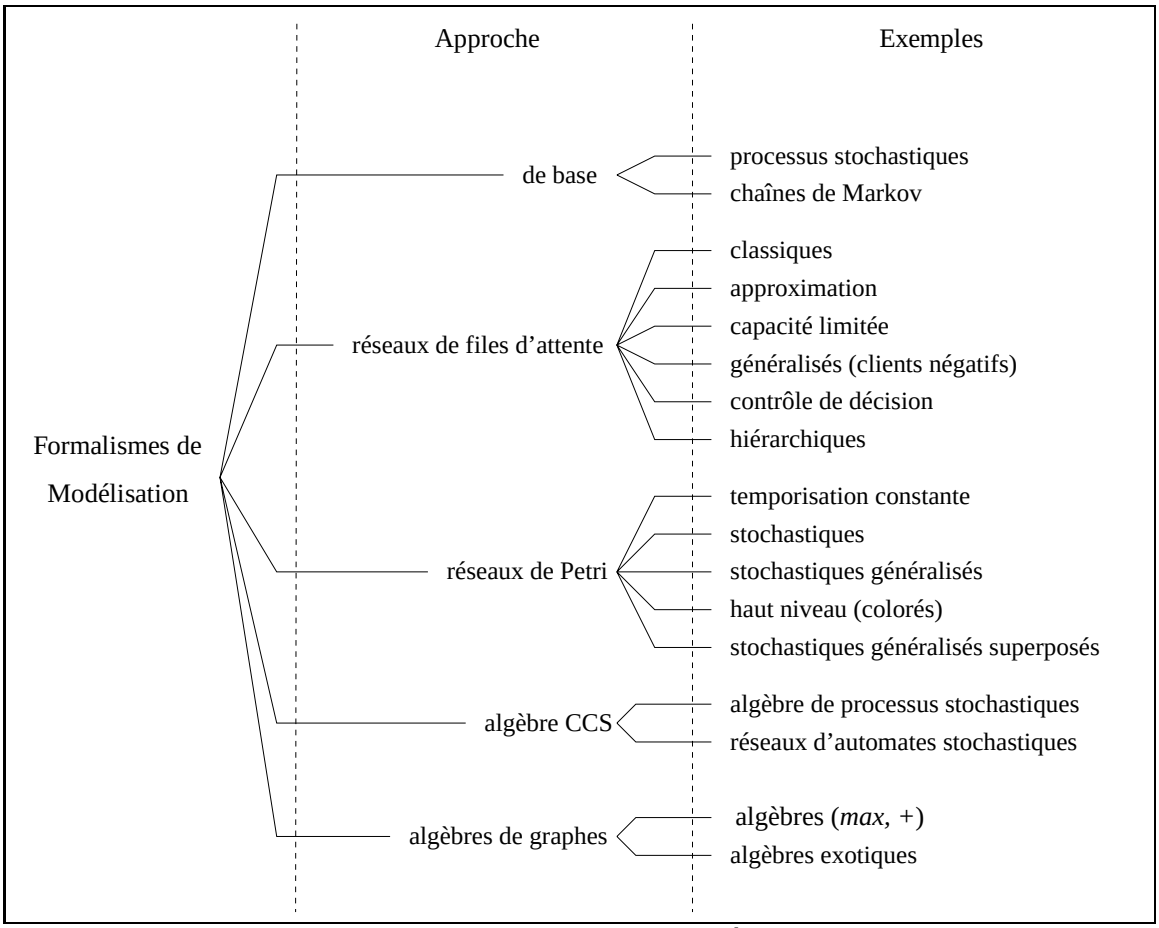


FIG. 1.1: Formalismes de Modélisation

1.1.2 Obtention des Indices de Performances

Nous sommes intéressés par calculer l'état stationnaire du modèle, *i.e.*, la proportion de temps que la chaîne de Markov[4] reste dans chacun des états sur une trajectoire de durée infinie (théorème ergodique [48]). Cette solution est exprimée par un vecteur de probabilité associant une probabilité à chaque état de la chaîne. À partir de ce vecteur nous pouvons calculer plusieurs informations sur le système modélisé, par exemple le nombre moyen de tâches traitées, les délais moyens, etc.

Cette résolution correspond à l'obtention du vecteur π solution du système:

$$\pi Q = 0 \qquad \|\pi\|_2 = 1 \tag{1.1}$$

où Q est une matrice décrivant le processus de Markov appelée matrice de transition ou générateur infinitésimal.

Parmi les méthodes de résolution nous pouvons faire la distinction entre:

– les méthodes analytiques [9, 56, 88];

[4]Nous allons restreint notre discours au cadre Markovien, que recouvre notre étude et une partie importante des travaux cités ci-dessous.

– les méthodes numériques [93, 96, 100];

– les simulations [91, 114, 84].

Nous n'allons pas considérer ici les simulations, qui est par ailleurs une voie intéressante à explorer.

Les méthodes analytiques sont les méthodes qui donnent une solution sans passer par la résolution numérique du système linéaire $\pi Q = 0$. Ces méthodes ont l'avantage d'éviter la résolution du système linéaire, système généralement très grand.

Les méthodes numériques peuvent être partagées en trois groupes:

– les méthodes numériques directes;

– les méthodes numériques itératives avec analyse structurelle;

– les méthodes numériques itératives pures.

La littérature décrivant les méthodes numériques de résolution de systèmes linéaires est très abondante [59, 93, 96, 100]. Il faut néanmoins tenir compte du fait que le générateur infinitésimal d'une chaîne de Markov est une matrice possédant des caractéristiques particulières[5], favorisant l'application de certaines méthodes mais empêchant l'utilisation d'autres. Les méthodes numériques de résolution de systèmes linéaires applicables aux chaînes de Markov sont généralement des méthodes itératives. Les méthodes directes, comme la méthode de Gauss [59, 96], ne sont pas utilisables pour des modèles de grande taille (nombre d'états) comme ceux qui nous préoccupent ici.

Les méthodes itératives peuvent aller des méthodes simples comme la méthode de la puissance, Jacobi, Gauss-Siedel et sur-relaxation successive, jusqu'à des méthodes plus complexes comme les méthodes de projection (Arnoldi, GMRES, Lanczos, etc). Le lecteur peut trouver des descriptions très précises de ces méthodes dans les ouvrages de Stewart [100] et de Saad [93]. Les méthodes de la puissance, Arnoldi et GMRES sont aussi présentées, avec moins de détails, dans le chapitre 6 de cette thèse.

Étant donné la grande taille des chaînes de Markov à résoudre nous avons intérêt à faciliter l'application des méthodes de résolution de systèmes linéaires. Un très grand nombre de solutions sont disponibles dans la littérature. Nous pouvons les diviser en deux grands groupes (qui ne sont pas forcément exclusifs):

– les méthodes qui font des analyses structurelles du générateur Q avant de résoudre le système [25, 39, 70, 71, 72];

– les méthodes qui ne font que des analyses numériques pour tirer profit du type de stockage utilisé pour le générateur [18, 30, 80, 86, 100].

Si pour des raisons de coût (généralement dû à la taille du problème) nous ne pouvons pas obtenir une solution exacte, il existe une solution alternative: la recherche de bornes. Le calcul de bornes de performance [22, 23] est basé sur des simplifications du générateur en déterminant une solution aussi proche que possible de la solution exacte pour une partie pertinente du modèle. Ce genre de résolution peut être fort intéressant lors de problèmes très grands (éventuellement même infinis [1, 64]). Cette résolution approchée

[5]Il faut tenir compte, entre autre, que le générateur est une matrice singulière et diagonale dominante [100].

est généralement employée en partant d'analyses structurelles de la chaîne de Markov [39, 68, 70, 94, 107]. Nous n'aborderons, dans cette thèse, que des méthodes donnant des solutions exactes. La figure 1.2 présente un résumé de la classification présentée.

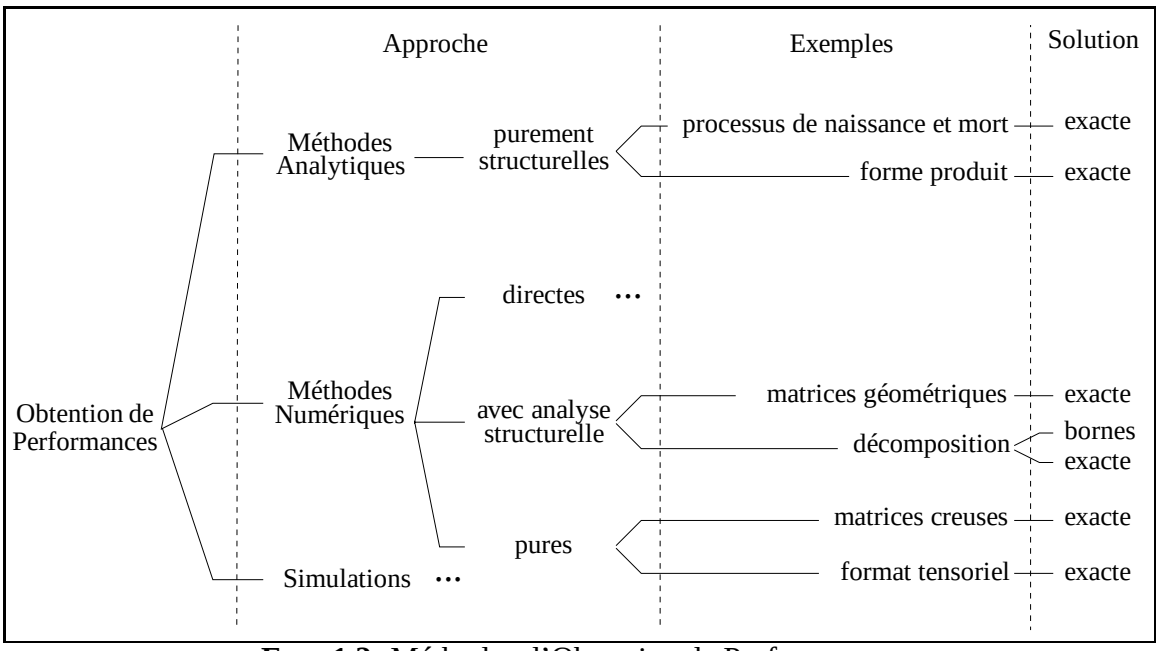


FIG. 1.2: Méthodes d'Obtention de Performances

Nous allons présenter brièvement quelques possibilités de méthodes de résolution des modèles Markoviens. Nous commençons par deux exemples de méthodes analytiques:

- la solution des processus de naissance et mort [106];
- les solutions à forme produit [2, 19, 56, 83].

Ensuite, nous citons deux méthodes numériques de résolution qui utilisent des analyses structurelles du générateur Q:

- la résolution des modèles où le générateur est une matrice géométrique [71, 72, 111];
- la résolution de modèles utilisant des techniques de décomposition [22, 39, 70, 86].

Finalement nous citons deux méthodes de résolution qui donnent une solution purement numériques de la résolution du système $\pi Q = 0$:

- la résolution des modèles où le générateur est stocké sous une forme creuse [85, 93, 96, 100];
- la résolution des modèles où le générateur est stocké par une matrice en format tensoriel [18, 30, 33, 80].

1.1.3 Méthodes Analytiques

Processus de Naissance et Mort

Un processus de naissance et mort (*birth and death process*) est un cas très particulier de chaîne de Markov. Une chaîne de Markov est un processus de naissance et mort [106]

si son générateur a la forme:

$$Q = \begin{pmatrix} -\lambda_0 & \lambda_0 & & & \\ \mu_1 & -(\lambda_1+\mu_1) & \lambda_1 & & \\ & \mu_2 & -(\lambda_2+\mu_2) & \lambda_2 & \\ & & & \ddots & \end{pmatrix} \qquad \begin{array}{rcll} q_{i,i+1} & = & \lambda_i & \text{pour } i = 0,1,\ldots \\ q_{i,i-1} & = & \mu_i & \text{pour } i = 1,2,\ldots \\ q_{i,i} & = & -(\lambda_i+\mu_i) & \text{pour } i = 0,1,\ldots \\ q_{i,j} & = & 0 & \text{pour } \|i-j\| > 1 \end{array}$$

Dans ce type de chaîne de Markov, chaque état i possède seulement deux transitions:

– une transition, dite naissance, vers son voisin $(i+1)$ avec un taux λ_i;

– une transition, dite mort, vers son voisin $(i-1)$ avec un taux μ_i.

Le premier état ne possède pas de transition de mort et le dernier état ne possède pas de transition de naissance.

Pour ce type de chaîne il est possible d'obtenir la probabilité d'un état quelconque par une formule simple. Chaque probabilité stationnaire est donnée par les formules:

$$\pi_k = \pi_0 \prod_{i=0}^{k-1} \left(\frac{\lambda_i}{\mu_{i+1}} \right) \qquad \text{pour } k \geq 1$$

et puisque le vecteur π est un vecteur de probabilités ($\|\pi\|_2 = 1$):

$$\pi_0 = \frac{1}{1 + \sum_{k \geq 1} \left(\prod_{i=0}^{k-1} \left(\frac{\lambda_i}{\mu_{i+1}} \right) \right)}$$

Ce type de chaîne de Markov, bien que très simple, couvre un grand nombre de cas, comme par exemple les files $M/M/1$ et $M/M/m$. D'autres modèles qui peuvent être décrits comme un processus de naissance et mort sont donnés dans le chapitre 8 du livre de Trivedi [106].

Forme Produit

Certaines chaînes de Markov, de par leur structure, ont des solutions avec une forme produit. Pour ces chaînes, nous ne sommes pas obligés de résoudre numériquement le système linéaire $\pi Q = 0$ pour calculer les vecteurs de probabilité décrivant l'état stationnaire de la chaîne. Les processus de naissance et mort sont un cas simple de solution à forme produit. Il est possible de trouver des solutions à forme produit pour d'autres chaînes plus complexes que les processus de naissance et mort, notamment des chaînes avec plus de transitions entre les états que seulement celles entre les états voisins.

Les cas les plus étudiés parmi les modèles Markoviens avec solution à forme produit sont probablement les réseaux de files d'attente [29, 56]. Les premiers travaux dans ce domaine sont les réseaux de Jackson [49, 50]. Ensuite plusieurs résultats [21, 41, 42, 53, 57] ont élargi les caractéristiques des réseaux de files d'attente avec solution à forme produit.

Pour un réseau de files d'attente, les solutions à forme produit passent par le calcul des taux de visite de chacune des files [9, 17, 63]. Le réseau est composé de K files et $p_{i,j}$

est la probabilité d'un client partant d'une file i aller vers la file j (probabilité de routage). Le taux de visite des files (V_k) est défini à une constante près par les K équations:

$$V_k = \sum_{i=1}^{K} V_i p_{i,k} \qquad \text{avec } k \in [1..K]$$

Les probabilités des états (globaux) de la chaîne sont calculables par produit des états individuels de chaque file.

D'autres solutions à forme-produit sont connues en partant d'autres formalismes de modélisation. Notamment quelques réseaux de Petri stochastiques [38] et réseaux de Petri stochastiques généralisés [2] ont cette propriété. Plus récemment, pour quelques modèles des réseaux d'automates stochastiques, une solution à forme produit a pu être trouvée [14, 83].

1.1.4 Méthodes Numériques avec Analyse Structurelle

Matrices Géométriques

Une solution semblable à celle des processus de naissance et mort est la technique basée sur les processus de quasi naissance et de mort (QBD - *quasi birth and death process*). Cette technique a été originellement proposée par Wallace [111] et généralisée par Neuts [71, 72] sous le nom de *solution de matrices géométriques*. L'idée de base est de considérer le générateur non pas état par état, mais bloc par bloc. Cette approche modélise une chaîne de Markov à deux composantes. En considérant comme un seul macro-état l'ensemble des états ayant la même valeur pour la première composante de la chaîne, nous avons des blocs qui représentent les transitions entre macro-états. La résolution est faite en cherchant la probabilité stationnaire pour chacun des blocs en fonction des probabilités de chacun de ses blocs adjacents.

Le cas de base, les processus de quasi naissance et de mort, sont applicables dès lors que la première composante de la chaîne est un processus de naissance et de mort, *i.e.*, dès lors que le générateur, divisé en blocs $B_{i,j}$ a la forme:

$$Q = \begin{pmatrix} B_{1,1} & B_{1,2} & & \\ B_{2,1} & B_{2,2} & B_{2,3} & \\ & B_{3,2} & B_{3,3} & B_{3,4} \\ & & & \ddots \end{pmatrix}$$

La généralisation proposée par Neuts se comprend intuitivement par le fait que si un processus à une seule composante conduit à une solution géométrique, *i.e.*, similaire à celle d'un processus de naissance et mort, un processus à deux composantes a le même type de solution si l'une des composantes a aussi une solution géométrique. Ceci est le cas des files du type G/M/1 et M/G/1, et d'autres cas [86].

Décomposition

À l'instar de la méthode des matrices géométriques, l'idée de base de ces méthodes est de décomposer le générateur en blocs $B_{i,j}$:

$$Q = \begin{pmatrix} B_{1,1} & B_{1,2} & B_{1,3} & \cdots & B_{1,N} \\ B_{2,1} & B_{2,2} & B_{2,3} & \cdots & \\ B_{3,1} & B_{3,2} & B_{3,3} & \cdots & \\ \vdots & \vdots & \vdots & \ddots & \\ B_{N,1} & & & & B_{N,N} \end{pmatrix}$$

Cette décomposition en blocs peut aussi être vue comme une agrégation [95] de certains états de la chaîne de Markov en N états agrégés, appelés macro-états. Nous pouvons donc construire une matrice Q' de dimension N, représentant le générateur de la chaîne agrégée, *i.e.*, le générateur de la chaîne composée des macro-états.

Pour chaque bloc $B_{i,j}$ nous calculons une matrice $C_{i,j}$ appelée *complément stochastique* [66] définie par:

$$C_{i,j} = B_{i,j} + B_{i\star} \left(I - B_i\right)^{-1} B_{\star i}$$

✍ **Où**

$B_{i\star}$	est la matrice bloc ligne $(B_{i,1}\ B_{i,2}\ B_{i,3}\ \ldots)$;
$B_{\star i}$	est la matrice bloc colonne $(B_{1,i}\ B_{2,i}\ B_{3,i}\ \ldots)$;
B_i	est la matrice Q sans la ligne $(B_{i,1}\ B_{i,2}\ B_{i,3}\ \ldots)$ ni la colonne $(B_{1,i}\ B_{2,i}\ B_{3,i}\ \ldots)$.

Le principe de résolution en un seul pas [22] commence par la résolution de la matrice Q'. Ensuite le complément stochastique de chaque bloc est résolu. Une fois connues les probabilités des macro-états et la solution des compléments stochastiques de chaque bloc, il est possible d'obtenir les probabilités des états originaux de la chaîne. Ceci est fait en considérant que les probabilités de l'état stationnaire du complément stochastique sont les probabilités conditionnelles des états qui composent chacun des macro-états.

Les points sensibles de cette méthode sont le calcul des éléments de la matrice Q' et le calcul des probabilités stationnaires du complément stochastique. Plusieurs méthodes [94] peuvent être employées pour éviter que ces calculs soient excessivement longs.

Les travaux de base en décomposition sont ceux des systèmes presque complètement décomposables (*nearly completely decomposable - NCD* [95]). Mise à part la méthode de décomposition en un seul pas [22], l'algorithme d'*agrégation et désagrégation* [103] présente une méthode itérative. Dans cette méthode, nous partons d'une estimation initiale pour les probabilités conditionnelles des états composants chacun des macro-états. Ensuite, une partie d'agrégation fait le calcul de la probabilité des macro-états à partir d'estimations des probabilités conditionnelles des états. La partie de désagrégation exécute le calcul de nouvelles estimations pour les probabilités conditionnelles. L'article de Schweitzer [94] présente un étude très complète des variations possibles dans les méthodes itératives de décomposition.

Les méthodes basées sur l'agrégation sont employées pour plusieurs formalismes de modélisation, notamment les réseaux de files d'attente [22], les réseaux de Petri stochastiques [15, 45, 46, 70] et les réseaux d'automates stochastiques [39, 86].

1.1.5 Méthodes Purement Numériques

Nous allons considérer deux types particuliers de stockage:

– le format creux, où seulement les éléments non nuls et leur position sont stockés;

– le format tensoriel, où le générateur est décrit par une formule contenant des opérateurs matriciels de l'algèbre tensorielle [26, 80].

Matrices Creuses

Les générateurs en format creux [100] sont largement employés et plusieurs méthodes de résolution sont adaptées à ce format. Il existe également un grand nombre de travaux numériques [8, 77, 93] pour accélérer les méthodes itératives, dont le préconditionnement, les méthodes de projection, etc.

Ces techniques représentent un gain significatif par rapport à l'utilisation de matrices stockées sous format plein. De cette façon, les restrictions ne sont plus établies en fonction du carré de la taille du problème (ordre du générateur), mais en fonction du nombre d'éléments non nuls du générateur.

Format Tensoriel

Le stockage sous format tensoriel est de plus en plus utilisé [18, 30, 70] à cause de sa faible exigence au niveau de la place mémoire requise (des matrices bien plus petites que le générateur sont stockées). Si les opérations de base, notamment la multiplication d'un vecteur par le générateur, sont déjà très efficaces pour les générateurs sous format creux, le plus grand obstacle à l'application des méthodes de résolution pour les générateurs sous format tensoriel reste la faible efficacité de ces mêmes opérations de base. Lorsque l'opération de multiplication vecteur-matrice est donnée, l'utilisation des méthodes itératives (simples ou sophistiquées) est immédiate.

Une fois résolue cette contrainte, la résolution des chaînes avec générateur sous format tensoriel reste inchangée du point de vue de la méthode itérative.

Le stockage en format tensoriel a toujours été employé pour le formalisme des réseaux d'automates stochastiques [5, 79]. Plus récemment des travaux utilisant le même type de stockage pour d'autres formalismes ont été publiés, notamment ceux utilisant les réseaux de files d'attente [18] et ceux utilisant les réseaux de Petri stochastiques [30].

1.2 Objectifs de cette Thèse

Les travaux présentés dans cette thèse se situent dans le contexte de la recherche de méthodes numériques capables d'accélérer les méthodes itératives de résolution de

modèles Markoviens. Plus précisément, nous abordons la résolution de modèles stockés dans un format tensoriel. Pour ces modèles, nous proposons d'abord des algorithmes pour rendre plus efficace la multiplication d'un vecteur de probabilités par une matrice en format tensoriel. Ensuite, nous implémentons des méthodes traditionnelles de résolution de systèmes linéaires (pour l'obtention de la solution stationnaire de la chaîne de Markov) en utilisant les algorithmes proposés.

Les résultats principaux de cette thèse se situent dans trois domaines:

– la preuve de propriétés de l'Algèbre Tensorielle Généralisée (une extension de l'algèbre tensorielle classique [26]);

– la définition d'un ensemble d'algorithmes et techniques d'optimisation pour rendre efficace la multiplication d'un vecteur par une matrice décrite sous un format tensoriel généralisé (appelée produit *vecteur-descripteur*);

– l'implémentation de méthodes itératives (avec et sans pré-conditionnement) [93, 100] pour résoudre des modèles sous format tensoriel généralisé.

L'Algèbre Tensorielle Généralisée

L'utilisation de l'algèbre tensorielle généralisée, par rapport à l'utilisation de l'algèbre tensorielle classique, permet une description plus compacte de certains générateurs de chaînes de Markov. Comme on le verra dans la suite (section 2.2.8), un produit tensoriel généralisé peut toujours être exprimé par une somme de produits tensoriels classiques. Notre but est de montrer que pour certains cas de produits tensoriels généralisés, nous pouvons garder des propriétés identiques à celles des produits tensoriels classiques.

Nous proposons des preuves pour une série de propriétés des opérateurs de l'algèbre tensorielle généralisée. Ces propriétés sont la base théorique nécessaire aux algorithmes qui rendent efficace le produit *vecteur-descripteur*.

Efficacité du Produit *Vecteur-Descripteur*

En partant des nouvelles propriétés de l'algèbre tensorielle généralisée, nous proposons plusieurs algorithmes pour effectuer le produit *vecteur-descripteur*. Ces algorithmes sont analysés de façon à montrer leur apport en terme d'efficacité à cette opération, ainsi que leur meilleure adéquation à certains types de *descripteurs*. De plus, un ensemble d'optimisations purement numériques est présenté de façon à accélérer le produit *vecteur-descripteur*.

Les algorithmes présentés permettent de garder, pour la plupart des descripteurs, la même complexité, avec l'utilisation de fonctions, que dans des cas où seule l'algèbre tensorielle classique est employée. Même lorsque nous sommes obligés d'augmenter la complexité du produit *vecteur-descripteur*, nous restons, au pire, dans le même ordre de complexité que les techniques précédemment employées [5].

Méthodes Itératives pour la Résolution des Modèles

Des algorithmes efficaces pour le produit *vecteur-descripteur* étant disponibles, nous montrons l'adéquation de ces algorithmes à trois méthodes itératives, notamment:

- la méthode de la puissance [100];
- la méthode d'Arnoldi (basée sur la projection sur un espace de Krylov [4, 93, 100]);
- la méthode GMRES (*Generalized Minimum Residual Method* - utilisant également la projection sur un espace de Krylov [92, 93, 100]).

Ensuite, nous proposons un ensemble de techniques de pré-conditionnement qui peuvent être appliquées à notre cas où les matrices ont un format tensoriel généralisé. Nous discutons les gains obtenus avec les méthodes itératives et leurs versions pré-conditionnées.

Autres Apports

Nous utilisons un formalisme de description de modèles Markoviens (les *Réseaux d'Automates Stochastiques* - RAS) qui favorise la représentation de la matrice de transition de la chaîne sous un format tensoriel. Les RAS ont été proposés dans [79] et cette approche de modélisation préconise la description d'un système par le biais de sous-systèmes qui peuvent interagir. Ces sous-systèmes sont des automates traditionnels auxquels on ajoute des comportements stochastiques et des mécanismes de synchronisation et d'échange d'informations.

Nous ne rentrons pas dans une discussion sur le mérite comparé du formalisme RAS par rapport à d'autres formalismes de modélisation (réseaux de files d'attente, réseaux de Petri, algèbre de processus, graphes de tâches, etc). Cependant le choix de ce formalisme est dû à la facilité d'obtenir une représentation tensorielle du générateur de la chaîne de Markov associée. Il est intéressant de remarquer que l'obtention du générateur en format tensoriel a fait aussi l'objet de travaux en réseaux de files d'attente [18] et en réseaux de Petri stochastiques [30, 70].

De plus, nous présentons une version académique d'un logiciel (PEPS 2.0 - *Performance Evaluation of Parallel Systems* [36]) qui intègre la définition de modèles dans le formalisme de modélisation RAS et toutes les méthodes de résolution et les optimisations numériques présentées dans cette thèse.

1.2.1 Plan de l'Ouvrage

Cette thèse est composée de deux parties. Dans la première partie nous explicitons les bases nécessaires à la compréhension des résultats numériques présentés dans la seconde. Les chapitres de la première partie de cette thèse sont:

- Algèbre Tensorielle;
- Réseaux d'Automates Stochastiques - RAS;
- Exemples de Modélisation avec les RAS.

Les chapitres de la deuxième partie sont:

– Multiplication Vecteur-Descripteur;

– Méthodes Itératives de Résolution des RAS;

– PEPS 2.0.

Partie Présentation

Le chapitre 2 donne les bases de l'algèbre tensorielle classique avec les opérateurs somme tensorielle et produit tensoriel et leurs propriétés. Des corollaires aux propriétés de base particulièrement utiles aux RAS sont montrés et une nouvelle propriété est énoncée et prouvée. L'algèbre tensorielle généralisée est introduite et toutes ses propriétés (à notre connaissance) sont énoncées et prouvées. Dans ce chapitre on montre les résultats théoriques qui sont à la base de tous les travaux développés dans cette thèse.

Le chapitre 3 présente le formalisme des RAS de façon informelle pour permettre la compréhension des modèles développés. Ensuite, une définition formelle est présentée pour éviter toute ambiguïté dans les modèles. Dans cette thèse, nous allons nous restreindre aux modèles RAS à échelle de temps continu. Le lecteur peut trouver une définition du formalisme RAS à échelle de temps discret dans [5, 80].

Le chapitre 4 donne des exemples de modélisation utilisant les RAS. Les exemples présentés ont pour but fournir des cas pratiques utilisés dans les mesures numériques des chapitres suivants. Comme cela a déjà été dit, la défense du choix des RAS en tant que formalisme de modélisation n'est pas du domaine de cette thèse. Les exemples présentés n'ont pas l'intention d'explorer ni de défendre les possibilités du formalisme, mais seulement de montrer un ensemble d'exemples qui peut fournir un jeu de générateurs en format tensoriel généralisé suffisamment représentatif.

Partie Méthodes Numériques

Le chapitre 5 présente des algorithmes et des techniques pour rendre aussi efficace que possible le produit d'un vecteur par une matrice en format tensoriel généralisé. Ce chapitre fait usage des résultats théoriques présentés au chapitre 2.

Les plus importants résultats théoriques de cette thèse ont été présentés dans le chapitre 2. De façon analogue, les résultats présentés dans le chapitre 5 sont les plus importants d'un point de vue pratique.

Le chapitre 6 montre qu'une fois établis des algorithmes efficaces pour le produit vecteur-descripteur, les méthodes itératives traditionnelles peuvent être employées. Nous présentons l'implémentation et discutons les résultats obtenus par trois méthodes de résolution: méthode de la puissance, méthode d'Arnoldi et méthode GMRES. Ensuite, nous proposons quelques possibilités de pré-conditionnement qui conviennent aux matrices décrites sous un format tensoriel généralisé. Bien que l'étude des pré-conditionnements ne soit pas tout à fait concluante en faveur d'une technique en particulier, nous montrons les directions vers lesquelles nous pouvons trouver des meilleurs résultats, *i.e.*, quelle paire de méthode itérative et technique de pré-conditionnement semble permettre la résolution la plus rapide.

Le chapitre 7 est un manuel d'utilisation du logiciel PEPS (*Performance Evaluation of Parallel Systems*) dans sa version 2.0 [36]. Le logiciel PEPS est l'outil informatique qui implémente *tous* les concepts avancés dans cette thèse. Le chapitre 7 donne un aperçu du mécanisme de fonctionnement de PEPS et montre le codage de certains exemples développés au chapitre 4.

Conclusion

Dans la conclusion nous faisons une rapide comparaison des résultats obtenus dans cette thèse par rapport à des résultats similaires de la littérature. Cette comparaison montre l'effet des résultats théoriques de l'algèbre tensorielle, et des résultats pratiques d'accélération des algorithmes et techniques proposés. Enfin, sont présentés des considérations sur les travaux futurs envisagés pour donner suite à cette thèse.

Annexes

Deux annexes font partie de ce document. La première est un résumé de toutes les notations utilisées dans cette thèse de façon à ce que le lecteur puisse avoir un accès rapide à ces informations. La seconde est composée des tableaux de résultats numériques (nombre d'itérations et temps de convergence) de plusieurs exemples utilisant plusieurs méthodes de résolution et techniques de pré-conditionnement. Bien que les principaux résultats de ces tableaux soient extraits dans le chapitre 6, nous mettons en annexe ces chiffres pour permettre au lecteur conforter ses convictions et vérifier pour chacun des cas les conclusions avancées.

Première partie

Présentation

Chapitre 2

Algèbre Tensorielle

Dans ce chapitre on présente l'algèbre tensorielle classique [26, 27, 10] ainsi qu'une extension appelée algèbre tensorielle généralisée [79, 5]. On utilise les abréviation ATC et ATG pour l'algèbre tensorielle classique et l'algèbre tensorielle généralisée respectivement. La première section présente l'ATC et ses propriétés. Dans cette même section une nouvelle propriété, particulièrement intéressante pour les RAS, est proposée et démontrée (section 2.1.3). Dans la deuxième section, l'ATG est introduite et les propriétés nécessaires pour son utilisation dans le cadre des RAS sont démontrées.

2.1 ATC - Algèbre Tensorielle Classique

L'algèbre tensorielle classique est définie par deux opérateurs matriciels:

– les produits tensoriels (aussi appelés produits de Kronecker) et

– les sommes tensorielles.

La définition des opérateurs de l'ATC et ses propriétés utilisent une notation introduite dans cette section et précédée du symbole ✍. L'annexe A contient l'ensemble des notations employées dans cette thèse pour une consultation rapide.

2.1.1 Opérateurs

Les définitions traditionnelles des ensembles de nombres naturels et réels, ainsi que des intervalles contenus dans ces ensembles, sont adoptées. L'annexe A présente la totalité de ces définitions. Cependant il est utile de rappeler la définition pour trois cas précis:

✍ **Soit**

$[a..b]$ le sous-ensemble de $\mathbb{N}$ contenant tout les valeurs de a jusqu'à b (ces valeurs incluses);

$[a,b]$ le sous-ensemble de $\mathbb{R}$ contenant toutes les valeurs de a jusqu'à b (ces valeurs incluses);

$]a,b]$ le sous-ensemble de $\mathbb{R}$ contenant toutes les valeurs de a jusqu'à b (a exclue, b incluse);

Produit Tensoriel

Le produit tensoriel de deux matrices A et B, de dimensions $(\alpha_1 \times \alpha_2)$ et $(\beta_1 \times \beta_2)$ respectivement, est une matrice de dimensions $(\alpha_1\beta_1 \times \alpha_2\beta_2)$. Cette matrice peut être vue comme une matrice avec $\alpha_1 \times \alpha_2$ blocs, chacun avec dimension $\beta_1 \times \beta_2$. La définition de chacun des éléments de la matrice résultante est faite en déterminant à quel bloc l'élément appartient et sa position interne dans le bloc.

✍ **Soit**

A	la matrice nommée A;
$A \otimes B$	le produit tensoriel des matrices A et B;
$A \times B$	le produit (traditionnel) des matrices A et B;
$a_{i,j}$	l'élément dans la ligne i et la colonne j de la matrice A;
$a_{[i,k],[j,l]}$	l'élément dans la ligne k du i-ème bloc horizontal et la colonne l du j-ème bloc vertical de la matrice A;

Prenons par exemple deux matrices A et B:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \qquad B = \begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4} \\ b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} \\ b_{3,1} & b_{3,2} & b_{3,3} & b_{3,4} \end{pmatrix}$$

Le produit tensoriel défini par $C = A \otimes B$ est égal à

$$C = \begin{pmatrix} a_{1,1}B & a_{1,2}B \\ a_{2,1}B & a_{2,2}B \end{pmatrix}$$

$$C = \left(\begin{array}{cccc|cccc} a_{1,1}b_{1,1} & a_{1,1}b_{1,2} & a_{1,1}b_{1,3} & a_{1,1}b_{1,4} & a_{1,2}b_{1,1} & a_{1,2}b_{1,2} & a_{1,2}b_{1,3} & a_{1,2}b_{1,4} \\ a_{1,1}b_{2,1} & a_{1,1}b_{2,2} & a_{1,1}b_{2,3} & a_{1,1}b_{2,4} & a_{1,2}b_{2,1} & a_{1,2}b_{2,2} & a_{1,2}b_{2,3} & a_{1,2}b_{2,4} \\ a_{1,1}b_{3,1} & a_{1,1}b_{3,2} & a_{1,1}b_{3,3} & a_{1,1}b_{3,4} & a_{1,2}b_{3,1} & a_{1,2}b_{3,2} & a_{1,2}b_{3,3} & a_{1,2}b_{3,4} \\ \hline a_{2,1}b_{1,1} & a_{2,1}b_{1,2} & a_{2,1}b_{1,3} & a_{2,1}b_{1,4} & a_{2,2}b_{1,1} & a_{2,2}b_{1,2} & a_{2,2}b_{1,3} & a_{2,2}b_{1,4} \\ a_{2,1}b_{2,1} & a_{2,1}b_{2,2} & a_{2,1}b_{2,3} & a_{2,1}b_{2,4} & a_{2,2}b_{2,1} & a_{2,2}b_{2,2} & a_{2,2}b_{2,3} & a_{2,2}b_{2,4} \\ a_{2,1}b_{3,1} & a_{2,1}b_{3,2} & a_{2,1}b_{3,3} & a_{2,1}b_{3,4} & a_{2,2}b_{3,1} & a_{2,2}b_{3,2} & a_{2,2}b_{3,3} & a_{2,2}b_{3,4} \end{array}\right)$$

Dans cet exemple l'élément $c_{4,7}(= a_{2,2}b_{1,3})$ est dans le bloc $(2,2)$ et sa position interne dans ce bloc est $(1,3)$. Le produit tensoriel $C = A \otimes B$ est défini algébriquement par l'affectation de la valeur $a_{i,j}b_{k,l}$ à l'élément dans la position (k,l) du bloc (i,j), *i.e.*:

$$c_{[i,k],[j,l]} = a_{i,j}b_{k,l} \quad \text{avec } i \in [1..\alpha_1],\ j \in [1..\alpha_2],\ k \in [1..\beta_1] \text{ et } l \in [1..\beta_2] \tag{2.1}$$

Cette représentation des éléments d'une matrice correspondant à un produit tensoriel induit un ordre sur les éléments $c_{[i,k],[j,l]}$ qui est l'ordre lexicographique sur les doublets d'indice.

Facteurs Normaux

Un cas spécifique de produit tensoriel est le produit tensoriel d'une matrice carrée par une matrice identité. Appelons ces produits des *facteurs normaux*. Avec une matrice carrée A et une matrice identité I_n deux facteurs normaux sont possibles: $(A \otimes I_n)$ et $(I_n \otimes A)$.

Reprenons la matrice A de l'exemple précédent et une matrice identité de taille 3:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \qquad I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Le facteur normal $A \otimes I_3$ est égal à

$$\left(\begin{array}{ccc|ccc} a_{1,1} & 0 & 0 & a_{1,2} & 0 & 0 \\ 0 & a_{1,1} & 0 & 0 & a_{1,2} & 0 \\ 0 & 0 & a_{1,1} & 0 & 0 & a_{1,2} \\ \hline a_{2,1} & 0 & 0 & a_{2,2} & 0 & 0 \\ 0 & a_{2,1} & 0 & 0 & a_{2,2} & 0 \\ 0 & 0 & a_{2,1} & 0 & 0 & a_{2,2} \end{array}\right)$$

Le facteur normal $I_3 \otimes A$ est égal à

$$\left(\begin{array}{cc|cc|cc} a_{1,1} & a_{1,2} & 0 & 0 & 0 & 0 \\ a_{2,1} & a_{2,2} & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & a_{1,1} & a_{1,2} & 0 & 0 \\ 0 & 0 & a_{2,1} & a_{2,2} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & a_{1,1} & a_{1,2} \\ 0 & 0 & 0 & 0 & a_{2,1} & a_{2,2} \end{array}\right)$$

✍ **Avec**

I_n la matrice identité de dimension n;

$\delta_{i,j}$ l'élément dans la ligne i et la colonne j d'une matrice identité ($\delta_{ij} = 1$, si $i = j$, sinon $\delta_{ij} = 0$);

Un cas encore plus particulier de produit tensoriel est le produit de deux matrices identités. Ce produit est une matrice identité dont la dimension est égale au produit des dimensions de chacune des matrices, *i.e.*:

$$I_n \otimes I_m = I_{nm}$$

Somme Tensorielle

La somme tensorielle de deux matrices carrées[1] A et B est définie comme la somme de facteurs normaux de chacune des matrices selon la formule:

$$A \oplus B = (A \otimes I_{n_B}) + (I_{n_A} \otimes B) \tag{2.2}$$

Prenons par exemple les matrices A et B définies par:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \qquad B = \begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix}$$

La somme tensorielle définie par $C = A \oplus B$ est égal à

$$C = \left(\begin{array}{ccc|ccc} a_{1,1} & 0 & 0 & a_{1,2} & 0 & 0 \\ 0 & a_{1,1} & 0 & 0 & a_{1,2} & 0 \\ 0 & 0 & a_{1,1} & 0 & 0 & a_{1,2} \\ \hline a_{2,1} & 0 & 0 & a_{2,2} & 0 & 0 \\ 0 & a_{2,1} & 0 & 0 & a_{2,2} & 0 \\ 0 & 0 & a_{2,1} & 0 & 0 & a_{2,2} \end{array}\right) + \left(\begin{array}{ccc|ccc} b_{1,1} & b_{1,2} & b_{1,3} & 0 & 0 & 0 \\ b_{2,1} & b_{2,2} & b_{2,3} & 0 & 0 & 0 \\ b_{3,1} & b_{3,2} & b_{3,3} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & b_{1,1} & b_{1,2} & b_{1,3} \\ 0 & 0 & 0 & b_{2,1} & b_{2,2} & b_{2,3} \\ 0 & 0 & 0 & b_{3,1} & b_{3,2} & b_{3,3} \end{array}\right)$$

$$C = \left(\begin{array}{ccc|ccc} a_{1,1}+b_{1,1} & b_{1,2} & b_{1,3} & a_{1,2} & 0 & 0 \\ b_{2,1} & a_{1,1}+b_{2,2} & b_{2,3} & 0 & a_{1,2} & 0 \\ b_{3,1} & b_{3,2} & a_{1,1}+b_{3,3} & 0 & 0 & a_{1,2} \\ \hline a_{2,1} & 0 & 0 & a_{2,2}+b_{1,1} & b_{1,2} & b_{1,3} \\ 0 & a_{2,1} & 0 & b_{2,1} & a_{2,2}+b_{2,2} & b_{2,3} \\ 0 & 0 & a_{2,1} & b_{3,1} & b_{3,2} & a_{2,2}+b_{3,3} \end{array}\right)$$

✍ **Soit**

$A \oplus B$	la somme tensorielle des matrices carrées A et B;
$A + B$	la somme (traditionnelle) des matrices A et B;
n_A	la dimension (nombre de lignes et de colonnes) de la matrice carrée A;

La somme tensorielle $C = A \oplus B$ est définie algébriquement par l'affectation de la valeur $a_{i,j}\delta_{k,l} + \delta_{i,j}b_{k,l}$ à l'élément dans la position (k, l) du bloc (i, j), *i.e.*:

$$c_{[i,k],[j,l]} = a_{i,j}\delta_{k,l} + \delta_{i,j}b_{k,l} \qquad \text{avec } i, j \in [1..n_A] \text{ et } k, l \in [1..n_B] \tag{2.3}$$

L'opérateur produit tensoriel ($\otimes$) est prioritaire sur l'opérateur somme tensorielle ($\oplus$) et les deux opérateurs tensoriels sont prioritaires sur les opérateurs traditionnels de multiplication et addition ($\times$ et $+$).

[1] Bien que le produit tensoriel soit défini pour des matrices non carrées, la somme tensorielle est définie exclusivement pour les matrices carrées.

2.1.2 Propriétés de Base

Propriétés Classiques

Ces propriétés ont été déjà prouvés dans le cadre de la définition des produits de Kronecker [10]. Pour ces propriétés aucune démonstration n'est présentée dans cette thèse. Le lecteur peut trouver dans les travaux de Davio, Deschamps et Thayse [27] et dans le livre de Bellman [10] la preuve des propriétés suivantes:

– Associativité

$$A \otimes (B \otimes C) = (A \otimes B) \otimes C \tag{2.4}$$

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C \tag{2.5}$$

– Distributivité sur la somme

$$(A + B) \otimes (C + D) = (A \otimes C) + (B \otimes C) + (A \otimes D) + (B \otimes D) \tag{2.6}$$

– Compatibilité avec la multiplication

$$(A \times B) \otimes (C \times D) = (A \otimes C) \times (B \otimes D) \tag{2.7}$$

– Compatibilité avec la transposition des matrices

$$(A \otimes B)^T = A^T \otimes B^T \tag{2.8}$$

– Compatibilité avec l'inversion des matrices (si A et B sont des matrices carrées et inversibles)

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1} \tag{2.9}$$

La propriété d'associativité pour les produits et sommes tensoriels nous permet d'affirmer que les opérateurs $\otimes_{k=1}^{N} A^{(k)}$ et $\oplus_{k=1}^{N} A^{(k)}$ sont bien définis.

✍ **Soit**

N	le nombre de matrices d'une suite finie[2];
$A^{(i)}$	la i-ème matrice d'une suite de matrices $A^{(1)}, A^{(2)}, \ldots, A^{(N-1)}, A^{(N)}$;
$a_{i,j}^{(k)}$	l'élément dans la ligne i et la colonne j de la matrice $A^{(k)}$;
n_i	la dimension de la matrice $A^{(i)}$;
$nleft_i$	le produit des dimensions de toutes les matrices avant $A^{(i)}$, *i.e.*, $\prod_{k=1}^{i-1} n_k$ (cas particulier: $nleft_1 = 1$);

[2]Dans le reste de ce document les *suites finies de matrices* seront appelées seulement *suites de matrices*. Or, seulement les suites finies seront abordées.

$nright_i$ le produit des dimensions de toutes les matrices après $A^{(i)}$, *i.e.*, $\prod_{k=i+1}^{N} n_k$ (cas particulier: $nright_N = 1$);

$\bar{n}_i$ le produit des dimensions de toutes les matrices sauf $A^{(i)}$, *i.e.*, $\prod_{k=1,k\neq i}^{N} n_k$ ($\bar{n}_i = nleft_i\ nright_i$);

$a_{[i_1,\ldots,i_m,],[j_1,\ldots,j_m]}$ l'élément de la matrice $A = \otimes_{l=1}^{N}$ de taille $\prod_{l=1}^{m} n_l$ localisé dans le bloc de taille $\prod_{l=2}^{m} n_l$ de coordonnées i_1, j_1, puis à l'intérieur de ce bloc, dans le bloc de taille $\prod_{l=3}^{m} n_l$ de coordonnées i_2, j_2 et ainsi de suite jusqu'à la position i_m, j_m du bloc (plus interne) de taille n_m;

Il est possible de généraliser la définition algébrique des produits tensoriels (équation 2.1) pour $A = \otimes_{k=1}^{N} A^{(k)}$:

$$a_{[i_1,\ldots,i_N][j_1,\ldots,j_N]} = \prod_{k=1}^{N} a_{i_k,j_k}^{(k)} \tag{2.10}$$

Pour définir un facteur normal avec une suite de matrices il faut que seulement l'une des matrices ne soit pas l'identité. Par exemple, les facteurs normaux pour la suite de deux matrices identités (de tailles n et m) et la matrice A sont:

$$A \otimes I_n \otimes I_m$$
$$I_n \otimes A \otimes I_m$$
$$I_m \otimes A \otimes I_n$$
$$I_n \otimes I_m \otimes A$$

La définition de la somme tensorielle de N matrices peut toujours être faite par la somme ordinaire de N facteurs normaux. Ceci permet la redéfinition de l'équation 2.2 de la façon suivante:

$$\bigoplus_{i=1}^{N} A^{(i)} = \sum_{i=1}^{N} \left[I_{nleft_i} \otimes A^{(i)} \otimes I_{nright_i} \right] \tag{2.11}$$

De la même façon, la définition algébrique de la somme tensorielle (équation 2.3) pour $\oplus_{k=1}^{N} A^{(k)}$ est:

$$a_{[i_1,\ldots,i_N],[j_1,\ldots,j_N]} = \sum_{k=1}^{N} \left[a_{i_k,j_k}^{(k)} \prod_{l=1,l\neq k}^{N} (\delta_{i_l,j_l}) \right] \tag{2.12}$$

Corollaires utiles pour les RAS

Comme Davio [26] l'a déjà remarqué pour les premières applications des produits de Kronecker [61], la propriété de la compatibilité avec la multiplication (équation 2.7) est fondamentale, car elle établit la relation entre les produits traditionnels de matrices et les produits tensoriels. Cette propriété est à la base des démonstrations des deux propriétés proposées dans les travaux directement liées au RAS [79, 81, 80]:

– Décomposition en Facteurs Normaux:

$$A \otimes B = (A \otimes I_{n_B}) \times (I_{n_A} \otimes B) \tag{2.13}$$

– Distributivité sur la multiplication par l'identité:

$$(A \times B) \otimes I_n = (A \otimes I_n) \times (B \otimes I_n) \tag{2.14}$$

$$I_n \otimes (A \times B) = (I_n \otimes A) \times (I_n \otimes B) \tag{2.15}$$

La décomposition en facteurs normaux est facilement vérifiée comme un cas particulier de l'équation 2.7, en sachant que

$$A \times I_{n_A} = A \quad \text{et} \quad I_{n_A} \times A = A$$

Il est possible de remplacer $A \otimes B$ par $(A \times I_{n_A}) \otimes (I_{n_B} \times B)$ dans 2.13:

$$(A \times I_{n_A}) \otimes (I_{n_B} \times B) = (A \otimes I_{n_B}) \times (I_{n_A} \otimes B)$$

Le résultat devient une application de 2.7.

Cette propriété peut être généralisée pour une suite de N matrices:

$$\bigotimes_{i=1}^{N} A^{(i)} = \prod_{i=1}^{N} I_{nleft_i} \otimes A^{(i)} \otimes I_{nright_i} \tag{2.16}$$

Les deux cas de la distributivité sur la multiplication par l'identité (2.14 et 2.15) sont prouvés de la même façon en remplaçant I_n par $I_n \times I_n$:

$$(A \times B) \otimes (I_n \times I_n) = (A \otimes I_n) \times (B \otimes I_n)$$

$$(I_n \times I_n) \otimes (A \times B) = (I_n \otimes A) \times (I_n \otimes B)$$

Pseudo-commutativité

Bien que le produit tensoriel ne soit pas commutatif il y a été prouvée [81] une propriété dite de pseudo-commutativité dans les termes suivants:

$$\bigotimes_{k=1}^{N} A^{(k)} = P_\sigma \times \left[\bigotimes_{k=1}^{N} A^{(\sigma_k)} \right] \times {P_\sigma}^T \tag{2.17}$$

où σ est une permutation sur l'intervalle $[1..N]$ et P_σ est une matrice de permutation[3] qui sera définie dans la suite.

[3]Une matrice de permutation est une matrice qui possède un seul élément non nul et égal à un par ligne et par colonne. La multiplication d'une matrice de permutation à gauche d'une matrice quelconque A représente un re-ordonancement des lignes de cette matrice. De façon analogue, la multiplication à droite d'une matrice de permutation représente un re-ordonancement des colonnes.

✍ **Soit**

σ une permutation nommée σ sur l'intervalle $[1..N]$, en pratique nous allons utiliser une permutation σ pour établir un nouvel ordre pour une suite de N matrices;

$\sigma(i)$ le rang de l'automate $\mathcal{A}^{(i)}$ dans l'ordre identifié par la permutation σ;

σ_k l'indice de l'automate placé au rang k de l'ordre identifié par la permutation σ (si $\sigma_k = i$, $\sigma(i) = k$).

Lorsque les automates sont ordonnés comme leurs indices ($\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(N)}$), les opérations tensorielles induisent un ordre sur les composants des vecteurs qui est l'ordre lexicographique sur les n-uplets d'indices $[i_1, \ldots, i_N]$ avec $i_k \in [1..n_k]$. Modifier l'ordre des automates par σ implique de modifier l'ordre des n-uplets que l'on notera $[i_1, \ldots, i_N]_\sigma$ avec $i_k \in [1..n_{\sigma_k}]$. La notation $[i_1, \ldots, i_N]_\sigma$ indique que le k-ème indice varie entre $[1..n_{\sigma_k}]$ et non plus $[1..n_k]$.

La modification de l'ordre des automates induit donc une permutation des composantes des vecteurs dont la matrice, notée P_σ, est définie par ses éléments:

$$p_{[i_1,\ldots,i_N][j_1,\ldots,j_N]_\sigma} = \begin{cases} 1 & \text{si } j_m = i_{\sigma_m} \ \ \forall m \in [1..N] \\ 0 & \text{sinon.} \end{cases}$$

Cette définition permet aussi de définir la matrice ${P_\sigma}^T$ qui est la transposé de P_σ, mais aussi son inverse.

Prenons par exemple le cas

$$A^{(1)} \otimes A^{(2)} = P_\sigma (A^{(2)} \otimes A^{(1)}) {P_\sigma}^T$$

avec $n_1 = 2$ et $n_2 = 3$. Les matrices de permutation qui vérifient cette équation sont:

$$P_\sigma = \left(\begin{array}{cc|cc|cc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right) \quad \text{et} \quad P_\sigma^T = \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

La preuve de cette propriété pour une matrice P_σ connue est présentée dans [81].

2.1.3 Nouvelle Propriété: Commutativité de Facteurs Normaux

Une nouvelle propriété pour l'ATC a été proposée dans [33]: les produits (traditionnels) de facteurs normaux sont commutatifs. Considérons deux matrices carrées (A et B), nous pouvons affirmer:

$$(A \otimes I_{n_B}) \times (I_{n_A} \otimes B) = (I_{n_A} \otimes B) \times (A \otimes I_{n_B}) \tag{2.18}$$

Démonstration Algébrique

Appelons:

– C le produit tensoriel $A \otimes I_{n_B}$

– D le produit tensoriel $I_{n_A} \otimes B$

– H le produit ordinaire $(A \otimes I_{n_B}) \times (I_{n_A} \otimes B) = C \times D$

– H' le produit ordinaire $(I_{n_A} \otimes B) \times (A \otimes I_{n_B}) = D \times C$

En calculant les éléments de la matrice $C = (A \otimes I_{n_B})$:

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$c_{[i,k],[j,l]} = a_{i,j}\delta_{k,l}$;

En calculant les éléments de la matrice $D = (I_{n_A} \otimes B)$:

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$d_{[i,k],[j,l]} = \delta_{i,j}b_{k,l}$;

En calculant les éléments de la matrice $H = C \times D$:

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$h_{[i,k],[j,l]} = \sum_{r=1}^{n_A} \left(\sum_{s=1}^{n_B} c_{[i,k],[r,s]}d_{[r,s],[j,l]}\right)$;

En remplaçant les éléments de C et D en H:

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$h_{[i,k],[j,l]} = \sum_{r=1}^{n_A} (\sum_{s=1}^{n_B} a_{i,r}\delta_{k,s}\delta_{r,j}b_{s,l})$;

En remplaçant les éléments connus (δ) en H:

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$\boxed{h_{[i,k],[j,l]} = a_{i,j}b_{k,l}}$

En calculant les éléments de la matrice $H' = D \times C$:

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$h'_{[i,k],[j,l]} = \sum_{r=1}^{n_A} \left(\sum_{s=1}^{n_B} d_{[i,k],[r,s]}c_{[r,s],[j,l]}\right)$;

En remplaçant les éléments de C et D en H':

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$h'_{[i,k],[j,l]} = \sum_{r=1}^{n_A} (\sum_{s=1}^{n_B} \delta_{i,r}b_{k,s}a_{r,j}\delta_{s,l})$;

En remplaçant les éléments connus (δ) en H':

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$\boxed{h'_{[i,k],[j,l]} = b_{k,l}a_{i,j}}$ ❑

Généralisation pour les Suites de Matrices

Cette propriété peut être généralisée à une suite de facteurs normaux, car dans une suite on peut toujours interchanger deux facteurs normaux voisins. Prenons par exemple la suite de trois facteurs normaux suivante:

$$(A \otimes I_{n_B} \otimes I_{n_C}) \times (I_{n_A} \otimes B \otimes I_{n_C}) \times (I_{n_A} \otimes I_{n_B} \otimes C)$$

Il est possible d'interchanger, par exemple, les deux premiers facteurs normaux de façon à obtenir:

$$(I_{n_A} \otimes B \otimes I_{n_C}) \times (A \otimes I_{n_B} \otimes I_{n_C}) \times (I_{n_A} \otimes I_{n_B} \otimes C)$$

Démonstration

Appelons:

– D le produit tensoriel $[A \otimes I_{n_B}]$;

– E le produit tensoriel $[I_{n_A} \otimes B]$;

L'équation précédente devient:

$$(D \otimes I_{n_C}) \times (E \otimes I_{n_C}) \times (I_{n_A} \otimes I_{n_B} \otimes C)$$

En utilisant la propriété de compatibilité avec la multiplication (2.7):

$$(D \times E) \otimes (I_{n_C} \times I_{n_C}) \times (I_{n_A} \otimes I_{n_B} \otimes C)$$

En remplaçant D et E:

$$[(A \otimes I_{n_B}) \times (I_{n_A} \otimes B)] \otimes (I_{n_C} \times I_{n_C}) \times (I_{n_A} \otimes I_{n_B} \otimes C)$$

En utilisant la propriété de commutativité des facteurs normaux (2.18) il est possible d'interchanger $[A \otimes I_{n_B}]$ avec $[I_{n_A} \otimes B]$:

$$[(I_{n_A} \otimes B) \times (A \otimes I_{n_B})] \otimes (I_{n_C} \times I_{n_C}) \times (I_{n_A} \otimes I_{n_B} \otimes C)$$

En utilisant la propriété de compatibilité avec la multiplication (2.7):

$$(I_{n_A} \otimes B \otimes I_{n_C}) \times (A \otimes I_{n_B} \otimes I_{n_C}) \times (I_{n_A} \otimes I_{n_B} \otimes C)$$ ❑

Plus généralement, étant donné σ une permutation quelconque de $[1..N]$, la formulation générale de cette propriété pour une suite de N matrices $A^{(i)}$ est:

$$\prod_{i=1}^{N} I_{nleft_i} \otimes A^{(i)} \otimes I_{nright_i} = \prod_{i=1}^{N} I_{nleft_{\sigma_i}} \otimes A^{(\sigma_i)} \otimes I_{nright_{\sigma_i}} \tag{2.19}$$

Cette propriété nous permet aussi de re-écrire l'équation 2.16 de la façon suivante pour toute permutation σ:

$$\bigotimes_{i=1}^{N} A^{(i)} = \prod_{i=1}^{N} I_{nleft_{\sigma_i}} \otimes A^{(\sigma_i)} \otimes I_{nright_{\sigma_i}}$$

2.2 ATG - Algèbre Tensorielle Généralisée

L'algèbre tensorielle généralisée est définie comme une extension de l'algèbre tensorielle classique. La différence fondamentale apportée par l'ATG est l'introduction du concept d'éléments fonctionnels. Une matrice peut désormais être composée d'éléments constants (appartenant à $\mathbb{R}$) ou d'*éléments fonctionnels*. Un *élément fonctionnel* est une fonction à valeur dans $\mathbb{R}$ selon un jeu de paramètres composé par les indices de ligne d'une ou de plusieurs matrices.

Un élément fonctionnel b qui possède l'indice de ligne de la matrice A dans son jeu de paramètres est dit *dépendant* de la matrice A. Par abus de langage, on appelle *paramètres* d'un élément fonctionnel toutes les matrices dont l'élément est dépendant. Une matrice contenant au moins un élément fonctionnel dépendant de la matrice A est dite *dépendante* de la matrice A. Les *paramètres* d'une matrice sont l'union des paramètres de tous ses éléments.

Dans le contexte des RAS, les matrices sont utilisées pour représenter des transitions entre les états d'un automate (l'élément i, j représente la transition de l'état a_i vers l'état a_j). Cette interprétation, particulière au RAS, nous permet de faire une correspondance bi-univoque entre l'état d'un automate dans un RAS et l'indice de ligne d'une matrice. Notons que ceci ne représente en aucun cas une restriction d'utilisation de l'ATG exclusivement aux RAS. Les définitions et propriétés présentées sont indépendantes de l'interprétation donnée à l'indice de ligne. Cependant, ceci est fait dans le cadre des RAS pour faciliter la compréhension sémantique des éléments fonctionnels.

À l'instar de l'ATC, l'ATG est définie par les deux opérateurs matriciels:

- les produits tensoriels généralisés;
- les sommes tensorielles généralisées.

Les notations définies dans la section précédente sont toujours valides pour les matrices n'ayant pas d'éléments fonctionnels, appelées *matrices constantes*. Les matrices avec des éléments fonctionnels, appelées *matrices fonctionnelles,* seront décrites par la notation suivante:

✍ **Soit**

a_k	l'indice de la ligne k de la matrice A, utilisé dans le contexte des RAS aussi comme l'état de l'automate $\mathcal{A}$ auquel la matrice A correspond;
$A(\mathcal{B}, \mathcal{C})$	la matrice fonctionnelle A qui possède comme paramètres les matrices B et C;
$a_{i,j}(\mathcal{B}, \mathcal{C})$	l'élément fonctionnel i, j de la matrice $A(\mathcal{B}, \mathcal{C})$;
$A(b_k, \mathcal{C})$	la matrice fonctionnelle $A(\mathcal{B}, \mathcal{C})$ où l'indice de ligne de la matrice B est déjà connu et égal à k (cette matrice peut être considéré comme dépendante de la matrice C seulement);
$a_{i,j}(b_k, \mathcal{C})$	l'élément fonctionnel i, j de la matrice $A(b_k, \mathcal{C})$;
$A(b_k, c_l)$	la matrice fonctionnelle $A(\mathcal{B}, \mathcal{C})$ avec les éléments évalués pour les états correspondants aux lignes k et l des matrices B et C respectivement

(cette matrice, ayant tous ses paramètres connus, est considérée comme constante);

$a_{i,j}(b_k, c_l)$ l'élément constant (élément fonctionnel évalué) i, j de la matrice $A(b_k, c_l)$;

$\ell_k(A)$ la matrice avec tous les éléments mis à zéro, sauf ceux appartenant à la ligne k qui est égale à la ligne k de la matrice A ($A = \sum_{k=1}^{n_A} \ell_k(A)$);

$A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A})$ le produit tensoriel généralisé entre les matrices $A(\mathcal{B})$ et $B(\mathcal{A})$;

$A(\mathcal{B}) \underset{g}{\oplus} B(\mathcal{A})$ la somme tensorielle généralisée entre les matrices $A(\mathcal{B})$ et $B(\mathcal{A})$;

2.2.1 Opérateurs

Produit Tensoriel Généralisé

Prenons par exemple deux matrices $A(\mathcal{B})$ et $B(\mathcal{A})$ définies par:

$$A(\mathcal{B}) = \begin{pmatrix} a_{1,1}[\mathcal{B}] & a_{1,2}[\mathcal{B}] \\ a_{2,1}[\mathcal{B}] & a_{2,2}[\mathcal{B}] \end{pmatrix} \qquad B(\mathcal{A}) = \begin{pmatrix} b_{1,1}[\mathcal{A}] & b_{1,2}[\mathcal{A}] & b_{1,3}[\mathcal{A}] \\ b_{2,1}[\mathcal{A}] & b_{2,2}[\mathcal{A}] & b_{2,3}[\mathcal{A}] \\ b_{3,1}[\mathcal{A}] & b_{3,2}[\mathcal{A}] & b_{3,3}[\mathcal{A}] \end{pmatrix}$$

Le produit tensoriel défini par $C = A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A})$ est égal à

$$C = \left(\begin{array}{ccc|ccc} a_{1,1}(b_1)b_{1,1}(a_1) & a_{1,1}(b_1)b_{1,2}(a_1) & a_{1,1}(b_1)b_{1,3}(a_1) & a_{1,2}(b_1)b_{1,1}(a_1) & a_{1,2}(b_1)b_{1,2}(a_1) & a_{1,2}(b_1)b_{1,3}(a_1) \\ a_{1,1}(b_2)b_{2,1}(a_1) & a_{1,1}(b_2)b_{2,2}(a_1) & a_{1,1}(b_2)b_{2,3}(a_1) & a_{1,2}(b_2)b_{2,1}(a_1) & a_{1,2}(b_2)b_{2,2}(a_1) & a_{1,2}(b_2)b_{2,3}(a_1) \\ a_{1,1}(b_3)b_{3,1}(a_1) & a_{1,1}(b_3)b_{3,2}(a_1) & a_{1,1}(b_3)b_{3,3}(a_1) & a_{1,2}(b_3)b_{3,1}(a_1) & a_{1,2}(b_3)b_{3,2}(a_1) & a_{1,2}(b_3)b_{3,3}(a_1) \\ \hline a_{2,1}(b_1)b_{1,1}(a_2) & a_{2,1}(b_1)b_{1,2}(a_2) & a_{2,1}(b_1)b_{1,3}(a_2) & a_{2,2}(b_1)b_{1,1}(a_2) & a_{2,2}(b_1)b_{1,2}(a_2) & a_{2,2}(b_1)b_{1,3}(a_2) \\ a_{2,1}(b_2)b_{2,1}(a_2) & a_{2,1}(b_2)b_{2,2}(a_2) & a_{2,1}(b_2)b_{2,3}(a_2) & a_{2,2}(b_2)b_{2,1}(a_2) & a_{2,2}(b_2)b_{2,2}(a_2) & a_{2,2}(b_2)b_{2,3}(a_2) \\ a_{2,1}(b_3)b_{3,1}(a_2) & a_{2,1}(b_3)b_{3,2}(a_2) & a_{2,1}(b_3)b_{3,3}(a_2) & a_{2,2}(b_3)b_{3,1}(a_2) & a_{2,2}(b_3)b_{3,2}(a_2) & a_{2,2}(b_3)b_{3,3}(a_2) \end{array}\right)$$

La définition algébrique du produit tensoriel généralisé $C = A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A})$ est faite par l'affectation de la valeur $a_{i,j}(b_k)b_{k,l}(a_i)$ à l'élément $c_{[i,k],[j,l]}$, *i.e.*:

$$c_{[i,k],[j,l]} = a_{i,j}(b_k)b_{k,l}(a_i) \qquad \text{avec } i, j \in [1..n_A] \text{ et } k, l \in [1..n_B] \tag{2.20}$$

Somme Tensorielle Généralisé

La définition de la somme tensorielle généralisée est faite en utilisant des produits tensoriels généralisés sur l'équation 2.2:

$$A \underset{g}{\oplus} B = (A \underset{g}{\otimes} I_{n_B}) + (I_{n_A} \underset{g}{\otimes} B) \tag{2.21}$$

Reprenons les matrices $A(\mathcal{B})$ et $B(\mathcal{A})$ utilisées pour décrire le produit tensoriel généralisé. La somme tensorielle défini par $C = A(\mathcal{B}) \underset{g}{\oplus} B(\mathcal{A})$ est égal à:

$$C = \left(\begin{array}{ccc|ccc} a_{1,1}(b_1)+b_{1,1}(a_1) & b_{1,2}(a_1) & b_{1,3}(a_1) & a_{1,2}(b_1) & 0 & 0 \\ b_{2,1}(a_1) & a_{1,1}(b_2)+b_{2,2}(a_1) & b_{2,3}(a_1) & 0 & a_{1,2}(b_2) & 0 \\ b_{3,1}(a_1) & b_{3,2}(a_1) & a_{1,1}(b_3)+b_{3,3}(a_1) & 0 & 0 & a_{1,2}(b_3) \\ \hline a_{2,1}(b_1) & 0 & 0 & a_{2,2}(b_1)+b_{1,1}(a_2) & b_{1,2}(a_2) & b_{1,3}(a_2) \\ 0 & a_{2,1}(b_2) & 0 & b_{2,1}(a_2) & a_{2,2}(b_2)+b_{2,2}(a_2) & b_{2,3}(a_2) \\ 0 & 0 & a_{2,1}(b_3) & b_{3,1}(a_2) & b_{3,2}(a_2) & a_{2,2}(b_3)+b_{3,3}(a_2) \end{array}\right)$$

La définition algébrique de la somme tensorielle généralisé $C = A(\mathcal{B}) \underset{g}{\oplus} B(\mathcal{A})$ est faite par l'affectation de la valeur $a_{i,j}(b_k)\delta_{k,l} + b_{k,l}(a_i)\delta_{i,j}$ à l'élément $c_{[i,k],[j,l]}$, *i.e.*:

$$\begin{aligned} c_{[i,k],[j,l]} = a_{i,j}(b_k)\delta_{k,l} \ + \ b_{k,l}(a_i)\delta_{i,j} \\ \text{avec } i,j \in [1..n_A] \text{ et } k,l \in [1..n_B] \end{aligned} \tag{2.22}$$

Propriétés

Dans cette section les propriétés suivantes sont prouvées:

– Distributivité du produit tensoriel généralisé sur la somme traditionnelle de matrices

$$\begin{aligned} [A(\mathcal{C},\mathcal{D}) + B(\mathcal{C},\mathcal{D})] \underset{g}{\otimes} [C(\mathcal{A},\mathcal{B}) + D(\mathcal{A},\mathcal{B})] = \\ A(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B}) \ + \ A(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} D(\mathcal{A},\mathcal{B}) \ + \\ B(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B}) \ + \ B(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} D(\mathcal{A},\mathcal{B}) \end{aligned} \tag{2.23}$$

– Associativité du produit tensoriel généralisé et de la somme tensorielle généralisée

$$[\ A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} B(\mathcal{A},\mathcal{C})\ ] \ \underset{g}{\otimes} C(\mathcal{A},\mathcal{B}) = A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} [\ B(\mathcal{A},\mathcal{C}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})\ ] \tag{2.24}$$

$$[\ A(\mathcal{B},\mathcal{C}) \underset{g}{\oplus} B(\mathcal{A},\mathcal{C})\ ] \ \underset{g}{\oplus} C(\mathcal{A},\mathcal{B}) = A(\mathcal{B},\mathcal{C}) \underset{g}{\oplus} [\ B(\mathcal{A},\mathcal{C}) \underset{g}{\oplus} C(\mathcal{A},\mathcal{B})\ ] \tag{2.25}$$

– Distributivité sur la multiplication par l'identité

$$[\ A(\mathcal{C}) \times B(\mathcal{C})\ ] \ \underset{g}{\otimes} I_{n_C} = A(\mathcal{C}) \underset{g}{\otimes} I_{n_C} \ \times \ B(\mathcal{C}) \underset{g}{\otimes} I_{n_C} \tag{2.26}$$

$$[\ I_{n_C} \underset{g}{\otimes} A(\mathcal{C})\ ] \ \times B(\mathcal{C}) = I_{n_C} \underset{g}{\otimes} A(\mathcal{C}) \ \times \ I_{n_C} \underset{g}{\otimes} B(\mathcal{C}) \tag{2.27}$$

– Décomposition en Facteurs Normaux I

$$A \underset{g}{\otimes} B(\mathcal{A}) = I_{n_A} \underset{g}{\otimes} B(\mathcal{A}) \ \times \ A \underset{g}{\otimes} I_{n_B} \tag{2.28}$$

– Décomposition en Facteurs Normaux II

$$A(\mathcal{B}) \underset{g}{\otimes} B = A(\mathcal{B}) \underset{g}{\otimes} I_{n_B} \ \times \ I_{n_A} \underset{g}{\otimes} B \tag{2.29}$$

– Pseudo-Commutativité

$$A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A}) = P_\sigma \times \ B(\mathcal{A}) \underset{g}{\otimes} A(\mathcal{B}) \ \times {P_\sigma}^T \tag{2.30}$$

– Décomposition en Produits Tensoriels Classiques

$$A \underset{g}{\otimes} B(\mathcal{A}) = \sum_{k=1}^{n_A} \ell_k(A) \otimes B(a_k) \tag{2.31}$$

2.2.2 Distributivité sur la Somme

Considérons quatre matrices carrées ($A(\mathcal{C},\mathcal{D})$, $B(\mathcal{C},\mathcal{D})$, $C(\mathcal{A},\mathcal{B})$ et $D(\mathcal{A},\mathcal{B})$), on montre que:

$$\begin{aligned}[A(\mathcal{C},\mathcal{D}) + B(\mathcal{C},\mathcal{D})] \underset{g}{\otimes} [C(\mathcal{A},\mathcal{B}) + D(\mathcal{A},\mathcal{B})] = \\ A(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B}) \ + \ A(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} D(\mathcal{A},\mathcal{B}) \ + \\ B(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B}) \ + \ B(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} D(\mathcal{A},\mathcal{B})\end{aligned} \tag{2.32}$$

Démonstration Algébrique

La définition algébrique des éléments de la matrice résultante du membre gauche de l'équation est[4]:

$$[\ a_{i,j}(c_k,d_k) + b_{i,j}(c_k,d_k)\ ] \ \times \ [\ c_{k,l}(a_i,b_i) + d_{k,l}(a_i,b_i)\ ] \quad \text{avec } i,j \in [1..n_A] \text{ et } k,l \in [1..n_C]$$

La définition algébrique des éléments de la matrice résultante du membre droit de l'équation est:

$$\begin{aligned}a_{i,j}(c_k,d_k)c_{k,l}(a_i,b_i) \ + \ a_{i,j}(c_k,d_k)d_{k,l}(a_i,b_i) + \\ b_{i,j}(c_k,d_k)c_{k,l}(a_i,b_i) \ + \ b_{i,j}(c_k,d_k)d_{k,l}(a_i,b_i)\end{aligned} \quad \text{avec } i,j \in [1..n_A] \text{ et } k,l \in [1..n_C]$$

❑

[4]Notons que pour que la somme traditionnelle de matrices $A(\mathcal{C},\mathcal{D})$ et $B(\mathcal{C},\mathcal{D})$, ainsi que celle des matrices $C(\mathcal{A},\mathcal{B})$ et $D(\mathcal{A},\mathcal{B})$, soit possible il faut que $n_A = n_B$ et que $n_C = n_D$

2.2.3 Associativité

Associativité du produit tensoriel généralisé

Considérons trois matrices carrées ($A(\mathcal{B},\mathcal{C})$, $B(\mathcal{A},\mathcal{C})$ et $C(\mathcal{A},\mathcal{B})$), on a:

$$[\ A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} B(\mathcal{A},\mathcal{C})\ ] \underset{g}{\otimes} C(\mathcal{A},\mathcal{B}) = A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} [\ B(\mathcal{A},\mathcal{C}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})\ ]$$

Démonstration Algébrique

Appelons:

- $D(\mathcal{C})$ le produit tensoriel généralisé $A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} B(\mathcal{A},\mathcal{C})$;
- $E(\mathcal{A})$ le produit tensoriel généralisé $B(\mathcal{A},\mathcal{C}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})$;
- H le produit tensoriel généralisé $[\ A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} B(\mathcal{A},\mathcal{C})\ ] \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})$;
- H' le produit tensoriel généralisé $A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} [\ B(\mathcal{A},\mathcal{C}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})\ ]$;

En calculant les éléments de la matrice $D(\mathcal{C}) = A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} B(\mathcal{A},\mathcal{C})$:

- $i,j \in [1..n_a]$, $k,l \in [1..n_b]$:
  $d_{[i,k],[j,l]}(\mathcal{C}) = a_{i,j}(b_k,\mathcal{C})b_{k,l}(a_i,\mathcal{C})$;

En calculant les éléments de la matrice $H = D(\mathcal{C}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})$:

- $i,j \in [1..n_a]$, $k,l \in [1..n_b]$, $m,n \in [1..n_c]$:
  $h_{[i,k,m],[j,l,n]} = d_{[i,k],[j,l]}(c_m)c_{m,n}(a_i,b_k)$;

En remplaçant les éléments de D en H:

- $i,j \in [1..n_a]$, $k,l \in [1..n_b]$, $m,n \in [1..n_c]$:
  $\boxed{h_{[i,k,m],[j,l,n]} = a_{i,j}(b_k,c_m)b_{k,l}(a_i,c_m)c_{m,n}(a_i,b_k)}$

En calculant les éléments de la matrice $E(\mathcal{A}) = B(\mathcal{A},\mathcal{C}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})$:

- $i,j \in [1..n_b]$, $k,l \in [1..n_c]$:
  $e_{[i,k],[j,l]}(\mathcal{A}) = b_{i,j}(\mathcal{A},c_k)c_{k,l}(\mathcal{A},b_i)$;

En calculant les éléments de la matrice $H' = A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} E(\mathcal{A})$:

- $i,j \in [1..n_a]$, $k,l \in [1..n_b]$, $m,n \in [1..n_c]$:
  $h'_{[i,k,m],[j,l,n]} = a_{i,j}(b_k,c_m)e_{[k,m],[l,n]}(a_i)$;

En remplaçant les éléments de E en H':

- $i,j \in [1..n_a]$, $k,l \in [1..n_b]$, $m,n \in [1..n_c]$:
  $\boxed{h'_{[i,k,m],[j,l,n]} = a_{i,j}(b_k,c_m)b_{k,l}(a_i,c_m)c_{m,n}(a_i,b_k)}$ ❑

Associativité de la somme tensorielle généralisée

Considérons trois matrices carrées ($A(\mathcal{B},\mathcal{C})$, $B(\mathcal{A},\mathcal{C})$ et $C(\mathcal{A},\mathcal{B})$), on a:

$$[\ A(\mathcal{B},\mathcal{C}) \underset{g}{\oplus} B(\mathcal{A},\mathcal{C})\ ] \underset{g}{\oplus} C(\mathcal{A},\mathcal{B}) = A(\mathcal{B},\mathcal{C}) \underset{g}{\oplus} [\ B(\mathcal{A},\mathcal{C}) \underset{g}{\oplus} C(\mathcal{A},\mathcal{B})\ ]$$

Démonstration

En évaluant $A(\mathcal{B},\mathcal{C}) \underset{g}{\oplus} B(\mathcal{A},\mathcal{C})$ dans le membre gauche de la formule, il vient:

$$\left[A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} I_{n_B} \ + \ I_{n_A} \underset{g}{\otimes} B(\mathcal{A},\mathcal{C})\right] \underset{g}{\oplus} C(\mathcal{A},\mathcal{B})$$

En évaluant la deuxième somme tensorielle généralisée toujours du membre gauche de l'équation, il vient:

$$\left[A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} I_{n_B} \ + \ I_{n_A} \underset{g}{\otimes} B(\mathcal{A},\mathcal{C})\right] \underset{g}{\otimes} I_{n_C} \ + \ I_{n_A n_B} \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})$$

En utilisant la propriété de distributivité sur la somme par l'identité:

$$\left[A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} I_{n_B}\right] \underset{g}{\otimes} I_{n_C} + \left[I_{n_A} \underset{g}{\otimes} B(\mathcal{A},\mathcal{C})\right] \underset{g}{\otimes} I_{n_C} \ + \ I_{n_A n_B} \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})$$

En sachant que le produit tensoriel généralisé est associatif:

$$\boxed{A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} I_{n_B n_C} \ + \ I_{n_A} \underset{g}{\otimes} B(\mathcal{A},\mathcal{C}) \underset{g}{\otimes} I_{n_C} \ + \ I_{n_A n_B} \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})}$$

En évaluant $B(\mathcal{A},\mathcal{C}) \underset{g}{\oplus} C(\mathcal{A},\mathcal{B})$ dans le membre droit de la formule:

$$A(\mathcal{B},\mathcal{C}) \underset{g}{\oplus} \left[B(\mathcal{A},\mathcal{C}) \underset{g}{\otimes} I_{n_C} \ + \ I_{n_B} \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})\right]$$

En évaluant la deuxième somme tensorielle généralisée:

$$A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} I_{n_B n_C} + I_{n_A} \underset{g}{\otimes} \left[B(\mathcal{A},\mathcal{C}) \underset{g}{\otimes} I_{n_C} \ + \ I_{n_B} \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})\right]$$

En utilisant la propriété de distributivité sur la somme par l'identité:

$$A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} I_{n_B n_C} + I_{n_A} \underset{g}{\otimes} \left[B(\mathcal{A},\mathcal{C}) \underset{g}{\otimes} I_{n_C}\right] + I_{n_A} \underset{g}{\otimes} \left[I_{n_B} \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})\right]$$

En sachant que le produit tensoriel généralisé est associatif:

$$\boxed{A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} I_{n_B n_C} \ + \ I_{n_A} \underset{g}{\otimes} B(\mathcal{A},\mathcal{C}) \underset{g}{\otimes} I_{n_C} \ + \ I_{n_A n_B} \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})} \qquad \square$$

Opérateurs de Suites de Matrices

Les propriétés d'associativité étant définies, il est possible de définir de façon unique les opérateurs[5]:

$$\bigotimes_{i=1}^{N}{}_{g} A^{(i)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(N)}) \text{ et}$$

$$\bigoplus_{i=1}^{N}{}_{g} A^{(i)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(N)}) = \sum_{i=1}^{N} \left[I_{nleft_i} \underset{g}{\otimes} A^{(i)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(N)}) \underset{g}{\otimes} I_{nright_i} \right]$$

Il est possible redéfinir algébriquement les produits tensoriels généralisés (équation 2.20) pour $A = \bigotimes_{g\,k=1}^{N} A^{(k)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(N)})$ par:

$$a_{[i_1,\ldots,i_N][j_1,\ldots,j_N]} = \prod_{k=1}^{N} a^{(k)}_{i_k,j_k}(a^{(1)}_{i_k}, \ldots, a^{(N)}_{i_k}) \tag{2.33}$$

La définition algébrique des sommes tensorielles généralisées (équation 2.22) pour $A = \bigoplus_{g\,k=1}^{N} A^{(k)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(N)})$ devient:

$$a_{[i_1,\ldots,i_N][j_1,\ldots,j_N]} = \sum_{k=1}^{N} \left[a^{(k)}_{i_k,j_k}(a^{(1)}_{i_k}, \ldots, a^{(N)}_{i_k}) \prod_{l=1,l\neq k}^{N} \delta i_l, j_l \right] \tag{2.34}$$

2.2.4 Distributivité sur la Multiplication par l'identité

Distributivité sur la multiplication par l'identité à droite

Considérons trois matrices carrées $A(\mathcal{C})$, $B(\mathcal{C})$ et C, on montre que:

$$[\ A(\mathcal{C}) \times B(\mathcal{C})\ ] \underset{g}{\otimes} I_{n_C} = A(\mathcal{C}) \underset{g}{\otimes} I_{n_C} \ \times\ B(\mathcal{C}) \underset{g}{\otimes} I_{n_C}$$

Démonstration Algébrique

Appelons

- $D(\mathcal{C})$ le produit ordinaire $A(\mathcal{C}) \times B(\mathcal{C})$;
- E le produit tensoriel généralisé $A(\mathcal{C}) \underset{g}{\otimes} I_{n_C}$;

[5]Une matrice $A^{(k)}$ dépendant d'elle même ($A^{(k)}(\mathcal{A}^{(k)})$) est, en effet une matrice constante. Or, chacun des éléments fonctionnels $a^{(k)}_{i,j}(\mathcal{A}^{(k)})$ peuvent être remplacés par les éléments constants $a^{(k)}_{i,j}(a_i)$. Cette remarque est valide pour exclure une matrice de son propre jeu de paramètres. Néanmoins, pour simplifier les notations, on peut utiliser la représentation de cette dépendance fictive d'une matrice par elle même.

– F le produit tensoriel généralisé $B(\mathcal{C}) \underset{g}{\otimes} I_{n_C}$;

– H le produit tensoriel généralisé $[\ A(\mathcal{C}) \times B(\mathcal{C})\ ] \underset{g}{\otimes} I_{n_C} = D(\mathcal{C}) \underset{g}{\otimes} I_{n_C}$;

– H' le produit ordinaire $A(\mathcal{C}) \underset{g}{\otimes} I_{n_C} \times B(\mathcal{C}) \underset{g}{\otimes} I_{n_C} = E \times F$;

En calculant les éléments de la matrice $D(\mathcal{C}) = A(\mathcal{C}) \times B(\mathcal{C})$:

– $i, j \in [1..n_A]$:
$d_{i,j}(\mathcal{C}) = \sum_{r=1}^{n_A} a_{i,r}(\mathcal{C}) b_{r,j}(\mathcal{C})$;

En calculant les éléments de la matrice $H = (D(\mathcal{C}) \underset{g}{\otimes} I_{n_C})$:

– $i, j \in [1..n_A], k, l \in [1..n_C]$:
$h_{[i,k],[j,l]} = d_{i,j}(c_k)\delta_{k,l}$;

En remplaçant les éléments de D en H:

– $i, j \in [1..n_A], k, l \in [1..n_C]$:
$\boxed{h_{[i,k],[j,l]} = \delta_{k,l} \sum_{r=1}^{n_A} a_{i,r}(c_k) b_{r,j}(c_k)}$

En calculant les éléments de la matrice $E = (A(\mathcal{C}) \underset{g}{\otimes} I_{n_C})$:

– $i, j \in [1..n_A], k, l \in [1..n_C]$:
$e_{[i,k],[j,l]} = a_{i,j}(c_k)\delta_{k,l}$;

En calculant les éléments de la matrice $F = (B(\mathcal{C}) \underset{g}{\otimes} I_{n_C})$:

– $i, j \in [1..n_A], k, l \in [1..n_C]$:
$f_{[i,k],[j,l]} = b_{i,j}(c_k)\delta_{k,l}$;

En calculant les éléments de la matrice $H' = E \times F$:

– $i, j \in [1..n_A], k, l \in [1..n_C]$:
$h'_{[i,k],[j,l]} = \sum_{r=1}^{n_A} \left(\sum_{s=1}^{n_C} e_{[i,k],[r,s]} f_{[r,s],[j,l]} \right)$;

En remplaçant les éléments de E et F en H':

– $i, j \in [1..n_A], k, l \in [1..n_C]$:
$h'_{[i,k],[j,l]} = \sum_{r=1}^{n_A} \left(\sum_{s=1}^{n_C} a_{i,r}(c_k)\delta_{k,s} b_{r,j}(c_s)\delta_{s,l} \right)$;

En remplaçant les éléments connus (δ) en H':

– $i, j \in [1..n_A], k, l \in [1..n_C]$:
$\boxed{h'_{[i,k],[j,l]} = \delta_{k,l} \sum_{r=1}^{n_A} a_{i,r}(c_k) b_{r,j}(c_k)}$ ❑

Distributivité sur la multiplication par l'identité à gauche

Considérons trois matrices carrées $A(\mathcal{C})$, $B(\mathcal{C})$ et C, on montre que:

$$I_{n_C} \underset{g}{\otimes} [\ A(\mathcal{C}) \times B(\mathcal{C})\ ] = I_{n_C} \underset{g}{\otimes} A(\mathcal{C}) \times I_{n_C} \underset{g}{\otimes} B(\mathcal{C})$$

Démonstration Algébrique

Appelons

- $D(\mathcal{C})$ le produit ordinaire $A(\mathcal{C}) \times B(\mathcal{C})$;
- E le produit tensoriel généralisé $I_{n_C} \underset{g}{\otimes} A(\mathcal{C})$;
- F le produit tensoriel généralisé $I_{n_C} \underset{g}{\otimes} B(\mathcal{C})$;
- H le produit tensoriel généralisé $I_{n_C} \underset{g}{\otimes} [\ A(\mathcal{C}) \times B(\mathcal{C})\ ] = I_{n_C} \underset{g}{\otimes} D(\mathcal{C})$;
- H' le produit ordinaire $I_{n_C} \underset{g}{\otimes} A(\mathcal{C}) \ \times \ I_{n_C} \underset{g}{\otimes} B(\mathcal{C}) = E \times F$;

En calculant les éléments de la matrice $D(\mathcal{C}) = A(\mathcal{C}) \times B(\mathcal{C})$:

- $i, j \in [1..n_A]$:
  $d_{i,j}(\mathcal{C}) = \sum_{r=1}^{n_A} a_{i,r}(\mathcal{C}) b_{r,j}(\mathcal{C})$;

En calculant les éléments de la matrice $H = I_{n_C} \underset{g}{\otimes} D(\mathcal{C})$:

- $i, j \in [1..n_C], k, l \in [1..n_A]$:
  $h_{[i,k],[j,l]} = \delta_{i,j} d_{k,l}(c_i)$;

En remplaçant les éléments de D en H:

- $i, j \in [1..n_C], k, l \in [1..n_A]$:
  $\boxed{h_{[i,k],[j,l]} = \delta_{i,j} \sum_{s=1}^{n_A} a_{k,s}(c_i) b_{s,l}(c_i)}$

En calculant les éléments de la matrice $E = I_{n_C} \underset{g}{\otimes} A(\mathcal{C})$:

- $i, j \in [1..n_C], k, l \in [1..n_A]$:
  $e_{[i,k],[j,l]} = \delta_{i,j} a_{k,l}(c_i)$;

En calculant les éléments de la matrice $F = I_{n_C} \underset{g}{\otimes} B(\mathcal{C})$:

- $i, j \in [1..n_C], k, l \in [1..n_A]$:
  $f_{[i,k],[j,l]} = \delta_{i,j} b_{k,l}(c_i)$;

En calculant les éléments de la matrice $H' = E \times F$:

- $i, j \in [1..n_C], k, l \in [1..n_A]$:
  $h'_{[i,k],[j,l]} = \sum_{r=1}^{n_C} \left(\sum_{s=1}^{n_A} e_{[i,k],[r,s]} f_{[r,s],[j,l]} \right)$;

En remplaçant les éléments de E et F en H':

- $i, j \in [1..n_C], k, l \in [1..n_A]$:
  $h'_{[i,k],[j,l]} = \sum_{r=1}^{n_C} \left(\sum_{s=1}^{n_A} \delta_{i,r} a_{k,s}(c_i) \delta_{r,j} b_{s,l}(c_r) \right)$;

En remplaçant les éléments connus (δ) en H':

- $i, j \in [1..n_A], k, l \in [1..n_C]$:
  $\boxed{h'_{[i,k],[j,l]} = \delta_{i,j} \sum_{s=1}^{n_A} a_{k,s}(c_i) b_{s,l}(c_i)}$

❑

2.2.5 Décomposition en Facteurs Normaux I

Considérons deux matrices A et $B(\mathcal{A})$, on montre que:

$$A \underset{g}{\otimes} B(\mathcal{A}) = I_{n_A} \underset{g}{\otimes} B(\mathcal{A}) \times A \underset{g}{\otimes} I_{n_B}$$

Démonstration Algébrique

Appelons:

– C le produit tensoriel généralisé $I_{n_A} \underset{g}{\otimes} B(\mathcal{A})$

– D le produit tensoriel généralisé $A \underset{g}{\otimes} I_{n_B}$

– H le produit tensoriel généralisé $A \underset{g}{\otimes} B(\mathcal{A})$

– H' le produit ordinaire $I_{n_A} \underset{g}{\otimes} B(\mathcal{A}) \times A \underset{g}{\otimes} I_{n_B} = C \times D$

En calculant les éléments de la matrice $H = A \underset{g}{\otimes} B(\mathcal{A})$:

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$$\boxed{h_{[i,k],[j,l]} = a_{i,j} b_{k,l}(a_i)}$$

En calculant les éléments de la matrice $C = I_{n_A} \underset{g}{\otimes} B(\mathcal{A})$:

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$c_{[i,k],[j,l]} = \delta_{i,j} b_{k,l}(a_i)$;

En calculant les éléments de la matrice $D = A \underset{g}{\otimes} I_{n_B}$:

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$d_{[i,k],[j,l]} = a_{i,j} \delta_{k,l}$;

En calculant les éléments de la matrice $H' = C \times D$:

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$h'_{[i,k],[j,l]} = \sum_{r=1}^{n_A} \left(\sum_{s=1}^{n_B} c_{[i,k],[r,s]} d_{[r,s],[j,l]} \right)$;

En remplaçant les éléments de C et D en H':

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$h'_{[i,k],[j,l]} = \sum_{r=1}^{n_A} (\sum_{s=1}^{n_B} \delta_{i,r} b_{k,s}(a_i) a_{r,j} \delta_{s,l})$;

En remplaçant les éléments connus (δ) en H':

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$$\boxed{h'_{[i,k],[j,l]} = b_{k,l}(a_i) a_{i,j}}$$

❑

Cette démonstration est facilement visualisée par les matrices résultantes de chaque membre de l'équation. La matrice résultante du membre gauche de l'équation est:

$$A \underset{g}{\otimes} B(\mathcal{A}) = \begin{pmatrix} a_{1,1}B(a_1) & a_{1,2}B(a_1) & \cdots & a_{1,n_a}B(a_1) \\ a_{2,1}B(a_2) & a_{2,2}B(a_2) & \cdots & a_{2,n_a}B(a_2) \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_A,1}B(a_{n_A}) & a_{n_A,2}B(a_{n_A}) & \cdots & a_{n_A,n_A}B(a_{n_A}) \end{pmatrix}$$

La matrice résultante du membre droit de l'équation est:

$$I_{n_A} \underset{g}{\otimes} B(\mathcal{A}) \times A \underset{g}{\otimes} I_{n_B} =$$

$$\begin{pmatrix} B(a_1) & & & \\ & B(a_2) & & \\ & & \ddots & \\ & & & B(a_{n_A}) \end{pmatrix} \times \begin{pmatrix} a_{1,1}I_{n_B} & a_{1,2}I_{n_B} & \cdots & a_{1,n_A}I_{n_B} \\ a_{2,1}I_{n_B} & a_{2,2}I_{n_B} & \cdots & a_{2,n_A}I_{n_B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_A,1}I_{n_B} & a_{n_A,2}I_{n_B} & \cdots & a_{n_A,n_A}I_{n_B} \end{pmatrix}$$

Généralisation pour les Suites de Matrices

Il est possible de généraliser cette propriété pour une suite de N matrices dans les termes suivants:

$$\begin{aligned} A^{(1)} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} A^{(3)}(\mathcal{A}^{(1)},\mathcal{A}^{(2)}) \underset{g}{\otimes} \cdots \underset{g}{\otimes} A^{(N)}(\mathcal{A}^{(1)},\ldots,\mathcal{A}^{(N-1)}) = \\ I_{nleft_N} \underset{g}{\otimes} A^{(N)}(\mathcal{A}^{(1)},\ldots,\mathcal{A}^{(N-1)}) \\ \times I_{nleft_{N-1}} \underset{g}{\otimes} A^{(N-1)}(\mathcal{A}^{(1)},\ldots,\mathcal{A}^{(N-2)}) \underset{g}{\otimes} I_{n_N} \\ \times \cdots \\ \times I_{n_1} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} I_{nright_2} \\ \times A^{(1)} \underset{g}{\otimes} I_{nright_1} \end{aligned} \tag{2.35}$$

Démonstration par Récurrence

Appelons:

- $H^{(1)}$ la matrice constante $A^{(1)}$;
- $H^{(2)}$ le produit tensoriel généralisé $H^{(1)} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(1)})$ (une matrice constante de dimension $n_1 n_2 = nleft_3$);
- $H^{(k)}$ le produit tensoriel généralisé $H^{(k-1)} \underset{g}{\otimes} A^{(k)}(\mathcal{A}^{(1)},\ldots,\mathcal{A}^{(k-1)})$ (une matrice constante de dimension $n_1 n_2 \ldots n_k = nleft_{k+1}$);

En utilisant la propriété de décomposition en facteurs normaux (équation 2.28) sur $H^{(2)}$:

$$H^{(2)} = I_{n_1} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(1)}) \times H^{(1)} \underset{g}{\otimes} I_{n_2} = I_{n_1} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(1)}) \times A^{(1)} \underset{g}{\otimes} I_{n_2}$$

En utilisant la propriété de décomposition en facteurs normaux (équation 2.28) sur $H^{(3)}$:

$$H^{(3)} = H^{(2)} \underset{g}{\otimes} A^{(3)}(\mathcal{A}^{(1)}, \mathcal{A}^{(2)}) = I_{nleft_3} \underset{g}{\otimes} A^{(3)}(\mathcal{A}^{(1)}, \mathcal{A}^{(2)}) \times H^{(2)} \underset{g}{\otimes} I_{n_3}$$

En remplaçant $H^{(2)}$:

$$H^{(3)} = I_{nleft_3} \underset{g}{\otimes} A^{(3)}(\mathcal{A}^{(1)}, \mathcal{A}^{(2)}) \times \left[I_{n_1} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(1)}) \times A^{(1)} \underset{g}{\otimes} I_{n_2} \right] \underset{g}{\otimes} I_{n_3}$$

En utilisant la propriété d'associativité (équation 2.24) et la propriété de distributivité sur la multiplication par l'identité à droite (équation 2.26):

$$H^{(3)} = I_{nleft_3} \underset{g}{\otimes} A^{(3)}(\mathcal{A}^{(1)}, \mathcal{A}^{(2)}) \times I_{n_1} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} I_{n_3} \times A^{(1)} \underset{g}{\otimes} I_{n_2} \underset{g}{\otimes} I_{n_3}$$

En sachant que $I_{n_2} \underset{g}{\otimes} I_{n_3} = I_{n_2 n_3} = I_{nright_1}$:

$$H^{(3)} = I_{nleft_3} \underset{g}{\otimes} A^{(3)}(\mathcal{A}^{(1)}, \mathcal{A}^{(2)}) \times I_{n_1} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} I_{n_3} \times A^{(1)} \underset{g}{\otimes} I_{nright_1}$$

Supposons la relation vrai pour $H^{(k)}$, alors la relation est vrai pour $H^{(k+1)}$:

$$H^{(k+1)} = H^{(k)} \underset{g}{\otimes} A^{(k+1)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(k)})$$

En utilisant la propriété de décomposition en facteurs normaux (équation 2.28):

$$H^{(k+1)} = I_{nleft_{k+1}} \underset{g}{\otimes} A^{(k+1)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(k)}) \times H^{(k)} \underset{g}{\otimes} I_{n_{k+1}}$$

En remplaçant $H^{(k)}$ et en utilisant la propriétés d'associativité (équation 2.24) et la propriété de distributivité sur la multiplication par l'identité à droite (équation 2.26):

$$\begin{aligned} H^{(k+1)} = & I_{nleft_{k+1}} \underset{g}{\otimes} A^{(k+1)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(k)}) \\ & \times I_{nleft_k} \underset{g}{\otimes} A^{(k)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(k-1)}) \underset{g}{\otimes} I_{n_{k-1}} \\ & \times I_{nleft_{k-1}} \underset{g}{\otimes} A^{(k-1)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(k-2)}) \underset{g}{\otimes} I_{nright_{k-1}} \\ & \times \cdots \\ & \times I_{n_1} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} I_{nright_2} \\ & \times A^{(1)} \underset{g}{\otimes} I_{nright_1} \end{aligned}$$

❑

2.2.6 Décomposition en Facteurs Normaux II

Considérons deux matrices $A(\mathcal{B})$ et B, on montre que:

$$A(\mathcal{B}) \underset{g}{\otimes} B = A(\mathcal{B}) \underset{g}{\otimes} I_{n_B} \times I_{n_A} \underset{g}{\otimes} B$$

Démonstration Algébrique

Appelons:

- C le produit tensoriel généralisé $A(\mathcal{B}) \underset{g}{\otimes} I_{n_B}$
- D le produit tensoriel généralisé $I_{n_A} \underset{g}{\otimes} B$
- H le produit tensoriel généralisé $A(\mathcal{B}) \underset{g}{\otimes} B$
- H' le produit ordinaire $A(\mathcal{B}) \underset{g}{\otimes} I_{n_B} \times I_{n_A} \underset{g}{\otimes} B = C \times D$

En calculant les éléments de la matrice $H = A(\mathcal{B}) \underset{g}{\otimes} B$:

- $i, j \in [1..n_A], k, l \in [1..n_B]$:
  $\boxed{h_{[i,k],[j,l]} = a_{i,j}(b_k)b_{k,l}}$

En calculant les éléments de la matrice $C = A(\mathcal{B}) \underset{g}{\otimes} I_{n_B}$:

- $i, j \in [1..n_A], k, l \in [1..n_B]$:
  $c_{[i,k],[j,l]} = a_{i,j}(b_k)\delta_{k,l}$;

En calculant les éléments de la matrice $D = I_{n_A} \underset{g}{\otimes} B$:

- $i, j \in [1..n_A], k, l \in [1..n_B]$:
  $c_{[i,k],[j,l]} = \delta_{i,j}b_{k,l}$;

En calculant les éléments de la matrice $H' = C \times D$:

- $i, j \in [1..n_A], k, l \in [1..n_B]$:
  $h'_{[i,k],[j,l]} = \sum_{r=1}^{n_A} \left(\sum_{s=1}^{n_B} c_{[i,k],[r,s]} d_{[r,s],[j,l]} \right)$;

En remplaçant les éléments de C et D en H':

- $i, j \in [1..n_A], k, l \in [1..n_B]$:
  $h'_{[i,k],[j,l]} = \sum_{r=1}^{n_A} (\sum_{s=1}^{n_B} a_{i,r}(b_k)\delta_{k,s}\delta_{r,j}b_{s,l})$;

En remplaçant les éléments connus (δ) en H':

- $i, j \in [1..n_A], k, l \in [1..n_B]$:
  $\boxed{h'_{[i,k],[j,l]} = a_{i,j}(b_k)b_{k,l}}$ ❑

Cette démonstration est facilement visualisée par les matrices résultantes de chaque membre de l'équation. La matrice résultante du membre gauche de l'équation est:

$$A(\mathcal{B}) \otimes_g B = \begin{pmatrix} a_{1,1}(\mathcal{B})I_{n_B} \times B & a_{1,2}(\mathcal{B})I_{n_B} \times B & \cdots & a_{1,n_A}(\mathcal{B})I_{n_B} \times B \\ a_{2,1}(\mathcal{B})I_{n_B} \times B & a_{2,2}(\mathcal{B})I_{n_B} \times B & \cdots & a_{2,n_A}(\mathcal{B})I_{n_B} \times B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_A,1}(\mathcal{B})I_{n_B} \times B & a_{n_A,2}(\mathcal{B})I_{n_B} \times B & \cdots & a_{n_A,n_A}(\mathcal{B})I_{n_B} \times B \end{pmatrix}$$

La matrice résultante du membre droit de l'équation est:

$$A(\mathcal{B}) \underset{g}{\otimes} I_{n_B} \times I_{n_A} \underset{g}{\otimes} B =$$

$$\begin{pmatrix} a_{1,1}(\mathcal{B})I_{n_B} & a_{1,2}(\mathcal{B})I_{n_B} & \cdots & a_{1,n_A}(\mathcal{B})I_{n_B} \\ a_{2,1}(\mathcal{B})I_{n_B} & a_{2,2}(\mathcal{B})I_{n_B} & \cdots & a_{2,n_A}(\mathcal{B})I_{n_B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_A,1}(\mathcal{B})I_{n_B} & a_{n_A,2}(\mathcal{B})I_{n_B} & \cdots & a_{n_A,n_A}(\mathcal{B})I_{n_B} \end{pmatrix} \times \begin{pmatrix} B & & & \\ & B & & \\ & & \ddots & \\ & & & B \end{pmatrix}$$

Il faut remarquer que la propriété de commutativité de facteurs normaux pour les produits tensoriels classiques (section 2.1.3, équation 2.18), n'est pas vérifié pour les produits tensoriels généralisés.

Généralisation pour les Suites de Matrices

A l'instar de la propriété précédente (2.28), il est possible de généraliser cette propriété (2.29) pour une suite de N matrices dans les termes suivants:

$$\begin{aligned} A^{(1)}(\mathcal{A}^{(2)}, \ldots, \mathcal{A}^{(N)}) \underset{g}{\otimes} \cdots \underset{g}{\otimes} A^{(N-2)}(\mathcal{A}^{(N-1)}, \mathcal{A}^{(N)}) \underset{g}{\otimes} A^{(N-1)}(\mathcal{A}^{(N)}) \underset{g}{\otimes} A^{(N)} = \\ A^{(1)}(\mathcal{A}^{(2)}, \ldots, \mathcal{A}^{(N)}) \underset{g}{\otimes} I_{nright_1} \\ \times I_{n_1} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(3)}, \ldots, \mathcal{A}^{(N)}) \underset{g}{\otimes} I_{nright_2} \\ \times \cdots \\ \times I_{nleft_{N-1}} \underset{g}{\otimes} A^{(N-1)}(\mathcal{A}^{(N)}) \underset{g}{\otimes} I_{n_N} \\ \times I_{nleft_N} \underset{g}{\otimes} A^{(N)} \end{aligned} \tag{2.36}$$

Démonstration par Récurrence

Appelons:

- $H^{(N)}$ la matrice constante $A^{(N)}$;
- $H^{(N-1)}$ le produit tensoriel généralisé $A^{(N-1)}(\mathcal{A}^{(N)}) \underset{g}{\otimes} H^{(N)}$ (aussi une matrice constante de dimension $n_N n_{N-1} = nright_{N-2}$);

– $H^{(k)}$ le produit tensoriel généralisé $A^{(k)}(\mathcal{A}^{(k+1)}, \ldots, \mathcal{A}^{(N)}) \underset{g}{\otimes} H^{(k+1)}$ (aussi une matrice constante de dimension $n_k n_{k+1} \ldots n_N = nright_{k-1}$);

En utilisant la propriété de décomposition en facteurs normaux (équation 2.29) sur $H^{(N-1)}$:

$$\begin{aligned} H^{(N-1)} =& A^{(N-1)}(\mathcal{A}^{(N)}) \underset{g}{\otimes} H^{(N)} \\ =& A^{(N-1)}(\mathcal{A}^{(N)}) \underset{g}{\otimes} I_{n_N} \times I_{n_{N-1}} \underset{g}{\otimes} H^{(N)} \end{aligned}$$

En remplaçant $H^{(N)}$:

$$H^{(N-1)} = A^{(N-1)}(\mathcal{A}^{(N)}) \underset{g}{\otimes} I_{n_N} \times I_{n_{N-1}} \underset{g}{\otimes} A^{(N)}$$

En utilisant la propriété de décomposition en facteurs normaux (équation 2.29) sur $H^{(N-2)}$:

$$\begin{aligned} H^{(N-2)} =& A^{(N-2)}(\mathcal{A}^{(N-1)}, \mathcal{A}^{(N)}) \underset{g}{\otimes} H^{(N-1)} \\ =& A^{(N-2)}(\mathcal{A}^{(N-1)}, \mathcal{A}^{(N)}) \underset{g}{\otimes} I_{nright_{N-2}} \times I_{n_{N-2}} \underset{g}{\otimes} H^{(N-1)} \end{aligned}$$

En remplaçant $H^{(N-1)}$:

$$\begin{aligned} H^{(N-2)} =& A^{(N-2)}(\mathcal{A}^{(N-1)}, \mathcal{A}^{(N)}) \underset{g}{\otimes} I_{nright_{N-2}} \\ & \times I_{n_{N-2}} \underset{g}{\otimes} \left[A^{(N-1)}(\mathcal{A}^{(N)}) \underset{g}{\otimes} I_{n_N} \times I_{n_{N-1}} \underset{g}{\otimes} A^{(N)} \right] \end{aligned}$$

En utilisant la propriété d'associativité (équation 2.24) et la propriété de distributivité sur la multiplication par l'identité à gauche (équation 2.27):

$$\begin{aligned} H^{(N-2)} =& A^{(N-2)}(\mathcal{A}^{(N-1)}, \mathcal{A}^{(N)}) \underset{g}{\otimes} I_{nright_{N-2}} \\ & \times I_{n_{N-2}} \underset{g}{\otimes} A^{(N-1)}(\mathcal{A}^{(N)}) \underset{g}{\otimes} I_{n_N} \times I_{n_{N-2}} \underset{g}{\otimes} I_{n_{N-1}} \underset{g}{\otimes} A^{(N)} \end{aligned}$$

En sachant que $I_{n_{N-2}} \underset{g}{\otimes} I_{n_{N-1}} = I_{n_{N-2} n_{N-1}}$:

$$\begin{aligned} H^{(N-2)} =& A^{(N-2)}(\mathcal{A}^{(N-1)}, \mathcal{A}^{(N)}) \underset{g}{\otimes} I_{nright_{N-2}} \\ & \times I_{n_{N-2}} \underset{g}{\otimes} A^{(N-1)}(\mathcal{A}^{(N)}) \underset{g}{\otimes} I_{n_N} \times I_{n_{N-2} n_{N-1}} \underset{g}{\otimes} A^{(N)} \end{aligned}$$

En appliquant ces mêmes pas jusqu'à $H^{(1)}$ on obtient:

$$\begin{aligned} H^{(1)} =& A^{(1)}(\mathcal{A}^{(2)}, \ldots, \mathcal{A}^{(N)}) \underset{g}{\otimes} I_{n_2 \ldots n_N} \\ & \times I_{n_1} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(3)}, \ldots, \mathcal{A}^{(N)}) \underset{g}{\otimes} I_{n_3 \ldots n_N} \\ & \times \cdots \\ & \times I_{n_1 \ldots n_{N-2}} \underset{g}{\otimes} A^{(N-1)}(\mathcal{A}^{(N)}) \underset{g}{\otimes} I_{n_N} \\ & \times I_{n_1 \ldots n_{N-1}} \underset{g}{\otimes} A^{(N)} \end{aligned}$$

Par définition de $nleft$ et $nright$ on a:

$$\begin{aligned} H^{(1)} =& A^{(1)}(\mathcal{A}^{(2)}, \dots, \mathcal{A}^{(N)}) \underset{g}{\otimes} I_{nright_1} \\ & \times I_{n_1} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(3)}, \dots, \mathcal{A}^{(N)}) \underset{g}{\otimes} I_{nright_2} \\ & \times \cdots \\ & \times I_{nleft_{N-1}} \underset{g}{\otimes} A^{(N-1)}(\mathcal{A}^{(N)}) \underset{g}{\otimes} I_{n_N} \\ & \times I_{nleft_N} \underset{g}{\otimes} A^{(N)} \end{aligned}$$

❑

Généralisation pour une suite de matrices

Connaissant les deux propriétés de décomposition en facteurs normaux (équations 2.28 et 2.29), il est possible de les généraliser à la décomposition en facteurs normaux des suites de matrices. Dans les généralisations présentées dans les équations 2.35 et 2.36, il y avait une matrice constante, une matrice dépendant seulement de la matrice précédente, une matrice dépendant des deux matrices précédentes et ainsi de suite jusqu'à une matrice dépendant de toutes les autres matrices. L'équation 2.35 donne un ordre commençant par la matrice constante jusqu'à la plus dépendante et l'équation 2.36 dans le sens inverse. En vérité, l'application alternée des propriétés de base (2.28 et 2.29) nous permet d'affirmer que pour n'importe quel ordre on peut toujours obtenir une décomposition en facteurs normaux. Le résultat est toujours un produit traditionnel de matrices décrites comme un facteur normal

$$I_{nleft_k} \underset{g}{\otimes} A^{(k)}(\dots) \underset{g}{\otimes} I_{nright_k}$$

L'ordre entre les facteurs normaux sera toujours du facteur contenant la matrice la plus dépendante jusqu'au facteur contenant la matrice constante.

Prenons par exemple, une suite des trois matrices $A^{(1)}$, $A^{(2)}(\mathcal{A}^{(3)})$ et $A^{(3)}(\mathcal{A}^{(1)})$. Le produit tensoriel généralisé de ces trois matrices est décomposable selon la formule:

$$\begin{aligned} A^{(1)} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(3)}) \underset{g}{\otimes} A^{(3)}(\mathcal{A}^{(1)}) = & \\ & I_{n_1} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(3)}) \underset{g}{\otimes} I_{n_3} \\ & \times I_{nleft_3} \underset{g}{\otimes} A^{(3)}(\mathcal{A}^{(1)}) \\ & \times A^{(1)} \underset{g}{\otimes} I_{nright_k} \end{aligned}$$

Prenons maintenant une exemple analogue où l'on change l'ordre des matrices $A^{(1)}$ et $A^{(3)}(\mathcal{A}^{(1)})$ en les appelant $A^{(1)}(\mathcal{A}^{(3)})$ et $A^{(3)}$, par conséquent $A^{(2)}(\mathcal{A}^{(3)})$ devient $A^{(2)}(\mathcal{A}^{(1)})$.

Le produit tensoriel généralisé de ces trois matrices est décomposable selon la formule:

$$\begin{aligned} A^{(1)}(\mathcal{A}^{(3)}) \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} A^{(3)} = \\ I_{n_1} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} I_{n_3} \\ \times A^{(1)}(\mathcal{A}^{(3)}) \underset{g}{\otimes} I_{right_1} \\ \times I_{nleft_k} \underset{g}{\otimes} A^{(3)} \end{aligned}$$

Notons que dans ces deux exemples, la matrice $A^{(2)}$ ne dépend que d'une seule matrice. Pourtant, $A^{(2)}$ est considérée comme la plus dépendante car elle dépend d'une matrice déjà dépendant elle-même d'une autre matrice. Le choix de l'ordre dans lequel les facteurs normaux doivent être multipliés peut être obtenu algorithmiquement par un parcours de graphe (voir la description de l'algorithme de tri topologique [11] dans la page 99).

Certains termes ont des degrés de liberté dans le choix de décomposition en facteurs normaux. La règle générale qui doit être respectée est que pour chaque matrice fonctionnelle, son facteur normal correspondant doit précéder les facteurs normaux correspondant à ses paramètres. Ceci définit un ordre partiel sur les matrices.

Prenons par exemple un terme produit tensoriel avec les dépendances fonctionnelles suivantes:

$$A^{(1)} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} A^{(3)}(\mathcal{A}^{(2)}) \underset{g}{\otimes} A^{(4)}(\mathcal{A}^{(5)}) \underset{g}{\otimes} A^{(5)} \underset{g}{\otimes} A^{(6)}(\mathcal{A}^{(1)})$$

Ce terme peut être décomposé en facteurs normaux selon plusieurs options. D'après les paramètres des matrices, les seules restrictions qui doivent être respectées sont:

- Le facteur correspondant à la deuxième matrice doit précéder le facteur normal correspondant à la première matrice;
- Le facteur correspondant à la troisième matrice doit précéder le facteur normal correspondant à la deuxième matrice;
- Le facteur correspondant à la quatrième matrice doit précéder le facteur normal correspondant à la cinquième matrice;
- Le facteur correspondant à la sixième matrice doit précéder le facteur normal correspondant à la première matrice;

Une décomposition valide est, par exemple:

$$\begin{aligned} A^{(1)} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} A^{(3)}(\mathcal{A}^{(2)}) \underset{g}{\otimes} A^{(4)}(\mathcal{A}^{(5)}) \underset{g}{\otimes} A^{(5)} \underset{g}{\otimes} A^{(6)}(\mathcal{A}^{(1)}) = \\ I_{nleft_3} \underset{g}{\otimes} A^{(3)}(\mathcal{A}^{(2)}) \underset{g}{\otimes} I_{nright_3} \\ \times\ I_{nleft_2} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} I_{nright_2} \\ \times\ I_{nleft_6} \underset{g}{\otimes} A^{(6)}(\mathcal{A}^{(1)}) \\ \times\ A^{(1)} \underset{g}{\otimes} I_{nright_1} \\ \times\ I_{nleft_4} \underset{g}{\otimes} A^{(4)}(\mathcal{A}^{(5)}) \underset{g}{\otimes} I_{nright_4} \\ \times\ I_{nleft_5} \underset{g}{\otimes} A^{(5)} \underset{g}{\otimes} I_{nright_5} \end{aligned}$$

En représentant cette décomposition par la suite $\{3,2,6,1,4,5\}$ (les indices des matrices correspondant aux facteurs normaux dans l'ordre de la décomposition), les autres décompositions possibles sont:

$\{6,3,2,1,4,5\}$	$\{3,6,2,1,4,5\}$	$\{4,3,2,6,1,5\}$	$\{6,4,3,2,1,5\}$	$\{4,6,3,2,1,5\}$
$\{4,3,6,2,1,5\}$	$\{4,5,3,2,6,1\}$	$\{6,4,5,3,2,1\}$	$\{4,6,5,3,2,1\}$	$\{4,5,6,3,2,1\}$
$\{4,5,3,6,2,1\}$	$\{4,3,5,2,6,1\}$	$\{6,4,3,5,2,1\}$	$\{4,6,3,5,2,1\}$	$\{4,3,6,5,2,1\}$
$\{4,3,5,6,2,1\}$	$\{4,3,2,5,6,1\}$	$\{6,4,3,2,5,1\}$	$\{4,3,6,2,5,1\}$	$\{4,3,2,6,5,1\}$
$\{4,3,2,6,5,1\}$	$\{6,4,3,2,5,1\}$	$\{4,6,3,2,5,1\}$	$\{4,3,6,2,5,1\}$	$\{3,4,2,6,1,5\}$
$\{6,3,4,2,1,5\}$	$\{3,6,4,2,1,5\}$	$\{3,4,6,2,1,5\}$	$\{3,4,5,2,6,1\}$	$\{6,3,4,5,2,1\}$
$\{3,6,4,5,2,1\}$	$\{3,4,6,5,2,1\}$	$\{3,4,5,6,2,1\}$	$\{3,4,2,5,6,1\}$	$\{6,3,4,2,5,1\}$
$\{3,6,4,2,5,1\}$	$\{3,4,6,2,5,1\}$	$\{3,4,2,6,5,1\}$	$\{6,3,4,2,5,1\}$	$\{3,6,4,2,5,1\}$
$\{3,4,6,2,5,1\}$	$\{3,2,4,6,1,5\}$	$\{6,3,2,4,1,5\}$	$\{3,6,2,4,1,5\}$	$\{3,2,6,4,1,5\}$
$\{3,2,4,5,6,1\}$	$\{6,3,2,4,5,1\}$	$\{3,6,2,4,5,1\}$	$\{3,2,6,4,5,1\}$	$\{3,2,4,6,5,1\}$
$\{3,2,4,6,5,1\}$	$\{6,3,2,4,5,1\}$	$\{3,6,2,4,5,1\}$	$\{3,2,6,4,5,1\}$	$\{3,2,6,4,1,5\}$
$\{6,3,2,4,1,5\}$	$\{3,6,2,4,1,5\}$	$\{3,2,6,4,5,1\}$	$\{6,3,2,4,5,1\}$	$\{3,6,2,4,5,1\}$

2.2.7 Pseudo-Commutativité

Considérons deux matrices $A(\mathcal{B})$ et $B(\mathcal{A})$ et la transposition σ sur $[1..2]$, il existe une matrice de permutation P_σ tel que:

$$A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A}) = P_\sigma \ \times \ B(\mathcal{A}) \underset{g}{\otimes} A(\mathcal{B}) \ \times \ {P_\sigma}^T$$

Au contraire des autres propriétés présentées, nous allons directement prouver la propriété plus générale que s'applique à une suite de N matrices dans les mêmes termes:

$$\bigotimes_{k=1}^{N}{}_g \, A^{(k)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(N)}) = P_\sigma \ \times \ \bigotimes_{k=1}^{N}{}_g \, A^{(\sigma_k)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(N)}) \ \times \ {P_\sigma}^T \qquad (2.37)$$

Cette démonstration est une généralisation de celle du produit tensoriel classique (équation 2.17) prouvée dans [81]. Rappelons les définitions suivantes:

✍ **Rappelons**

σ une permutation nommée σ sur l'intervalle $[1..N]$, une permutation σ établit un nouvel ordre pour une suite de N matrices;

$\sigma(i)$ le rang de l'automate $\mathcal{A}^{(i)}$ dans l'ordre identifié par la permutation σ;

σ_k l'indice de l'automate placé au rang k de l'ordre identifié par la permutation σ (si $\sigma_k = i$, $\sigma(i) = k$);

$[i_1, \ldots, i_N]_\sigma$ une modification de l'ordre des n-uplets d'une coordonné selon une permutation σ, dans la pratique on change l'intervalle de variation de chacun des indices de façon à que le k-ème indice varie de $[1..n_k]$ vers $[1..n_{\sigma_k}]$;

P_σ la matrice de permutation des composantes d'un vecteur induite par la modification de l'ordre des automates définie par σ, dont les éléments sont définis par:

$$p_{[i_1,\ldots,i_N][j_1,\ldots,j_N]_\sigma} = \begin{cases} 1 & \text{si } j_m = i_{\sigma_m} \text{ avec } m \in [1..N] \\ 0 & \text{sinon} \end{cases};$$

P_σ^T la transposée et aussi l'inverse de la matrice P_σ.

Démonstration Algébrique

Appelons:

- Y le produit tensoriel généralisé, membre gauche de 2.37, $\bigotimes_{g\,k=1}^{N} A^{(k)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(N)})$;
- X le produit tensoriel généralisé $\bigotimes_{g\,k=1}^{N} A^{(\sigma_k)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(N)})$;
- Z le membre droit de 2.37, $P_\sigma X P_\sigma^T = P_\sigma \bigotimes_{g\,k=1}^{N} A^{(\sigma_k)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(N)}) P_\sigma^T$;

Par définition les éléments de Y sont:

- $i_k, j_k \in [1..n_k]$:

  $y_{[i_1,\ldots,i_N][j_1,\ldots,j_N]} = \prod_{k=1}^{N} a^{(k)}_{i_k,j_k}(a^{(1)}_{i_1}, \ldots, a^{(N)}_{i_N})$;

Le produit XP_σ^T permutant les colonnes de X et le produit $P_\sigma X$ permutant les lignes de X, on calcule les éléments de $P_\sigma X P_\sigma^T$ selon:

- $i_{\sigma_k}, j_{\sigma_k} \in [1..n_{\sigma_k}]$:

  $z_{[i_1,\ldots,i_N][j_1,\ldots,j_N]} = x_{[i_{\sigma_1},\ldots,i_{\sigma_N}]\sigma[j_{\sigma_1},\ldots,j_{\sigma_N}]_\sigma}$

  $x_{[i_{\sigma_1},\ldots,i_{\sigma_N}]\sigma[j_{\sigma_1},\ldots,j_{\sigma_N}]_\sigma} = \prod_{k=1}^{N} a^{(\sigma_k)}_{i_{\sigma_k},j_{\sigma_k}}(a^{(1)}_{i_1}, \ldots, a^{(N)}_{i_N})$;

Étant donné que σ est une bijection sur $[1..N]$, il est possible de remplacer σ_k par l et k par $\sigma(l)$:

- $i_l, j_l \in [1..n_l]$:

  $z_[i_1, \ldots, i_N][j_1, \ldots, j_N] = \prod_{l=\sigma(1)}^{\sigma(N)} a^{(l)}_{i_l,j_l}(a^{(1)}_{i_1}, \ldots, a^{(N)}_{i_N})$;

En sachant que l'opérateur $\prod$ est commutatif:

– $i_k, j_k \in [1..n_k]$:

$$\prod_{k=1}^{N} a^{(k)}_{i_k,j_k}(a^{(1)}_{i_1},\ldots,a^{(N)}_{i_N}) = \prod_{l=\sigma(1)}^{\sigma(N)} a^{(l)}_{i_l,j_l}(a^{(1)}_{i_1},\ldots,a^{(N)}_{i_N});$$

Par la définition des X et Y:

– $y_{[}i_1,\ldots,i_N][j_1,\ldots,j_N] = x_{[}i_{\sigma_1},\ldots,i_{\sigma_N}]_\sigma[j_{\sigma_1},\ldots,j_{\sigma_N}]_\sigma$;

Par la définition des Z:

– $\boxed{y_{[}i_1,\ldots,i_N][j_1,\ldots,j_N] = z_{[}i_1,\ldots,i_N][j_1,\ldots,j_N]}$ ❑

Cette propriété dérive directement de la propriété analogue de l'algèbre tensorielle classique (équation 2.17) qui a été prouvée dans [81]. L' emploi de la pseudo-commutativité au cas généralisé (avec éléments fonctionnels) ne dépend que du maintien de la même évaluation pour les éléments fonctionnels lors des permutations de lignes et colonnes effectuées par les matrices P_σ et P^T_σ. Ceci peut être intuitivement perçu par le fait qu'avec n'importe quel ordre pour les automates, les paramètres des N éléments fonctionnels dans un produit

$$a^{(1)}_{i_1,j_1}(a^{(1)}_{i_1},\ldots,a^{(N)}_{i_N}) \times \cdots \times a^{(N)}_{i_N,j_N}(a^{(1)}_{i_1},\ldots,a^{(N)}_{i_N})$$

seront toujours les indices de lignes des éléments composant le même produit. Par exemple, dans un terme avec trois matrices ($A^{(1)}, A^{(2)}, A^{(3)}$ et $n_1 = 10, n_2 = 15, n_3 = 20$) l'évaluation du élément $a_{[9,8,7][5,6,13]}$ est:

$$a_{[9,8,7][5,6,13]} = a^{(1)}_{9,5}(a^{(1)}_9, a^{(2)}_8, a^{(3)}_7) \times a^{(2)}_{8,6}(a^{(1)}_9, a^{(2)}_8, a^{(3)}_7) \times a^{(3)}_{7,13}(a^{(1)}_9, a^{(2)}_8, a^{(3)}_7)$$

2.2.8 Décomposition en Produits Tensoriels Classiques

Considérons deux matrices A et $B(\mathcal{A})$, il est possible d'affirmer:

$$A \underset{g}{\otimes} B(\mathcal{A}) = \sum_{m=1}^{n_A} \ell_m(A) \otimes B(a_m)$$

Démonstration Algébrique

Appelons:

– C le produit tensoriel généralisé $\ell_m(A) \otimes B(a_m)$;

– H le produit tensoriel généralisé $A \underset{g}{\otimes} B(\mathcal{A})$;

– H' la somme traditionnelle des matrices $\sum_{m=1}^{n_A} \ell_m(A) \otimes B(a_m)$;

En calculant les éléments de la matrice $H = A \underset{g}{\otimes} B(\mathcal{A})$:

– $i, j \in [1..n_A]$, $k, l \in [1..n_B]$:

$\boxed{h_{[i,k],[j,l]} = a_{i,j} b_{k,l}(a_i)}$

En calculant les éléments de la matrice $C_m = \ell_m(A) \otimes B(a_m)$:

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$$c_{[i,k],[j,l]} = \begin{cases} a_{m,j}(b_k) b_{k,l}(a_m); & \text{si } i = m \\ 0; & \text{si } i \neq m \end{cases}$$

En calculant les éléments de la matrice $H' = \sum_{m=1}^{n_A} C$:

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$h'_{[i,k],[j,l]} = \sum_{m=1}^{n_A} c_{[i,k],[j,l]}$;

En remplaçant les éléments de chaque matrice C:

– $i, j \in [1..n_A], k, l \in [1..n_B]$:

$\boxed{h'_{[i,k],[j,l]} = a_{i,j} b_{k,l}(a_i)}$ ❑

Cette propriété peut être facilement visualisée en regardant les matrices $A \underset{g}{\otimes} B(\mathcal{A})$ et $\ell_m(A) \otimes B(a_m)$:

$$A \otimes_g B[\mathcal{A}] = \begin{pmatrix} a_{1,1}B(a_1) & a_{1,2}B(a_1) & \cdots & a_{1,n_A}B(a_1) \\ a_{2,1}B(a_2) & a_{2,2}B(a_2) & \cdots & a_{2,n_A}B(a_2) \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_A,1}B(a_{n_A}) & a_{n_A,2}B(a_{n_A}) & \cdots & a_{n_A,n_A}B(a_{n_A}) \end{pmatrix}$$

$$\ell_m(A) \otimes B(a_m) = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ a_{m,1}B(a_m) & a_{m,2}B(a_m) & \cdots & a_{m,n_a}B(a_m) \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

Ceci termine l'exposition des propriétés algébriques de l'ATC et l'ATG. Ces propriétés seront utilisées le chapitre 3 pour la définition de la matrice de transition d'un RAS. Dans le chapitre 5 ces propriétés seront utilisées pour former l'algorithme du produit vecteur-descripteur RAS. L'introduction des éléments fonctionnels dans les opérateurs tensoriels traditionnels permet d'avoir une représentation matricielle compacte de fonctionnements probabilistes complexes. De plus, cette généralisation des opérateurs tensoriels préserve la plupart des propriétés algébriques.

Chapitre 3

Réseaux d'Automates Stochastiques - RAS

Dans ce chapitre, le formalisme RAS est décrit d'abord de façon informelle [32] pour comprendre l'outil de modélisation sans préoccupation théorique. Ensuite, une description formelle est rappelée [5, 80] afin d'éviter les ambiguïtés dans l'utilisation de ce formalisme pour la spécification d'un RAS. La notation utilisée pour les RAS dans l'ensemble de cette thèse est introduite dans ce chapitre et résumé dans l'annexe A.

3.1 Description Informelle des RAS

Les Réseaux d'Automates Stochastiques [79] sont un formalisme pour la définition et la solution des systèmes complexes à grand espace d'états. Le principe de base des RAS, en tant qu'outil de modélisation, est la description d'un système comme un ensemble de sous-systèmes. Chacun de ces sous-systèmes est modélisé comme un automate stochastique. L'interaction entre les sous-systèmes est modélisée par des règles établies entre les états internes de chaque automate.

Le formalisme RAS peut être utilisé pour des modèles à échelle de temps discret ou continu. Cependant dans l'ensemble de cette thèse seuls les modèles à temps continu seront traités, sauf mention explicite contraire. Le lecteur peut trouver dans [5] une description formelle et exemples sur les RAS à échelle de temps discret.

3.1.1 Automates, Événements Synchronisants et Transitions Fonctionnelles

L'état d'un RAS est défini comme la combinaison de tous les états internes de chaque automate. Le changement de l'état global d'un RAS peut être le résultat de l'occurrence soit d'un événement local, soit d'un événement synchronisant. Un événement local représente le changement de l'état interne d'un seul automate. L'occurrence d'un événement synchronisant, par opposition, correspond au changement simultané de l'état interne de plusieurs (au moins deux) automates.

Selon cette définition, on peut décrire les automates comme un ensemble d'états et un ensemble de transitions entre eux [81, 32]. Ces transitions sont répertoriées comme transitions locales ou transitions synchronisées.

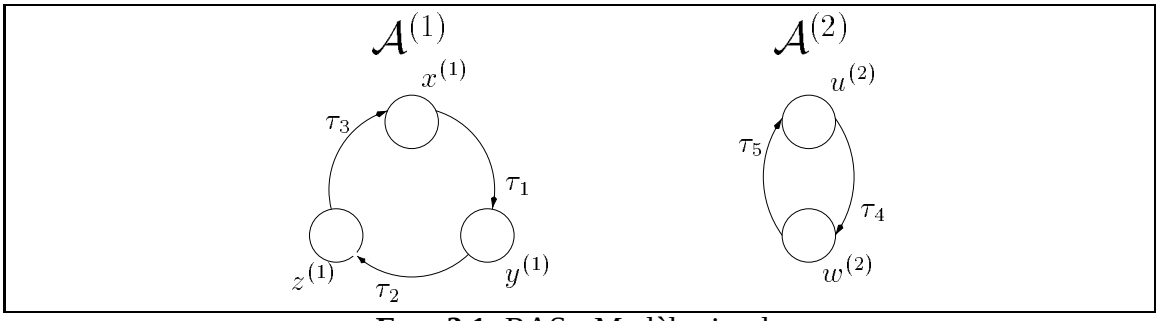


FIG. 3.1: RAS - Modèle simple

Chaque transition locale possède un taux de franchissement. Prenons, par exemple, le RAS de la figure 3.1. Le premier automate ($\mathcal{A}^{(1)}$) possède trois états internes ($x^{(1)}$, $y^{(1)}$ et $z^{(1)}$). Le second ($\mathcal{A}^{(2)}$) en possède deux ($u^{(2)}$ et $w^{(2)}$). Dans ce modèle toutes les transitions sont locales, donc tous les arcs ont comme étiquette le taux de franchissement correspondant à la transition. Le fonctionnement de chacun des automates est indépendant. L'automate équivalent à ce RAS est présenté dans la figure 3.2. Bien qu'un système puisse être défini ainsi, ce genre de modèle présente peu d'intérêt. En effet, la solution de ce type de modèle peut être calculée de façon complètement indépendante pour chacun des sous-systèmes. Autrement dit, chaque automate peut être étudié individuellement, car ils ne présentent aucune interaction entre eux.

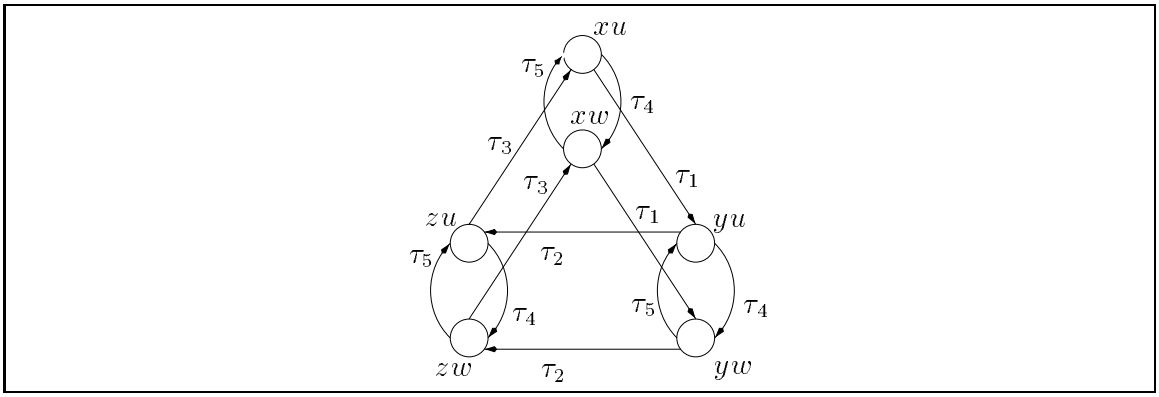


FIG. 3.2: Automate équivalent - Modèle simple

Si les transitions locales sont très simples à définir, les transitions synchronisées, au contraire, sont plus complexes. Un événement synchronisant est associé à un ensemble de transitions synchronisées. Un taux de franchissement doit être associé à chaque transition synchronisée. L'occurrence de l'événement synchronisant se manifeste simultanément dans tous les automates concernés. De ce fait, il est possible de voir ce processus de synchronisation comme un *rendez-vous*. Néanmoins, la synchronisation dans le RAS peut aussi être vue comme une relation de type maître-esclave. Dans ce cas, un des automates peut être arbitrairement choisi comme maître et les autres comme esclaves.

Les transitions synchronisées ont besoin d'une structure plus complexe qu'un simple taux de transition, cette structure est appelée *triplet de synchronisation*. Un triplet de synchronisation est composé de l'identificateur de l'événement synchronisant, un taux de franchissement et une probabilité d'occurrence.

Bien que les événements synchronisants ne doivent pas toujours être interprétés comme des synchronisations de type maître-esclave, cette interprétation s'avère très utile pour comprendre le principe de construction des triplets de synchronisation associés à une transition synchronisée. Supposons que pour chaque événement synchronisant un des automates soit arbitrairement choisi comme maître. Les étiquettes des arcs de l'automate maître possèdent des taux correspondants à l'occurrence de l'événement synchronisant. Les étiquettes des arcs des automates esclaves auront des taux égaux à un (1.0). Notons que le choix d'un automate maître est fait pour chacun des événements synchronisants. Un automate peut être choisi comme maître d'un événement et comme esclave d'un autre.

En plus du taux associé à l'événement synchronisant, les transitions causées par un même événement synchronisant partant d'un même état local, doivent posséder une probabilité de choix entre elles. La somme de ces probabilités doit être égale à un (1.0) pour les transitions du même événement partant du même état local.

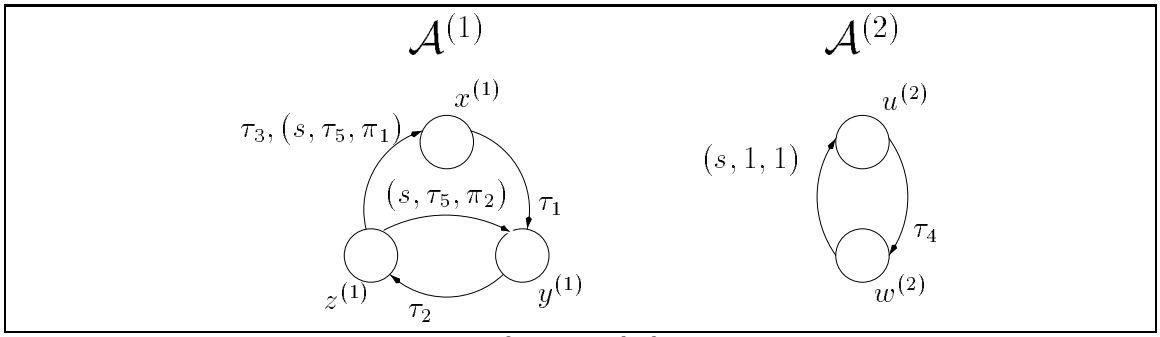


FIG. 3.3: RAS - Modèle avec événement synchronisant

Prenons par exemple, le RAS de la figure 3.3. Les transitions des états $x^{(1)}$ vers $y^{(1)}$, $y^{(1)}$ vers $z^{(1)}$, $z^{(1)}$ vers $x^{(1)}$ et $u^{(2)}$ vers $w^{(2)}$ sont des transitions locales. Les transitions des états $z^{(1)}$ vers $x^{(1)}$ et $z^{(1)}$ vers $y^{(1)}$ (dans le premier automate) et $w^{(2)}$ vers $u^{(2)}$ (dans le deuxième automate) sont des transitions synchronisées correspondant à un même événement synchronisant (s). Il faut remarquer que la transition de l'état $z^{(1)}$ vers $x^{(1)}$ peut être franchie par un événement local (avec un taux τ_3) ou par l'événement synchronisant s. Ce RAS est équivalent à l'automate de la figure 3.4.

Dans cet exemple, l'événement synchronisant ne peut se passer que si les automates se trouvent dans les états locaux $z^{(1)}$ et $w^{(2)}$. L'occurrence de l'événement pourra causer le franchissement de l'une ou l'autre des deux paires de transitions simultanées:

- soit $z^{(1)} \rightarrow x^{(1)}$ et $w^{(2)} \rightarrow u^{(2)}$ avec un taux $\tau_5\ \pi_1$;
- soit $z^{(1)} \rightarrow y^{(1)}$ et $w^{(2)} \rightarrow u^{(2)}$ avec un taux $\tau_5\ \pi_2$;

Le concept des transitions fonctionnelles, en plus du concept des événements synchronisants, est la deuxième possibilité d'interaction entre les automates. Une transition

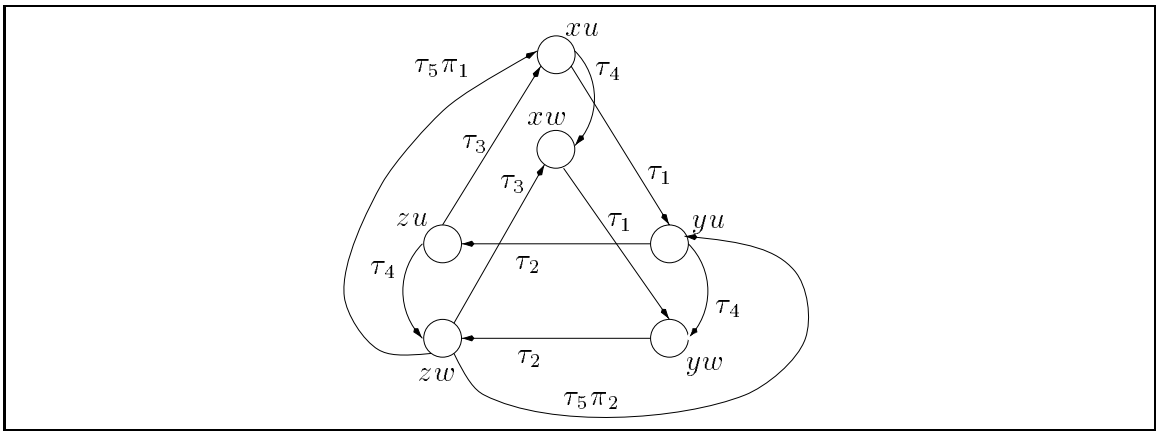


FIG. 3.4: Automate équivalent - Modèle avec événement synchronisant

fonctionnelle possède un taux de franchissement variable. Ce taux, au lieu d'être constant, est une fonction de l'état interne d'autres automates.

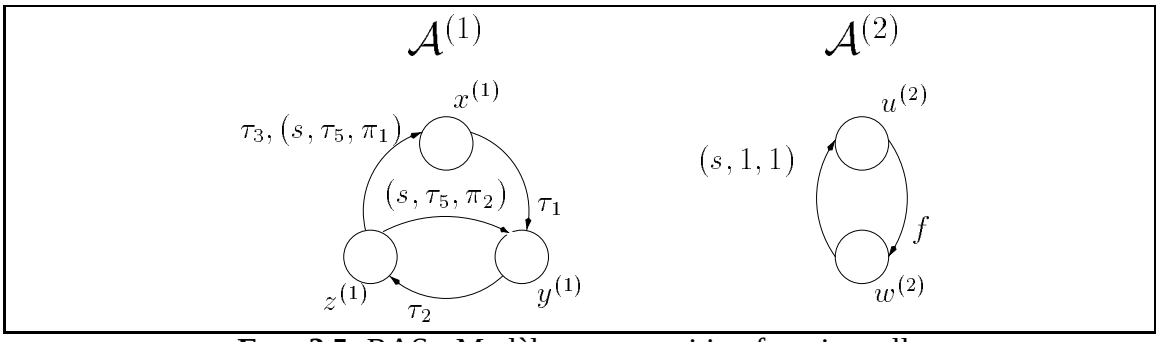


FIG. 3.5: RAS - Modèle avec transition fonctionnelle

Supposons que dans l'exemple de la figure 3.5 le taux de la transition de l'état $u^{(2)}$ vers l'état $w^{(2)}$ (deuxième automate) n'est plus constant (comme pour l'exemple de la figure 3.3), mais est une fonction f de l'état du premier automate. Cette fonction peut être exprimée comme:

$$f = \begin{cases} \lambda_1 & \text{si le premier automate est dans l'état } x^{(1)}; \\ 0 & \text{si le premier automate est dans l'état } y^{(1)}; \\ \lambda_2 & \text{si le premier automate est dans l'état } z^{(1)}; \end{cases}$$

L'automate équivalent au RAS de la figure 3.5 est présenté dans la figure 3.6.

3.1.2 Descripteur Markovien

L'un des avantages apporté par le formalisme RAS par rapport à d'autres formalismes est la facilité de passer à une description compacte du générateur infinitésimal correspondant à la chaîne de Markov associée au modèle complet [81]. Cette façon compacte de décrire le générateur est appelée *descripteur*.

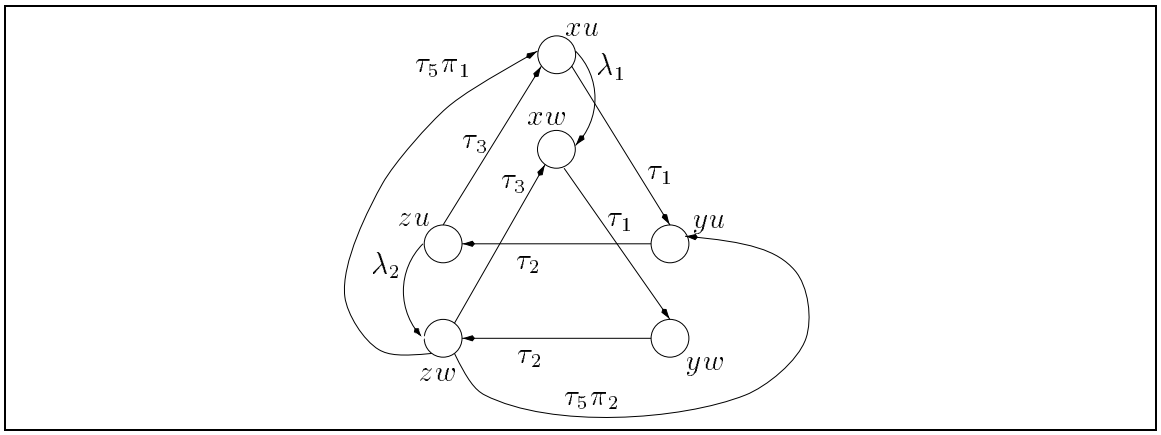


FIG. 3.6: Automate équivalent - Modèle avec transition fonctionnelle

Reprenons le RAS décrit par la figure 3.1. Une matrice de transition locale ($Q_l^{(i)}$) étant associée à chacun des automates ($\mathcal{A}^{(i)}$), le générateur de l'automate équivalent à ce modèle (Q) est, tout simplement, équivalent à la somme tensorielle des matrices de transition locales[1].

$$Q = Q_l^{(1)} \oplus Q_l^{(2)} = \begin{pmatrix} -\tau_1 & \tau_1 & 0 \\ 0 & -\tau_2 & \tau_2 \\ \tau_3 & 0 & -\tau_3 \end{pmatrix} \bigoplus \begin{pmatrix} -\tau_4 & \tau_4 \\ \tau_5 & -\tau_5 \end{pmatrix}$$

$$Q = \left(\begin{array}{cc|cc|cc} -(\tau_1+\tau_4) & \tau_4 & \tau_1 & 0 & 0 & 0 \\ \tau_5 & -(\tau_1+\tau_5) & 0 & \tau_1 & 0 & 0 \\ \hline 0 & 0 & -(\tau_2+\tau_4) & \tau_4 & \tau_2 & 0 \\ 0 & 0 & \tau_5 & -(\tau_2+\tau_5) & 0 & \tau_2 \\ \hline \tau_3 & 0 & 0 & 0 & -(\tau_3+\tau_4) & \tau_4 \\ 0 & \tau_3 & 0 & 0 & \tau_5 & -(\tau_3+\tau_5) \end{array}\right)$$

L'exemple de la figure 3.3 possédant un événement synchronisant, montre le besoin d'avoir plusieurs matrices associées à chaque automate. Comme pour l'exemple précédent, une matrice de transition locale regroupe tous les taux des événements locaux de l'automate. Ensuite, chaque événement synchronisant du modèle correspond à une paire de matrices pour chaque automate. La première matrice décrit l'occurrence de l'événement synchronisant. La deuxième matrice doit faire l'ajustement diagonal correspondant aux taux exprimés dans la matrice d'occurrence. Étant donné que les matrices d'occurrence ont pour objectif d'exprimer des taux positifs, et que les matrices d'ajustement diagonal ont pour objectif d'exprimer des taux négatifs[2], on appellera ces matrices respectivement *matrices positives* et *matrices négatives*.

[1] La définition de la somme tensorielle est dans la page 20.

[2] En vérité, seulement les matrices correspondant à l'automate maître d'un événement peuvent posséder des taux négatifs.

Les automates non concernés par un événement synchronisant ont des identités pour les matrices positives et négatives. Cela vient du fait que ces automates ne changent pas d'état lors de l'occurrence de l'événement synchronisant. Les matrices positives correspondant à l'automate maître[3] contiennent les taux d'occurrence de l'événement synchronisant. L'existence de probabilités associées à une transition aura pour effet la multiplication du taux correspondant par la probabilité. Les matrices des automates esclaves contiennent un taux égal à un (1.0).

De façon analogue, les matrices d'ajustement contiennent des taux négatifs seulement dans la matrice de l'automate maître. Les matrices d'ajustement des automates esclaves, à l'image des matrices d'occurrence, possèdent des taux égaux à un (1.0).

Le descripteur sera donc exprimé en deux parties: une partie dite *locale* correspondant aux transitions locales et une partie dite *synchronisante* regroupant tous les événements synchronisants. La partie locale est définie, comme dans l'exemple précédant, par une somme tensorielle des matrices locales de chaque automate. Pour la partie synchronisante, chaque événement correspondra à deux produits tensoriels, un pour les matrices d'occurrence et l'autre pour les matrices d'ajustement. Le descripteur sera la somme de la partie locale et de la partie synchronisante.

L'exemple de la figure 3.3 peut avoir comme descripteur deux représentations équivalentes selon l'automate choisi comme maître de l'événement synchronisant. Il faut noter que dans les deux options le générateur correspondant est absolument identique.

Partie locale:

$$Q_l = Q_l^{(1)} \oplus Q_l^{(2)} = \begin{pmatrix} -\tau_1 & \tau_1 & 0 \\ 0 & -\tau_2 & \tau_2 \\ \tau_3 & 0 & -\tau_3 \end{pmatrix} \bigoplus \begin{pmatrix} -\tau_4 & \tau_4 \\ 0 & 0 \end{pmatrix}$$

$$Q_l = \left(\begin{array}{cc|cc|cc} -(\tau_1+\tau_4) & \tau_4 & \tau_1 & 0 & 0 & 0 \\ 0 & -\tau_1 & 0 & \tau_1 & 0 & 0 \\ \hline 0 & 0 & -(\tau_2+\tau_4) & \tau_4 & \tau_2 & 0 \\ 0 & 0 & 0 & -\tau_2 & 0 & \tau_2 \\ \hline \tau_3 & 0 & 0 & 0 & -(\tau_3+\tau_4) & \tau_4 \\ 0 & \tau_3 & 0 & 0 & 0 & -\tau_3 \end{array}\right)$$

Partie synchronisante positive (le premier automate est choisi maître de l'événement s):

$$Q_{s+} = Q_{s+}^{(1)} \otimes Q_{s+}^{(2)} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \tau_5\pi_1 & \tau_5\pi_2 & 0 \end{pmatrix} \bigotimes \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$$

[3]Pour chaque événement synchronisant un des automates est choisi comme automate maître. Ce choix peut correspondre à une interprétation du système modelé, ou il peut s'agir d'un choix arbitraire, que n'apporte aucune restriction à la généralité du formalisme.

$$Q_{s^+} = \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \tau_5\pi_1 & 0 & \tau_5\pi_2 & 0 & 0 & 0 \end{array}\right)$$

Partie synchronisante négative (le premier automate est choisi maître de l'événement s):

$$Q_{s^-} = Q_{s^-}^{(1)} \otimes Q_{s^-}^{(2)} = \left(\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\tau_5 \end{array}\right) \bigotimes \left(\begin{array}{cc} 0 & 0 \\ 0 & 1 \end{array}\right)$$

$$Q_{s^-} = \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\tau_5 \end{array}\right)$$

Partie synchronisante positive (le deuxième automate est choisi maître de l'événement s):

$$Q_{s^+} = Q_{s^+}^{(1)} \otimes Q_{s^+}^{(2)} = \left(\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \pi_1 & \pi_2 & 0 \end{array}\right) \bigotimes \left(\begin{array}{cc} 0 & 0 \\ \tau_5 & 0 \end{array}\right)$$

$$Q_{s^+} = \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \tau_5\pi_1 & 0 & \tau_5\pi_2 & 0 & 0 & 0 \end{array}\right)$$

Partie synchronisante négative (le deuxième automate est choisi maître de l'événement s):

$$Q_{s^-} = Q_{s^-}^{(1)} \otimes Q_{s^-}^{(2)} = \left(\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{array}\right) \bigotimes \left(\begin{array}{cc} 0 & 0 \\ 0 & -\tau_5 \end{array}\right)$$

$$Q_{s^-} = \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\tau_5 \end{array}\right)$$

Générateur de l'automate global:

$$Q = Q_l + Q_{s^+} + Q_{s^-}$$

$$Q = \left(\begin{array}{cc|cc|cc} -(\tau_1+\tau_4) & \tau_4 & \tau_1 & 0 & 0 & 0 \\ 0 & -\tau_1 & 0 & \tau_1 & 0 & 0 \\ \hline 0 & 0 & -(\tau_2+\tau_4) & \tau_4 & \tau_2 & 0 \\ 0 & 0 & 0 & -\tau_2 & 0 & \tau_2 \\ \hline \tau_3 & 0 & 0 & 0 & -(\tau_3+\tau_4) & \tau_4 \\ \tau_5\pi_1 & \tau_3 & \tau_5\pi_2 & 0 & 0 & -(\tau_3+\tau_5) \end{array}\right)$$

Il est intéressant de remarquer que la somme tensorielle correspondant à la partie locale du descripteur est une somme de produits tensoriels particuliers (voir la définition des facteurs normaux dans la page 19). Une interprétation possible des événements locaux est de les voir comme un cas très particulier d'événement synchronisant, où seulement un automate est concerné. Ceci revient à dire que les autres automates seront représentés par des matrices identités.

Les transitions fonctionnelles, comme dans l'exemple de la figure 3.5, n'apportent pas de modifications structurelles au descripteur, seule l'utilisation des produits tensoriels généralisés est nécessaire. De ce fait, l'expression du descripteur de la figure 3.5 est la suivante (le premier automate est choisi comme maître de l'événement s):

Partie locale:

$$Q_l = Q_l^{(1)} \underset{g}{\oplus} Q_l^{(2)} = \begin{pmatrix} -\tau_1 & \tau_1 & 0 \\ 0 & -\tau_2 & \tau_2 \\ 0 & 0 & 0 \end{pmatrix} \bigoplus_g \begin{pmatrix} -f & f \\ 0 & 0 \end{pmatrix}$$

$$Q_l = \left(\begin{array}{cc|cc|cc} -(\tau_1+\lambda_1) & \lambda_1 & \tau_1 & 0 & 0 & 0 \\ 0 & -\tau_1 & 0 & \tau_1 & 0 & 0 \\ \hline 0 & 0 & -\tau_2 & 0 & \tau_2 & 0 \\ 0 & 0 & 0 & -\tau_2 & 0 & \tau_2 \\ \hline \tau_3 & 0 & 0 & 0 & -(\lambda_2+\tau_3) & \lambda_2 \\ 0 & \tau_3 & 0 & 0 & 0 & -\tau_3 \end{array}\right)$$

Partie synchronisante positive:

$$Q_{s^+} = Q_{s^+}^{(1)} \underset{g}{\otimes} Q_{s^+}^{(2)} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \tau_5\pi_1 & \tau_5\pi_2 & 0 \end{pmatrix} \bigotimes_g \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$$

$$Q_{s^+} = \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \tau_5\pi_1 & 0 & \tau_5\pi_2 & 0 & 0 & 0 \end{array}\right)$$

Partie synchronisante négative:

$$Q_{s^-} = Q_{s^-}^{(1)} \underset{g}{\otimes} Q_{s^-}^{(2)} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\tau_5 \end{pmatrix} \underset{g}{\bigotimes} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$$Q_{s^-} = \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\tau_5 \end{array}\right)$$

Générateur de l'automate global:

$$Q = Q_l + Q_{s^+} + Q_{s^-} = \left(\begin{array}{cc|cc|cc} -(\tau_1+\lambda_1) & \lambda_1 & \tau_1 & 0 & 0 & 0 \\ 0 & -\tau_1 & 0 & \tau_1 & 0 & 0 \\ \hline 0 & 0 & -\tau_2 & 0 & \tau_2 & 0 \\ 0 & 0 & 0 & -\tau_2 & 0 & \tau_2 \\ \hline \tau_3 & 0 & 0 & 0 & -(\tau_3+\lambda_2) & \lambda_2 \\ \tau_5\pi_1 & \tau_3 & \tau_5\pi_2 & 0 & 0 & -(\tau_3+\tau_5) \end{array}\right)$$

3.2 Description Formelle des RAS

Dans cette section, on présente un ensemble de définitions de base avec les notations utilisées tout au long de cette thèse. Ensuite, des restrictions à ces définitions sont ajoutées précisant le concept de RAS *bien défini*. Le générateur Markovien obtenu à partir du RAS est présenté à la fin de cette section. Les notations sont précédées par le symbole ✍. L'annexe A contient l'ensemble des notations pour une consultation rapide.

3.2.1 Définitions de Base

On va considérer la formalisation d'un RAS comprenant N automates $\mathcal{A}^{(i)}$ ($i \in [1..N]$) et E événements synchronisants.

✍ **Soit**

ε	l'ensemble des identificateurs d'événements synchronisants;
$\mathcal{A}^{(i)}$	le i-ème automate d'un RAS;
$S^{(i)}$	l'ensemble des états (locaux) de l'automate $\mathcal{A}^{(i)}$;
n_i	le nombre des états de $S^{(i)}$ ($n_i = cardinal(S^{(i)})$);
$x^{(i)}$	un état (local) de l'automate $\mathcal{A}^{(i)}$ ($x^{(i)} \in S^{(i)}$);
$\tilde{x}$	l'état global (combinaison d'états locaux) d'un RAS à N automates, $\tilde{x} = (x^{(1)}, \ldots, x^{(N)})$ où $x^{(i)}$ est l'état local de l'automate $\mathcal{A}^{(i)}$ ($\tilde{x} \in \prod_{i=1}^{N} S^{(i)}$);

$\tilde{x}(x^{(i)} \xrightarrow{l} y^{(i)})$ l'état global obtenu en remplaçant l'état local $x^{(i)}$ par l'état local $y^{(i)}$ dans l'automate $\mathcal{A}^{(i)}$;

$x^{(\omega)}$ la composition des états locaux $x^{(i)}$ où $i \in \omega$, avec $\omega \subset [1..N]$;

Notons que la définition d'un état local d'automate ($x^{(i)}$) et la définition d'un état global ($\tilde{x}$) peuvent être vues comme des cas particuliers des $x^{(\omega)}$. Un état local est le cas particulier où $\omega = \{i\}$, tandis que l'état global est le cas où $\omega = [1..N]$.

Définition 1 *Un élément fonctionnel $f(\mathcal{A}^{(\omega)})$ est une fonction de $\prod_{i \in \omega} S^{(i)}$ dans $\mathbb{R}^+$, où ω est un sous-ensemble de $[1..N]$.*

Les automates $\mathcal{A}^{(i)}$ avec $i \in \omega$ sont les paramètres de l'élément $f(\mathcal{A}^{(\omega)})$. Les éléments fonctionnels servent à définir les probabilités et les taux fonctionnels comme décrit dans ce qui suit.

✍ **Soit**

$f(x^{(\omega)})$ l'élément fonctionnel $f(\mathcal{A}^{(\omega)})$ évalué pour les états locaux $x^{(\omega)}$;

Notons que tous les éléments constants peuvent être vus comme des éléments fonctionnels qui ont toujours la même évaluation pour n'importe quel jeu de paramètres $x^{(\omega)}$. De même, les éléments fonctionnels avec un jeu de paramètres restreint à un sous-ensemble ω de $[1..N]$ peuvent être vus comme des fonctions d'un état global $\tilde{x}$ avec des évaluations identiques pour tout $\tilde{x}(x^{(i)} \rightarrow y^{(i)})$ avec $i \notin \omega$. De cette façon, tous les éléments d'un RAS peuvent être considérés comme fonctions de $\prod_{i=1}^{N} S^{(i)}$ dans $\mathbb{R}^+$.

Définition 2 *Chaque événement synchronisant d'un RAS est identifié par:*

2.1. un identificateur e, avec $e \in \varepsilon$;

2.2. un indice de son automate maître $\iota^{(e)}$, avec $\iota^{(e)} \in [1..N]$.

Définition 3 *Un triplet de synchronisation (e, τ_e, π_e) est composé de trois informations:*

3.1. $e \in \varepsilon$, l'identificateur d'un événement synchronisant;

3.2. τ_e, un élément fonctionnel défini de $\prod_{i=1}^{N} S^{(i)}$ dans $\mathbb{R}^+$, représentant un taux d'occurrence de l'événement synchronisant e;

3.3. π_e un élément fonctionnel défini de $\prod_{i=1}^{N} S^{(i)}$ dans $[0, 1]$, représentant la probabilité de routage d'une transition synchronisée lors de l'occurrence de l'événement synchronisant e;

Définition 4 *L'ensemble L contient toutes les étiquettes de transition. Chaque étiquette de transition de cet ensemble est composée de:*

4.1. τ_l un élément fonctionnel défini de $\prod_{i=1}^{N} S^{(i)}$ dans $\mathbb{R}^+$, représentant le taux d'occurrence local de la transition;

4.2. une liste de triplets de synchronisation (e, τ_e, p_e), qui contient au plus une fois chaque événement $e \in \varepsilon$.

L'interprétation de ces étiquettes sur la transition d'un automate $\mathcal{A}^{(i)}$ sont les suivantes:

- le taux de transition local peut être nul si la transition ne peut pas se dérouler indépendamment des autres automates;
- la liste de triplets de synchronisation est une liste avec un nombre de triplets égal au nombre d'événements synchronisants qui peuvent causer cette transition (ce nombre peut être zéro - liste vide - si la transition en question ne peut pas être franchie de façon synchronisée avec d'autres automates).

Définition 5 *Chaque automate $\mathcal{A}^{(i)}$, est défini par:*

5.1. $S^{(i)}$ est l'ensemble des états de l'automate $\mathcal{A}^{(i)}$;

5.2. $Q^{(i)}$ est la fonction de transition de l'automate $\mathcal{A}^{(i)}$ définie de $S^{(i)} \times S^{(i)}$ dans L;

Un automate $\mathcal{A}^{(i)}$ a comme paramètres l'union des paramètres de tous les éléments fonctionnels contenus dans ses étiquettes de transition.

✍ **Soit**

$\mathcal{A}^{(i)}(\mathcal{A}^{(\omega)})$ l'automate $\mathcal{A}^{(i)}$ qui possède comme paramètres les automates $\mathcal{A}^{(j)}$ avec $j \in \omega$;

$\mathcal{A}^{(i)}(x^{(\omega)})$ l'automate $\mathcal{A}^{(i)}(\mathcal{A}^{(\omega)})$ avec tous ses éléments fonctionnels évalués pour la composition des états locaux $x^{(\omega)}$);

$\eta^{(e)}$ l'ensemble des indices i ($i \in [1..N]$), tel que l'automate $\mathcal{A}^{(i)}$ possède dans au moins une des étiquettes de $Q^{(i)}$ un triplet de synchronisation avec l'identificateur de l'événement e;

$Q^{(i)}(x^{(i)}, y^{(i)})$ l'étiquette de transition de l'état local $x^{(i)}$ vers l'état local $y^{(i)}$;

$\tau_l[x^{(i)}, y^{(i)}]$ le taux local de l'étiquette $Q^{(i)}(x^{(i)}, y^{(i)})$;

$succ_l(x^{(i)})$ l'ensemble des états $y^{(i)}$ tels que le taux $\tau_l[x^{(i)}, y^{(i)}]$ est non identiquement nul;

$\tilde{x}(x^{(i)} \xrightarrow{l} y^{(i)})$ l'état global obtenu en remplaçant l'état local $x^{(i)}$ par l'état local $y^{(i)}$ dans l'automate $\mathcal{A}^{(i)}$ ($y^{(i)} \in succ_l(x^{(i)})$);

$\tau_e[x^{(i)}, y^{(i)}]$ le taux synchronisant du triplet de synchronisation (e, τ_e, π_e) associée à la transition $Q^{(i)}(x^{(i)}, y^{(i)})$;

$\pi_e[x^{(i)}, y^{(i)}]$ la probabilité de routage du triplet de synchronisation (e, τ_e, π_e) associée à la transition $Q^{(i)}(x^{(i)}, y^{(i)})$;

$succ_e(x^{(i)})$ l'ensemble des états $y^{(i)}$ tels que l'étiquette $Q^{(i)}(x^{(i)}, y^{(i)})$ possède un triplet de synchronisation avec l'identificateur e et $\tau_e[x^{(i)}, y^{(i)}] \neq 0$, $\pi_e[x^{(i)}, y^{(i)}] \neq 0$ (éléments fonctionnels non identiquement nuls);

$\tilde{x}(x^{(i)} \overset{i\in\eta^{(e)}}{\rightarrow} y^{(i)})$ l'état global obtenu en remplaçant tous les états locaux $x^{(i)}$ par $y^{(i)}$ dans tous les automates $\mathcal{A}^{(i)}$ ($i \in \eta^{(e)}$ et $y^{(i)} \in succ_e(x^{(i)})$);

On dit qu'un événement synchronisant e est *réalisable* dans l'état global $\tilde{x}$ si et seulement si $\forall i \in \eta^{(e)}$ les ensembles $succ_e(x^{(i)})$ ne sont pas vides.

Définition 6 *Un RAS composé de N automates et E événements synchronisants est défini par:*

6.1. chacun des automates $\mathcal{A}^{(i)}$ ($i \in [1..N]$);

6.2. chacun des événements synchronisants e ($e \in \varepsilon$);

6.3. la fonction d'atteignabilité F, un élément fonctionnel défini de $\prod_{i=1}^{N} S^{(i)}$ dans $[0..1]$.

La fonction F associe aux états globaux de $\prod_{i=1}^{N} S^{(i)}$ la valeur 1 si ils sont atteignables et 0 dans le cas contraire.

✍ **Soit**

S	l'espace d'état du RAS défini comme $\prod_{i=1}^{N} S^{(i)}$;
R	le sous-ensemble de S comprenant tous les états $\tilde{x}$ tels que $F(\tilde{x}) = 1$;

Un RAS définit un seul automate global équivalent. Cette approche à été développée en [5, 80] et ne serait pas développée dans cette thèse. Après la section suivante où l'on impose des restrictions à la définition d'un RAS, on donnera directement la construction du descripteur à partir d'un RAS défini selon la définition 6.

3.2.2 RAS *bien définis*

La définition d'un RAS doit être non ambiguë, c'est à dire qu'un seul générateur Markovien peut être obtenu à partir d'un RAS. Pour cela un certain nombre de restrictions doivent être posées. Les RAS observant ces restrictions sont dit RAS *bien définis*.

Restriction 1 *Un automate $\mathcal{A}^{(i)}$ est* bien défini *si et seulement si pour tout $\tilde{x} \in S$, pour tout $x^{(i)} \in S^{(i)}$ et pour tout $e \in \varepsilon$ tel que $succ_e(x^{(i)})$ n'est pas vide:*

1.1. $\forall y^{(i)}, z^{(i)} \in succ_e(x^{(i)})$
$\tau_e[x^{(i)}, y^{(i)}](\tilde{x}) = \tau_e[x^{(i)}, z^{(i)}](\tilde{x})$;

1.2. $\left(\sum_{y^{(i)} \in succ_e(x^{(i)})} \pi_e[x^{(i)}, y^{(i)}](\tilde{x}) \right) = 1$ *ou un élément fonctionnel évaluable à* 0 *ou* 1;

Les triplets de synchronisation concernant un même événement synchronisant e et concernant des transitions de $\mathcal{A}^{(i)}$ issues du même état doivent posséder le même taux de transition (restriction 1.1) et la somme des probabilités de routage sur toutes les transitions issues de ce même état doit être égale à un (1.0) ou à un élément fonctionnel évaluable sur $[0..1]$ (restriction 1.2). Ces restrictions ont pour objectif de garantir l'unicité de la

définition des taux des événements synchronisants par rapport à l'ensemble des transitions synchronisées dans chacun des automates.

Restriction 2 *Un événement synchronisant* $e \in \varepsilon$ *est* bien défini *si et seulement si* $\eta^{(e)}$ *n'est pas vide, son automate maître est tel que* $\iota^{(e)} \in \eta^{(e)}$ *et pour tout état global* $\tilde{x}$ *où* e *est réalisable:*

2.1. $\forall i \in \eta^{(e)}$ *et* $i \neq \iota^{(e)}$
$\tau_e[x^{(i)}, y^{(i)}](\tilde{x}) = 1$ *ou un élément fonctionnel évaluable à* 0 *ou* 1*;*

La restriction 2 affirme que les taux d'occurrence d'un événement synchronisant e ne doivent apparaître que dans les triplets de synchronisation de son automate maître ($\mathcal{A}^{(\iota^{(e)})}$). Les autres automates concernés par e (automates esclaves) ont des triplets avec des taux constants égaux à un (1.0) ou des taux fonctionnels évaluables à 0 ou 1. Rappellons que cette attribution de l'automate *maître* peut être arbitraire. Le choix de l'automate maître est indifférent à la sémantique de l'événement synchronisant.

Restriction 3 *Une fonction d'atteignabilité* F *est bien définie si et seulement si l'ensemble des états atteignables (*R*) est un graphe de transition fortement connexe.*

La troisième restriction assure irreductibilité de la chaîne de Markov correspondante au RAS et permet d'employer les théorèmes standards.

Restriction 4 *Un RAS est* bien défini *si et seulement si*

4.1. tous ses automates sont bien définis;

4.2. tous ses événements synchronisants sont bien définis;

4.3. sa fonction d'atteignabilité est bien définie.

Toutes les restrictions précedentes sont indispensables pour déterminer un RAS bien défini.

3.2.3 Descripteur Markovien

Le descripteur Markovien est une formule algébrique qui permet une écriture compacte du générateur infinitésimal de la chaîne de Markov correspondante à un RAS par le biais d'une formulation mathématique [79, 80, 5]. Cette formulation mathématique décrit, à partir des matrices de transition de chaque automate, le générateur infinitésimal de la chaîne de Markov associée au RAS.

À tout automate $\mathcal{A}^{(i)}$ sont associées:

- une matrice regroupant tous les taux de transitions locaux, appelée $Q_l^{(i)}$, et
- $2E$ matrices regroupant tous les triplets de synchronisation pour les événements e de l'ensemble ε, appelées $Q_{e^+}^{(i)}$ et $Q_{e^-}^{(i)}$.

✍ **Soit**

$Q_j^{(i)}(x^{(i)}, y^{(i)})$ l'élément de la matrice $Q_j^{(i)}$ dans la ligne $x^{(i)}$ et la colonne $y^{(i)}$, avec $i \in [1..N]$ et $j \in \{l, e^+, e^-\}$;

I_{n_i} la matrice identité de taille n_i, avec $i \in [1..N]$;

Définition 7 *Les éléments de la matrice de transition locale de l'automate $\mathcal{A}^{(i)}$ sont définis par:*

7.1. $\forall x^{(i)}, y^{(i)} \in S^{(i)} \mid x^{(i)} \neq y^{(i)}$

$$Q_l^{(i)}(x^{(i)}, y^{(i)}) = \tau_l[x^{(i)}, y^{(i)}];$$

7.2. $\forall x^{(i)} \in S^{(i)}$

$$Q_l^{(i)}(x^{(i)}, x^{(i)}) = - \sum_{y^{(i)} \in succ_l(x^{(i)})} \tau_l[x^{(i)}, y^{(i)}];$$

La définition 7.1 correspond aux éléments non diagonaux de la matrice de transition locale (taux des événements locaux), tandis que la définition 7.2 correspond aux éléments diagonaux (ajustement diagonal des taux des événements locaux).

Définition 8 *Les matrices des transition synchronisantes représentant l'occurrence de l'événement $e \in \varepsilon$ sont définies par:*

8.1. $\forall i \notin \eta^{(e)}$

$$Q_{e^+}^{(i)} = I_{n_i};$$

8.2. $\forall i \in \eta^{(e)}$, $\forall x^{(i)}, y^{(i)} \in S^{(i)}$ *tel que* $y^{(i)} \in succ_e(x^{(i)})$

$$Q_{e^+}^{(i)}(x^{(i)}, y^{(i)}) = \tau_e[x^{(i)}, y^{(i)}]\pi_e[x^{(i)}, y^{(i)}];$$

8.3. $\forall i \in \eta^{(e)}$, $\forall x^{(i)}, y^{(i)} \in S^{(i)}$ *tel que* $y^{(i)} \notin succ_e(x^{(i)})$

$$Q_{e^+}^{(i)}(x^{(i)}, y^{(i)}) = 0;$$

La définition 8.1 correspond aux automates non concernés par l'événement synchronisant e. La définition 8.2 défini les éléments non nuls (diagonaux ou non) des matrices pour les automates concernés par l'événement e. La définition 8.3 défini les éléments nuls des matrices pour les automates concernés par l'événement e. Il faut remarquer que la même formulation est utilisée pour tous les automates concernés ($\mathcal{A}^{(i)}$ où $i \in \eta^{(e)}$), qu'il soient maître ou esclaves.

Définition 9 *Les matrices de transition synchronisées représentant l'ajustement nécessaire à l'occurrence de l'événement $e \in \varepsilon$ sont définies par:*

9.1. $\forall i \notin \eta^{(e)}$

$$Q_{e^-}^{(i)} = I_{n_i};$$

9.2. $\forall x^{(\iota^{(e)})} \in S^{(\iota^{(e)})}$

$$Q_{e^-}^{(\iota^{(e)})}(x^{(\iota^{(e)})}, x^{(\iota^{(e)})}) = - \sum_{y^{(\iota^{(e)})} \in succ_e(x^{(\iota^{(e)})})} \tau_e[x^{(\iota^{(e)})}, y^{(\iota^{(e)})}]\pi_e[x^{(\iota^{(e)})}, y^{(\iota^{(e)})}];$$

9.3. $\forall i \in \eta^{(e)}, i \neq \iota^{(e)}$ *et* $\forall x^{(i)} \in S^{(i)}$
$$Q_{e^-}^{(i)}(x^{(i)}, x^{(i)}) = \sum_{y^{(i)} \in succ_e(x^{(i)})} \pi_e[x^{(i)}, y^{(i)}];$$

9.4. $\forall i \in \eta^{(e)}, \forall x^{(i)}, y^{(i)} \in S^{(i)}$ *et* $x^{(i)} \neq y^{(i)}$
$$Q_{e^-}^{(i)}(x^{(i)}, y^{(i)}) = 0;$$

La définition 9.1 correspond aux automates non concernés par l'événement synchronisant e. La définition 9.2 correspond aux éléments non nuls de la matrice de l'automate maître de l'événement e, alors que la définition 9.3 correspond aux automates esclaves. La définition 9.4 correspond aux éléments nuls des matrices des automates concernés par l'événement synchronisant e (ces matrices sont diagonales).

Définition 10 *Le générateur Markovien Q correspondant à la chaîne de Markov associée à un RAS bien défini est défini par la formule tensorielle appelée Descripteur Markovien [79, 80, 5]:*

$$Q = \bigoplus_{i=1}^{N}{}_g \, Q_l^{(i)} + \sum_{e \in \varepsilon} \left(\bigotimes_{i=1}^{N}{}_g \, Q_{e^+}^{(i)} + \bigotimes_{i=1}^{N}{}_g \, Q_{e^-}^{(i)} \right) \tag{3.1}$$

Étant donné que toute somme tensorielle est équivalente à une somme de produits tensoriels particuliers (voir chapitre 2), le descripteur peut être présenté par:

$$Q = \sum_{j=1}^{(N+2E)} \bigotimes_{i=1}^{N}{}_g \, Q_j^{(i)} \tag{3.2}$$

$$\text{où } Q_j^{(i)} = \begin{cases} I_{n_i} & \text{pour } j \leq N \text{ et } j \neq i; \\ Q_l^{(i)} & \text{pour } j \leq N \text{ et } j = i; \\ Q_{(j-N)^+}^{(i)} & \text{pour } N < j \leq N + E; \\ Q_{(j-(N+E))^-}^{(i)} & \text{pour } j > N + E; \end{cases}$$ [4]

Le tableau 3.1 représente les matrices de transition nécessaires à l'écriture de l'équation 3.2. Le logiciel PEPS (chapitre 7) utilise ce tableau comme modèle pour le stockage informatique du descripteur.

[4] Cette formulation utilise comme ensemble d'identificateurs des événements synchronisants ε l'intervalle $[1..E]$

$\sum$	N		$Q_l^{(1)} \underset{g}{\otimes} I_{n_2} \underset{g}{\otimes} \cdots \underset{g}{\otimes} I_{n_{N-1}} \underset{g}{\otimes} I_{n_N}$ $I_{n_1} \underset{g}{\otimes} Q_l^{(2)} \underset{g}{\otimes} \cdots \underset{g}{\otimes} I_{n_{N-1}} \underset{g}{\otimes} I_{n_N}$ $\vdots$ $I_{n_1} \underset{g}{\otimes} I_{n_2} \underset{g}{\otimes} \cdots \underset{g}{\otimes} Q_l^{(N-1)} \underset{g}{\otimes} I_{n_N}$ $I_{n_1} \underset{g}{\otimes} I_{n_2} \underset{g}{\otimes} \cdots \underset{g}{\otimes} I_{n_{N-1}} \underset{g}{\otimes} Q_l^{(N)}$
	$2E$	e^+	$Q_{1^+}^{(1)} \underset{g}{\otimes} Q_{1^+}^{(2)} \underset{g}{\otimes} \cdots \underset{g}{\otimes} Q_{1^+}^{(N-1)} \underset{g}{\otimes} Q_{1^+}^{(N)}$ $\vdots$ $Q_{E^+}^{(1)} \underset{g}{\otimes} Q_{E^+}^{(2)} \underset{g}{\otimes} \cdots \underset{g}{\otimes} Q_{E^+}^{(N-1)} \underset{g}{\otimes} Q_{E^+}^{(N)}$
		e^-	$Q_{1^-}^{(1)} \underset{g}{\otimes} Q_{1^-}^{(2)} \underset{g}{\otimes} \cdots \underset{g}{\otimes} Q_{1^-}^{(N-1)} \underset{g}{\otimes} Q_{1^-}^{(N)}$ $\vdots$ $Q_{E^-}^{(1)} \underset{g}{\otimes} Q_{E^-}^{(2)} \underset{g}{\otimes} \cdots \underset{g}{\otimes} Q_{E^-}^{(N-1)} \underset{g}{\otimes} Q_{E^-}^{(N)}$

TAB. 3.1: Descripteur Markovien

Chapitre 4

Exemples de Modélisation avec les RAS

Dans ce chapitre, quelques exemples sont présentés pour illustrer l'utilisation pratique du formalisme RAS. Des variations de ces exemples sont reprises dans les chapitres suivants de cette thèse.

Ce chapitre présente trois exemples de modélisation avec les RAS. Chacun des exemples a pour but d'illustrer l'utilisation pratique d'un type d'interaction entre les automates. Les exemples développés sont:

- partage de ressources;
- réseau de files d'attente fermé;
- réseau de files d'attente ouvert;
  - mécanisme de perte;
  - mécanisme de blocage;
  - mécanisme de priorité;
  - mécanisme de routage variable;

Le premier exemple illustre l'utilisation des éléments fonctionnels dans un modèle sans événements synchronisants. Le partage des ressources est un exemple où l'approche modulaire de la modélisation avec les RAS est important.

L'exemple des réseaux de files d'attente fermés est largement couvert par la littérature et des solutions à forme produit sont connues. Cependant, le modèle RAS proposé est un cas assez pédagogique où les événements synchronisants sont utilisés comme unique forme d'interaction entre les automates. Ce modèle n'utilise ni élément fonctionnel, ni transition locale.

Le troisième exemple montre l'utilisation combinée des éléments fonctionnels et événements synchronisants. De plus, dans cet exemple plusieurs automates peuvent être utilisés pour décrire un même sous-système. Cet exemple montre aussi une bonne flexibilité des RAS pour adapter le modèle à des cas plus complexes (par rapport à l'approche traditionnelle des réseaux de files d'attente).

4.1 Partage de ressources

Dans cet exemple N clients distincts se partagent l'utilisation des P ressources communes identiques entre elles. Chacun des clients est modélisé par deux états: *repos* et *activité*. Dans l'état *repos*, le client n'utilise pas de ressource alors que dans l'état *activité* il utilise l'une des ressources communes.

Notons que si $P = 1$, ce modèle représente un cas classique d'exclusion mutuelle entre N clients. De même, si $P \geq N$ ce modèle représente un cas où le fonctionnement des clients est entièrement indépendant.

Cet exemple peut être représenté par le modèle en réseaux de Petri stochastique généralisés [75, 38, 2] (GSPN)[1] de la figure 4.1.

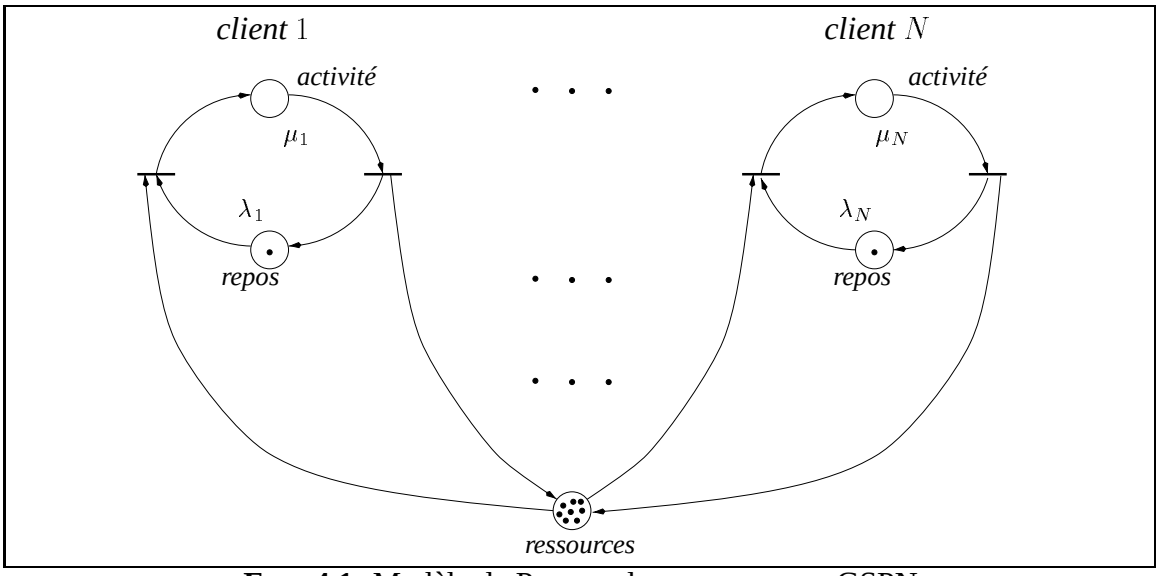


FIG. 4.1: Modèle de Partage de ressources en GSPN

Le nombre de clients est représenté par le nombre de sous-systèmes formés par deux places (*repos* et *activité*). Le nombre de ressources est représenté par le marquage initial de la place *ressources*. La possession d'une ressource par un client i est représentée par la transition qui consomme un jeton (le client) de la place *repos* du sous-système i et un jeton de la place *ressources* (le ressource). L'état *activité* du sous-système i représente l'utilisation d'une ressource par le client i. Les taux exponentiels λ_i et μ_i représentent respectivement les taux des requêtes et des libérations de ressources. Ces taux sont associés aux places *repos* et *activité* de chaque sous-système.

Ce modèle peut être traduit par le RAS décrit dans la figure 4.2. Dans ce modèle chaque client est représenté par un automate à deux états: *repos* et *activité*. Un automate $\mathcal{A}^{(i)}$ dans l'état *repos* peut passer à l'état *activité* selon un taux λ_i multiplié par une fonc-

[1] Le formalisme des réseaux de Petri stochastiques généralises (GSPN) peut être consulté dans les références [3]

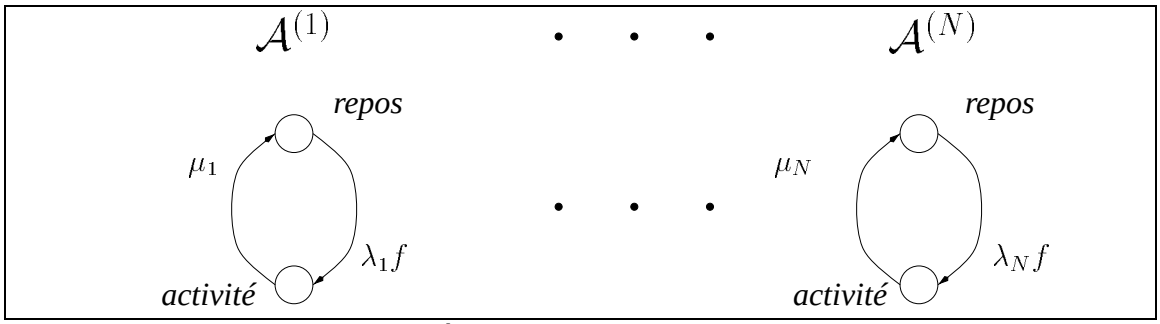


FIG. 4.2: Modèle de Partage de ressources en RAS

tion f définie par[2]:

$$f(\tilde{x}) = \delta\left(\sum_{i=1}^{N} \delta(x^{(i)} = \textit{activité}) < P\right)$$

Cette fonction représente le mécanisme de restriction d'accès aux ressources. La sémantique de cette fonction est: *la permission d'accès est accordée si au moins une des ressources est libre*.

La libération des ressources se fait avec un taux distinct pour chaque automate $\mathcal{A}^{(i)}$ (μ_i). La libération, au contraire de la prise des ressources, se passe de façon indépendante.

La fonction d'atteignabilité de ce modèle est définie de façon analogue à la fonction de restriction à l'accès aux ressources (f). Un état est atteignable si *le nombre total de clients en possession d'une ressource est inférieur ou égal au nombre de ressources*. La fonction d'atteignabilité (F) est défini par

$$F(\tilde{x}) = \delta\left(\sum_{i=1}^{N} \delta(x^{(i)} = \textit{activité}) \leq P\right)$$

.

La matrice de transition locale de chaque automate $\mathcal{A}^{(i)}$ est:

$$Q_l^{(i)} = \begin{pmatrix} -\lambda_i f & \lambda_i f \\ \mu_i & -\mu_i \end{pmatrix}$$

N'ayant pas d'événements synchronisants, le descripteur de ce modèle est donc défini par la somme tensorielle:

$$Q = \bigoplus_{i=1}^{N}{}_g \, Q_l^{(i)}$$

La taille de l'espace d'état produit de ce modèle est égal à 2^N et le nombre d'états atteignables est égal à[3] $\sum_{i=0}^{P} C_N^i$. De ce fait, on peut noter que pour des valeurs de P

[2]Rappelons que $\delta(b)$ est une fonction qui vaut 1 si l'expression b est vrai, sinon cette fonction vaut 0.

[3]Notons C_a^b le nombre de combinaisons non ordonnées de taille b avec a éléments distincts. Sa définition numérique est $C_a^b = \frac{a!}{(a-b)!b!}$.

petites par rapport à N la proportion d'états atteignables par rapport au nombre total d'états de l'espace produit peut être très bas. Ainsi, ce modèle est plus efficace pour des cas où P est proche de N.

Pour illustrer cet exemple, prenons un cas avec trois clients et deux ressources ($N = 3$ et $P = 2$). Le descripteur correspondra à:

$$\begin{pmatrix} -\lambda f & \lambda f \\ \mu & -\mu \end{pmatrix} \bigoplus_g \begin{pmatrix} -\lambda f & \lambda f \\ \mu & -\mu \end{pmatrix} \bigoplus_g \begin{pmatrix} -\lambda f & \lambda f \\ \mu & -\mu \end{pmatrix} = Q$$

$$Q = \left(\begin{array}{cc|cc||cc|cc} -3\lambda & \lambda & \lambda & 0 & \lambda & 0 & 0 & 0 \\ \mu & -(2\lambda+\mu) & 0 & \lambda & 0 & \lambda & 0 & 0 \\ \hline \mu & 0 & -(2\lambda+\mu) & 0 & 0 & 0 & \lambda & 0 \\ 0 & \mu & 0 & -\mu & 0 & 0 & 0 & 0 \\ \hline\hline \mu & 0 & 0 & 0 & -(2\lambda+\mu) & \lambda & \lambda & 0 \\ 0 & \mu & 0 & 0 & \mu & -2\mu & 0 & 0 \\ \hline 0 & 0 & \mu & 0 & \mu & 0 & -2\mu & 0 \\ 0 & 0 & 0 & \mu & 0 & \mu & \mu & -3\mu \end{array}\right)$$

Notons que parmi les huit états globaux, seulement le dernier état n'est pas atteignable. Les états atteignables sont:

$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	position dans le générateur
repos	*repos*	*repos*	première ligne/colonne
repos	*repos*	*activité*	deuxième ligne/colonne
repos	*activité*	*repos*	troisième ligne/colonne
repos	*activité*	*activité*	quatrième ligne/colonne
activité	*repos*	*repos*	cinquième ligne/colonne
activité	*repos*	*activité*	sixième ligne/colonne
activité	*activité*	*repos*	septième ligne/colonne

4.2 Réseau de files d'attente fermé

Pour cet exemple, prenons un réseau fermé de Jackson [19, 50, 56], *i.e.*, un réseau fermé de N files d'attente (s_i avec $i \in 1..N$) caractérisées par un temps de service exponentiel μ_i. Le nombre clients dans le réseau est P. Un client ayant fini son service dans la file s_i se dirige vers la file s_j avec probabilité p_{ij}.

Le modèle RAS de cet exemple possède un automate pour chaque file s_i ($\mathcal{A}^{(i)}$) et un événement synchronisant pour chaque routage de file s_i vers la file file s_j possible ($p_{ij} \neq 0$) et non bouclé ($i \neq j$). L'identificateur d'un tel événement sera $i{\mapsto}j$. Par exemple, $2{\mapsto}3$ est l'événement représentant le départ d'un client de la file s_2 et l'arrivée de ce client dans la file s_3. Chaque événement $i{\mapsto}j$ aura comme maître l'automate $\mathcal{A}^{(i)}$ ($\mathcal{A}^{(j)}$ sera l'automate esclave). Les triplets de synchronisation des événements dans l'automate maître seront égaux à $(i{\mapsto}j, \mu_i p_{ij}, 1)$. Il faut remarquer que le taux d'occurrence de l'événement n'est pas le taux de service de la file s_i, mais son taux de service multiplié par la probabilité de routage correspondante.

Le nombre d'automates est égal à N, pourtant le nombre d'événements synchronisants peut s'élever à $N \times (N-1)$. La taille de chaque automate est donnée par le nombre P de clients dans le réseau. Chaque automate possédera $P+1$ états locaux représentant le nombre de clients quelle contient - de 0 (file vide) jusqu'à P (file pleine).

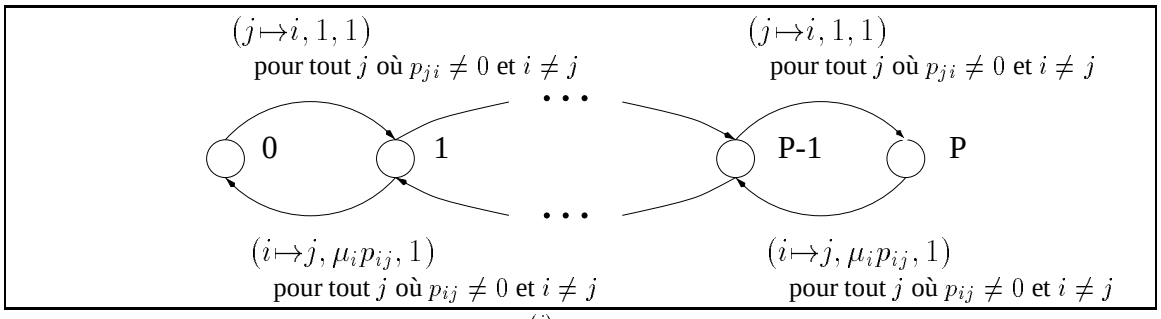


FIG. 4.3: L'automate $\mathcal{A}^{(i)}$ d'un réseau fermé de Jackson

La figure 4.3 représente l'automate $\mathcal{A}^{(i)}$ ($i \in 1..N$). Dans cette figure les étiquettes des arcs supérieurs représentent l'arrivée des clients dans la file s_i. Les arcs inférieurs représentent le départ des clients de la file s_i. Les étiquettes des deux types d'arc sont des listes de triplets de synchronisation.

Chaque triplet (e, τ_e, π_e) de la liste d'événements de l'arrivée est du type $(j \mapsto i, 1, 1)$ pour tout j où la probabilité p_{ji} est non nulle et $i \neq j$. Les événements $j \mapsto i$ (l'arrivée d'un client venu de la file s_j) ont comme automate maître l'automate $\mathcal{A}^{(j)}$, donc le taux d'occurrence de l'événement n'est pas défini dans ce triplet.

La liste d'événements de départ contiendra autant de triplets de synchronisation que le nombre de possibilités de destination différente de i des clients après le service à la file s_i. Chaque triplet de synchronisation est défini comme $(i \mapsto j, \mu_i p_{ij}, 1)$, car l'automate $\mathcal{A}^{(i)}$ est le maître de l'événement $i \mapsto j$.

La fonction d'atteignabilité de ce modèle est définie par le nombre de clients de chaque file. Ce nombre doit être toujours égal à P, car il s'agit d'un réseau fermé. La taille de l'espace d'état produit étant égal à $(P+1)^N$, le nombre d'états atteignables pour de grandes valeurs de N est proportionnellement très petit car égal à C_{P+N+1}^{N+1}. La fonction $F(\tilde{x})$ est définie par:

$$F(\tilde{x}) = \delta\left(\left[\sum_{i \in \eta} x^{(i)}\right] = P\right)$$

Pour illustrer cet exemple, prenons le réseau de la figure 4.4 où figurent trois files ($N = 3$) de taux de service (μ_0, μ_1 et μ_2) et le nombre de clients dans le réseau est égal à 2 ($P = 2$, $n_i = 3$). La figure 4.5 représente le modèle RAS équivalent.

La première caractéristique intéressante de ce modèle est l'absence d'événements locaux. En conséquence, la partie locale du descripteur (somme tensorielle) sera nulle. Les événements synchronisants de cet exemple correspondent:

- au départ d'un client de la file s_1 vers la file s_2 avec taux $\pi_1\mu_1$, nommé $1 \mapsto 2$;
- au départ d'un client de la file s_1 vers la file s_3 avec taux $\pi_2\mu_1$, nommé $1 \mapsto 3$;

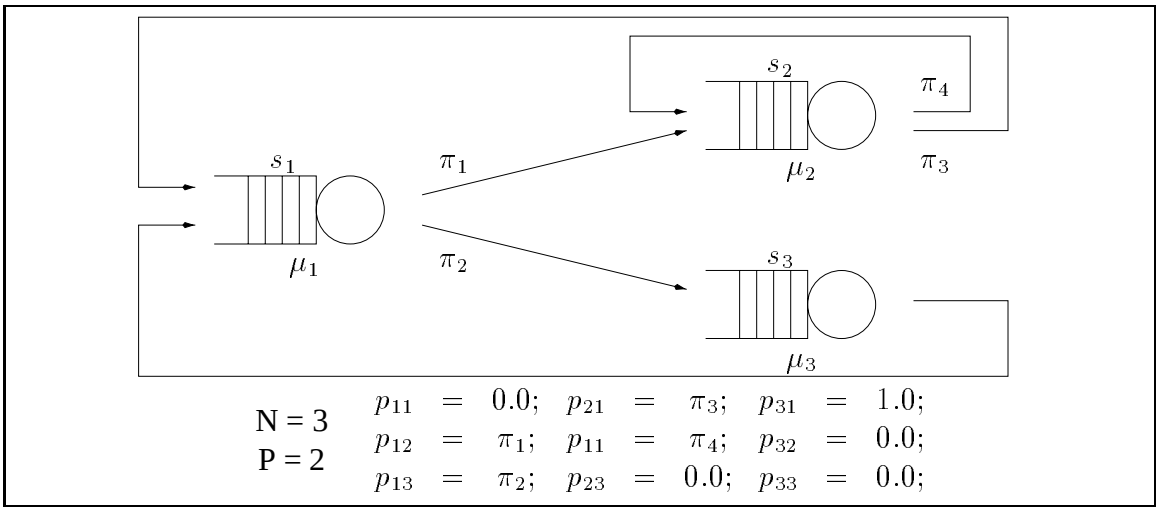


FIG. 4.4: Réseau fermé à trois files

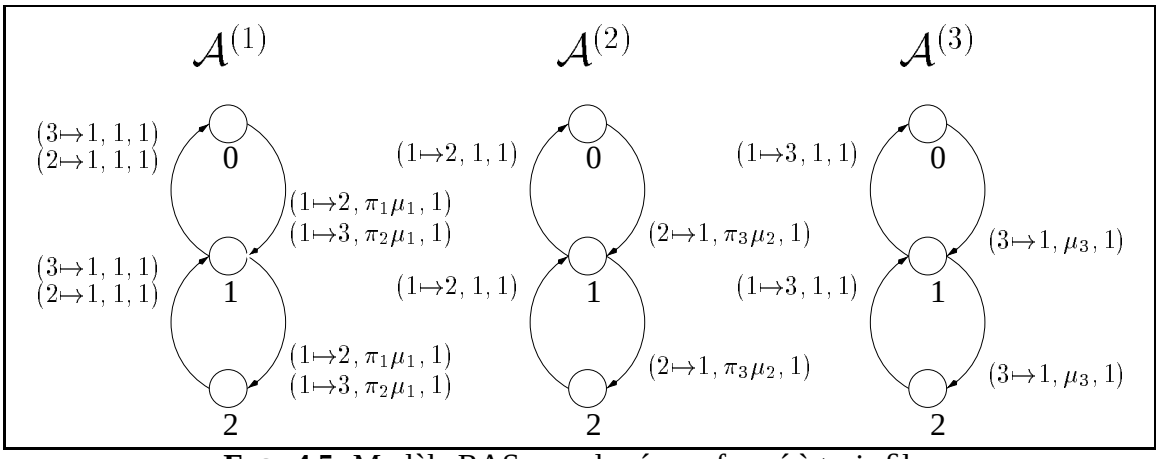


FIG. 4.5: Modèle RAS pour le réseau fermé à trois files

– au départ d'un client de la file s_2 vers la file s_1 avec taux $\pi_3\mu_2$, nommé $2\mapsto 1$;

– au départ d'un client de la file s_3 vers la file s_1 avec taux $1.0\mu_3$, nommé $3\mapsto 1$;

Le descripteur est obtenu par la formule:

$$Q = \sum_{e\in\varepsilon} \left(\bigotimes_{i\in\eta,\,g} Q_{e^+}^{(i)} + \bigotimes_{i\in\eta,\,g} Q_{e^-}^{(i)} \right)$$

où $\varepsilon = \{1\mapsto 2, 1\mapsto 3, 2\mapsto 1, 3\mapsto 1\}$.

Selon les règles de formation des matrices de la section précédente (page 62), on a les matrices suivantes pour l'exemple de la figure 4.5.

$$Q^{(1)}_{1\mapsto 2^+} = \begin{pmatrix} 0 & 0 & 0 \\ \pi_1\mu_1 & 0 & 0 \\ 0 & \pi_1\mu_1 & 0 \end{pmatrix} \quad Q^{(2)}_{1\mapsto 2^+} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad Q^{(3)}_{1\mapsto 2^+} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$Q^{(1)}_{1\mapsto 2^-} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\pi_1\mu_1 & 0 \\ 0 & 0 & -\pi_1\mu_1 \end{pmatrix} \quad Q^{(2)}_{1\mapsto 2^-} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad Q^{(3)}_{1\mapsto 2^-} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$Q^{(1)}_{1\mapsto 3^+} = \begin{pmatrix} 0 & 0 & 0 \\ \pi_2\mu_1 & 0 & 0 \\ 0 & \pi_2\mu_1 & 0 \end{pmatrix} \quad Q^{(2)}_{1\mapsto 3^+} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad Q^{(3)}_{1\mapsto 3^+} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

$$Q^{(1)}_{1\mapsto 3^-} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\pi_2\mu_1 & 0 \\ 0 & 0 & -\pi_2\mu_1 \end{pmatrix} \quad Q^{(2)}_{1\mapsto 3^-} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad Q^{(3)}_{1\mapsto 3^-} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$Q^{(1)}_{2\mapsto 1^+} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad Q^{(2)}_{2\mapsto 1^+} = \begin{pmatrix} 0 & 0 & 0 \\ \pi_3\mu_2 & 0 & 0 \\ 0 & \pi_3\mu_2 & 0 \end{pmatrix} \quad Q^{(3)}_{2\mapsto 1^+} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$Q^{(1)}_{2\mapsto 1^-} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad Q^{(2)}_{2\mapsto 1^-} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\pi_3\mu_2 & 0 \\ 0 & 0 & -\pi_3\mu_2 \end{pmatrix} \quad Q^{(3)}_{2\mapsto 1^-} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$Q^{(1)}_{3\mapsto 1^+} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad Q^{(2)}_{3\mapsto 1^+} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad Q^{(3)}_{3\mapsto 1^+} = \begin{pmatrix} 0 & 0 & 0 \\ \mu_3 & 0 & 0 \\ 0 & \mu_3 & 0 \end{pmatrix}$$

$$Q^{(1)}_{3\mapsto 1^-} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad Q^{(2)}_{3\mapsto 1^-} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad Q^{(3)}_{3\mapsto 1^-} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\mu_3 & 0 \\ 0 & 0 & -\mu_3 \end{pmatrix}$$

Le générateur infinitésimal correspondant à ce descripteur est présenté dans la page 72. Dans cette matrice les éléments nuls ne sont pas représentés et les éléments diagonaux sont représentés par $\bullet$. La valeur numérique des éléments diagonaux est l'opposé de la somme des autres éléments de la ligne (la somme des éléments d'une ligne vaut zéro).

Le nombre d'états globaux atteignables de ce modèle est égal à 6, tandis que l'espace produit est égal à 27. Les états $\tilde{x}$ atteignables sont:

$\pi_4\mu_3$
$\pi_3\mu_2$
$\pi_2\mu_1$
$\pi_1\mu_1$

$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	position dans le générateur
0	0	2	troisième ligne/colonne
0	1	1	cinquième ligne/colonne
0	2	1	septième ligne/colonne
1	0	1	onzième ligne/colonne
1	1	0	treizième ligne/colonne
2	0	0	dix-neuvième ligne/colonne

4.3 Réseau de Files d'Attente Ouvert

Dans cette section on présente d'abord le cas général des réseaux de files d'attente avec blocage et priorité [29, 56, 21]. Ensuite, les modifications nécessaires pour inclure les mécanismes de perte et de routage probabiliste [101, 106, 65] sont présentées pour des cas précis.

4.3.1 Réseau de Files d'Attente Ouvert avec Blocage et Priorité

Supposons l'existence de K classes distinctes de clients visitant C serveurs (files) nommés s_i avec $i \in [1..C]$. Les caractéristiques de chaque file s_i sont:

- le nombre de clients (toutes classes confondues) dans l'attente de service peut aller de 0 à $C_i - 1$;
- le taux exponentiel d'arrivée des clients de la classe k venus de l'extérieur est égal à λ_i^k;
- le taux exponentiel de service des clients de la classe k est égal à μ_i^k;
- la probabilité de départ des clients de la classe k vers le serveur s_j est égal à p_{ij}^k;
- la probabilité de départ des clients de la classe k vers l'extérieur est égal à $p_{i\chi}^k$;

Le modèle RAS de cet exemple représente chaque file s_i avec autant d'automates que le nombre de classes de clients qui peuvent visiter cette file. On va appeler $\mathcal{A}^{(ik)}$ l'automate représentant les clients de la classe k dans la file s_i. La taille de chaque automate $\mathcal{A}^{(ik)}$ est, indépendamment de la classe k, égale à C_i.

Le nombre d'événements synchronisants sera égal au nombre de probabilités de routages p_{ij}^k non nulles (avec i et j variant selon le nombre de files et k variant selon le nombre de classes). Chaque événement représentant le routage d'un client de la classe k de la file s_i vers la file s_j possible ($p_{ij}^k \neq 0$) et non bouclé ($i \neq j$) sera identifié $i \overset{k}{\mapsto} j$.

Définissons pour chaque groupe d'automates représentant une même file s_i une fonction de restriction des arrivés selon la capacité de la file (C_i).

$$f^{(i)}(\tilde{x}) = \delta\left(\left[\sum_{k=1}^{K} x^{(ik)}\right] < C_i - 1\right)$$

L'interprétation de cette fonction est *un client ne peut arriver dans une file s_i que si le nombre des clients dans tous les automates qui représentent cette file ($\mathcal{A}^{(ik)}$, où*

$k \in [1..K]$*) n'a pas encore atteint la capacité de la file*. Cette fonction représente le mécanisme de blocage.

Définissons pour chaque file s_i une priorité entre les classes k de façon à ce qu'un client ne puisse être servi que si il n'y a aucun client d'une classe prioritaire à lui. Cette priorité peut être exprimée par une fonction de l'état des autres automates qui représentent la même file. Supposons que la priorité entre les classes soit faite selon l'indice de la classe (la classe $k = 1$ est prioritaire sur la classe $k = 2$ qui est elle-même prioritaire sur la classe $k = 3$ et ainsi de suite). Dans ce cas, la fonction blocage à cause de la priorité pour un automate $\mathcal{A}^{(ik)}$ peut être exprimée comme

$$g^{(ik)}(\tilde{x}) = \delta\left(\left[\sum_{m=1}^{k-1} x^{(im)}\right] = 0\right)$$

L'interprétation de cette fonction est *il ne doit pas y avoir de clients prioritaires à la classe* k *(*$\mathcal{A}^{(im)}$*, où* $m \in [1..k-1]$*) dans la file* s_i. Notons que dans ce cas, $g^{(i1)}(\tilde{x}) = 1$, car la classe $k = 1$ est prioritaire sur toutes les autres classes. Bien que cette politique de choix des clients prioritaires soit assez générique pour la plupart des cas pratiques, n'importe quelle autre politique peut être utilisée sans perdre la généralité du modèle. Il est seulement nécessaire de pouvoir exprimer la fonction adéquate à la politique désirée. Notons que des politiques de priorité distinctes peuvent ainsi être définies pour chaque file s_i.

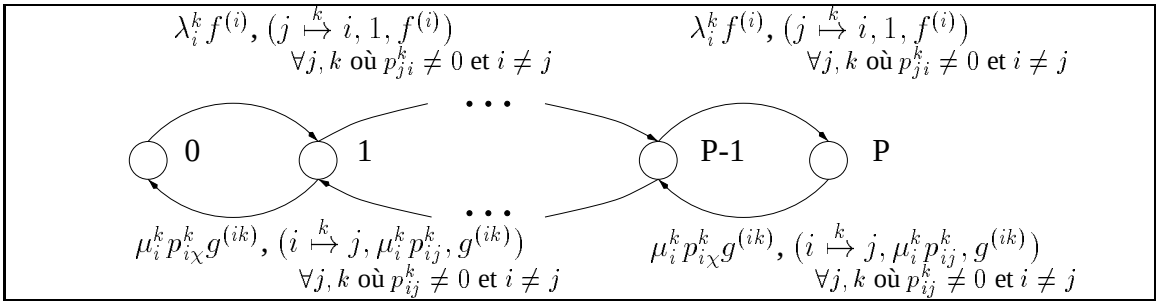


FIG. 4.6: L'automate $\mathcal{A}^{(ik)}$ d'un réseau ouvert avec blocage

La figure 4.6 représente de façon générique l'automate $\mathcal{A}^{(ik)}$ (clients de la classe k dans la file s_i). Les étiquettes des arcs supérieurs représentent l'arrivée des clients, tandis que le départ est représenté par les étiquettes des arcs inférieurs.

Les étiquettes représentant l'arrivée des clients sont composées d'un taux local (λ_i^k) multiplié par la fonction qui autorise l'*admission* des clients dans la file s_i ($f^{(i)}$) et d'une liste de triplets de synchronisation. Dans cette liste il existe un triplet pour chaque file s_j d'où un client de la classe k peut arriver ($p_{ji}^k \neq 0$). L'automate $\mathcal{A}^{(ik)}$ n'étant pas le maître des événements $j \overset{k}{\mapsto} i$, les taux utilisés sont égaux à 1 et la probabilité d'occurrence de l'événement est la fonction $f^{(i)}$ (restriction d'admission).

Les étiquettes représentant le départ de clients sont composées d'un taux local pour la sortie vers l'extérieur multiplié par la fonction établissant la priorité entre les classes ($\mu_i^k p_{i\chi}^k g^{(ik)}$) et une liste de triplets de synchronisation. Chaque triplet correspond à l'une

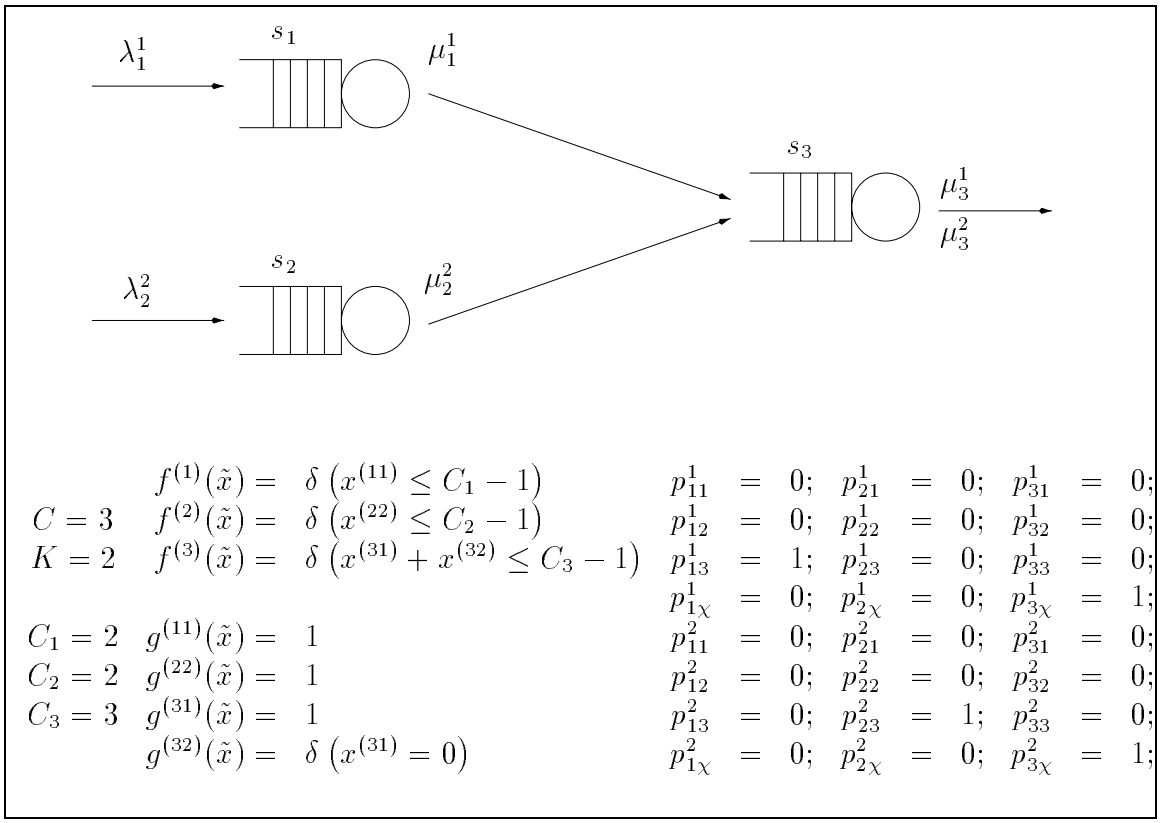
des destinations s_j possibles ($p_{ij}^k \neq 0$ et $i \neq j$). Le taux de chaque événement est donné par le taux de service correspondant ($\mu_i^k p_{ij}^k$) et la probabilité d'occurrence de l'événement est la fonction établissant la priorité entre classes ($g^{(ik)}$).

Les états atteignables du modèle sont définis pour chaque groupe d'automates décrivant une même file s_i. La fonction F définit les états atteignables.

$$F(\tilde{x}) = \prod_{i=1}^{C} \delta \left(\left[\sum_{k=1}^{K} x^{(ik)} \right] \leq C_i - 1 \right)$$

L'interprétation de cette fonction est *un état global est atteignable si tout les automates représentant une même file s_i ont un nombre de clients inférieur ou égal à la capacité de la file*. Ceci doit être vrai pour tous les files (produit des restrictions).

La taille de l'espace produit du modèle est égale au produit de la taille de tous les automates $\mathcal{A}^{(ik)}$, avec $i \in [1..C]$ et $k \in [1..K]$. Pourtant le nombre d'états atteignables est réduit à cause de l'utilisation de plusieurs automates pour une même file. L'espace produit de chaque groupe d'automates représentant une même file s_i est égal à $(C_i)^{\kappa^{(i)}}$, où $\kappa^{(i)}$ est le nombre de classes qui visitent la file s_i (nombre d'automates utilisés pour décrire la file s_i). Cependant, le nombre d'états atteignables pour ce groupe d'automates est seulement $C^{C_i}_{\kappa^{(i)}+C_i}$. De cette façon, le nombre total d'états atteignables est égal à $\prod_{i=1}^{C} C^{C_i}_{\kappa^{(i)}+C_i}$ (le produit des états atteignables de chaque groupe d'automates décrivant une file s_i).



$$\begin{array}{lll}
 & f^{(1)}(\tilde{x}) = & \delta\left(x^{(11)} \leq C_1 - 1\right) \\
C = 3 & f^{(2)}(\tilde{x}) = & \delta\left(x^{(22)} \leq C_2 - 1\right) \\
K = 2 & f^{(3)}(\tilde{x}) = & \delta\left(x^{(31)} + x^{(32)} \leq C_3 - 1\right) \\
 & & \\
C_1 = 2 & g^{(11)}(\tilde{x}) = & 1 \\
C_2 = 2 & g^{(22)}(\tilde{x}) = & 1 \\
C_3 = 3 & g^{(31)}(\tilde{x}) = & 1 \\
 & g^{(32)}(\tilde{x}) = & \delta\left(x^{(31)} = 0\right)
\end{array}$$

$$\begin{array}{llllllllll}
p_{11}^1 = 0; & p_{21}^1 = 0; & p_{31}^1 = 0; \\
p_{12}^1 = 0; & p_{22}^1 = 0; & p_{32}^1 = 0; \\
p_{13}^1 = 1; & p_{23}^1 = 0; & p_{33}^1 = 0; \\
p_{1\chi}^1 = 0; & p_{2\chi}^1 = 0; & p_{3\chi}^1 = 1; \\
p_{11}^2 = 0; & p_{21}^2 = 0; & p_{31}^2 = 0; \\
p_{12}^2 = 0; & p_{22}^2 = 0; & p_{32}^2 = 0; \\
p_{13}^2 = 0; & p_{23}^2 = 1; & p_{33}^2 = 0; \\
p_{1\chi}^2 = 0; & p_{2\chi}^2 = 0; & p_{3\chi}^2 = 1;
\end{array}$$

FIG. 4.7: Réseau ouvert à trois files avec blocage et priorité

Prenons par exemple, le réseau avec trois files de la figure 4.7 avec deux classes de clients. Les clients de la classe 1 arrivent de l'extérieur avec un taux λ_1^1, sont servis dans la file s_1 et se dirigent vers la file s_3. Les clients de la classe 2 arrivent par la file s_2, sont servis dans cette file et se dirigent également vers la file s_3. Dans cette file les clients de la classe 1 sont servis avant les clients de la classe 2. Après leur service les deux types de clients sortent vers l'extérieur.

Notons que les files s_1 et s_2 ne reçoivent qu'une seule classe de clients, donc un seul automate est nécessaire pour chacune des files. Pour cette même raison, les fonctions de restriction du nombre de clients dans les automates représentant ces files ($f^{(1)}$ et $f^{(2)}$) sont constantes, car leur évaluation ne dépend que de l'état de l'automate où elles sont utilisées. La fonction décrivant la priorité des clients dans la file s_3 est définie par $g^{(32)}(\tilde{x}) = \delta\left(x^{(31)} = 0\right)$, car un client de la classe 2 ne peut être servi que s'il n'y a aucun client de la classe 1.

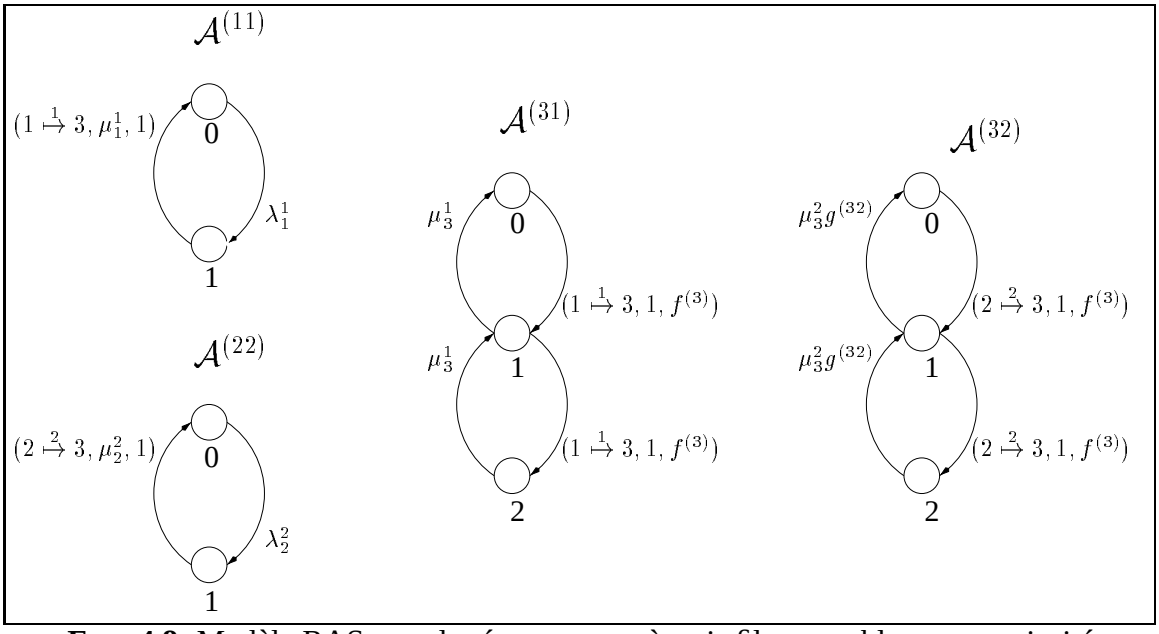


FIG. 4.8: Modèle RAS pour le réseau ouvert à trois files avec blocage et priorité

Les automates pour ce modèle sont décrit dans la figure 4.8. Les matrices locales du descripteur Markovien sont:

$$Q_l^{(11)} = \begin{pmatrix} -\lambda_1^1 & \lambda_1^1 \\ 0 & 0 \end{pmatrix} \qquad Q_l^{(22)} = \begin{pmatrix} -\lambda_2^2 & \lambda_2^2 \\ 0 & 0 \end{pmatrix}$$

$$Q_l^{(31)} = \begin{pmatrix} 0 & 0 & 0 \\ \mu_3^1 & -\mu_3^1 & 0 \\ 0 & \mu_3^1 & -\mu_3^1 \end{pmatrix} \qquad Q_l^{(32)} = \begin{pmatrix} 0 & 0 & 0 \\ \mu_3^2 g^{(32)} & -\mu_3^2 g^{(32)} & 0 \\ 0 & \mu_3^2 g^{(32)} & -\mu_3^2 g^{(32)} \end{pmatrix}$$

Les matrices correspondant à l'événement $1 \overset{1}{\mapsto} 3$ sont:

$$
\begin{array}{lll}
Q^{(11)}_{1\overset{1}{\mapsto}3^+} = \begin{pmatrix} 0 & 0 \\ \mu_1^1 & 0 \end{pmatrix} & Q^{(11)}_{1\overset{1}{\mapsto}3^-} = \begin{pmatrix} 0 & 0 \\ 0 & -\mu_1^1 \end{pmatrix} \\
Q^{(22)}_{1\overset{1}{\mapsto}3^+} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & Q^{(22)}_{1\overset{1}{\mapsto}3^-} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\
Q^{(31)}_{1\overset{1}{\mapsto}3^+} = \begin{pmatrix} 0 & f^{(3)} & 0 \\ 0 & 0 & f^{(3)} \\ 0 & 0 & 0 \end{pmatrix} & Q^{(31)}_{1\overset{1}{\mapsto}3^-} = \begin{pmatrix} f^{(3)} & 0 & 0 \\ 0 & f^{(3)} & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
Q^{(32)}_{1\overset{1}{\mapsto}3^+} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & Q^{(32)}_{1\overset{1}{\mapsto}3^-} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}
\end{array}
$$

Les matrices correspondant à l'événement $2 \overset{2}{\mapsto} 3$ sont:

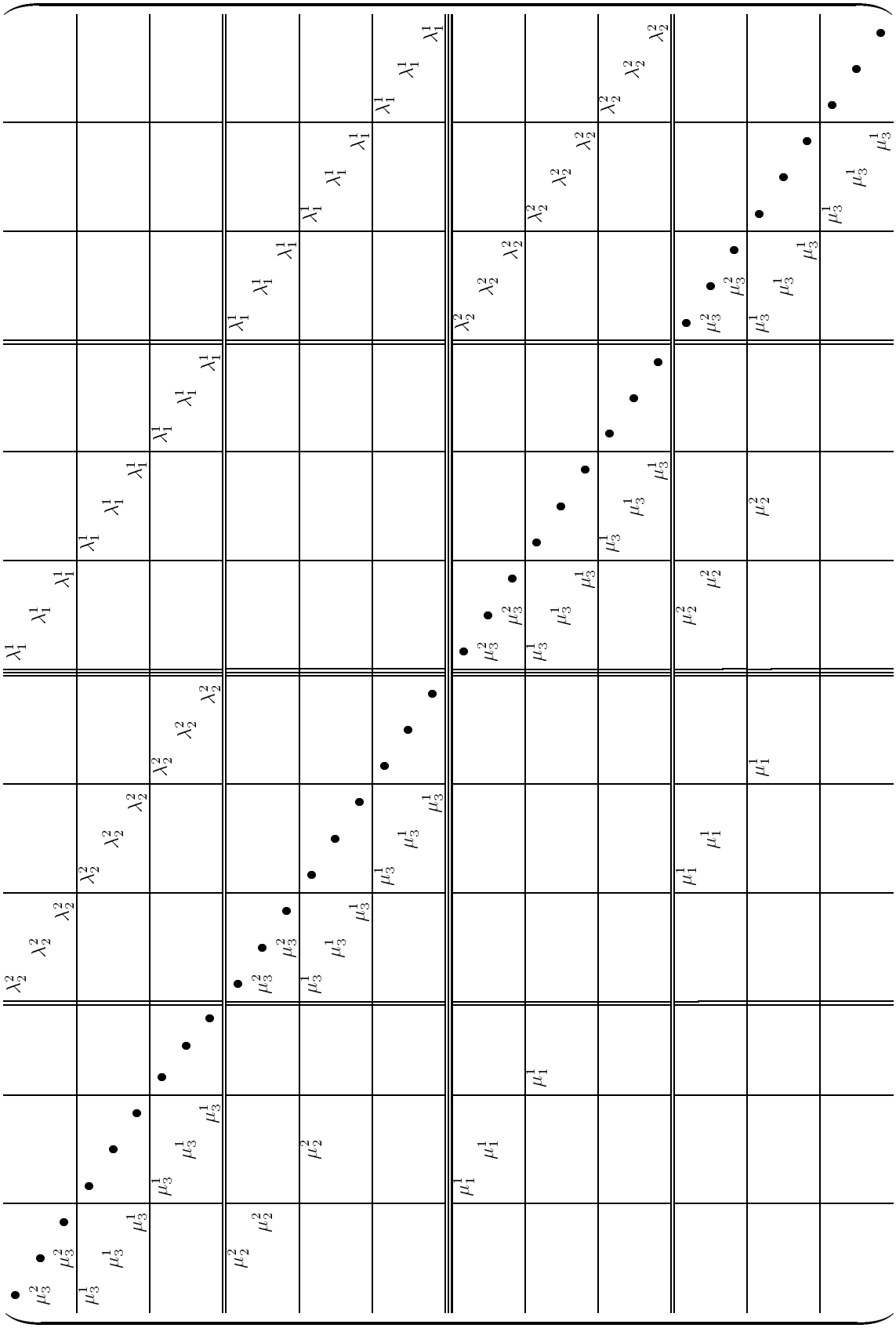
$$
\begin{array}{lll}
Q^{(11)}_{2\overset{2}{\mapsto}3^+} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & Q^{(11)}_{2\overset{2}{\mapsto}3^-} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\
Q^{(22)}_{2\overset{2}{\mapsto}3^+} = \begin{pmatrix} 0 & 0 \\ \mu_2^2 & 0 \end{pmatrix} & Q^{(22)}_{2\overset{2}{\mapsto}3^-} = \begin{pmatrix} 0 & 0 \\ 0 & -\mu_2^2 \end{pmatrix} \\
Q^{(31)}_{2\overset{2}{\mapsto}3^+} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & Q^{(31)}_{2\overset{2}{\mapsto}3^-} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
Q^{(32)}_{2\overset{2}{\mapsto}3^+} = \begin{pmatrix} 0 & f^{(3)} & 0 \\ 0 & 0 & f^{(3)} \\ 0 & 0 & 0 \end{pmatrix} & Q^{(32)}_{2\overset{2}{\mapsto}3^-} = \begin{pmatrix} f^{(3)} & 0 & 0 \\ 0 & f^{(3)} & 0 \\ 0 & 0 & 0 \end{pmatrix}
\end{array}
$$

Le descripteur de ce modèle est donné par

$$
\begin{aligned}
Q = {} & Q_l^{(11)} \underset{g}{\oplus} Q_l^{(22)} \underset{g}{\oplus} Q_l^{(31)} \underset{g}{\oplus} Q_l^{(32)} \\
& + Q^{(11)}_{1\overset{1}{\mapsto}3^+} \underset{g}{\otimes} Q^{(22)}_{1\overset{1}{\mapsto}3^+} \underset{g}{\otimes} Q^{(31)}_{1\overset{1}{\mapsto}3^+} \underset{g}{\otimes} Q^{(32)}_{1\overset{1}{\mapsto}3^+} \\
& + Q^{(11)}_{1\overset{1}{\mapsto}3^-} \underset{g}{\otimes} Q^{(22)}_{1\overset{1}{\mapsto}3^-} \underset{g}{\otimes} Q^{(31)}_{1\overset{1}{\mapsto}3^-} \underset{g}{\otimes} Q^{(32)}_{1\overset{1}{\mapsto}3^-} \\
& + Q^{(11)}_{2\overset{2}{\mapsto}3^+} \underset{g}{\otimes} Q^{(22)}_{2\overset{2}{\mapsto}3^+} \underset{g}{\otimes} Q^{(31)}_{2\overset{2}{\mapsto}3^+} \underset{g}{\otimes} Q^{(32)}_{2\overset{2}{\mapsto}3^+} \\
& + Q^{(11)}_{2\overset{2}{\mapsto}3^-} \underset{g}{\otimes} Q^{(22)}_{2\overset{2}{\mapsto}3^-} \underset{g}{\otimes} Q^{(31)}_{2\overset{2}{\mapsto}3^-} \underset{g}{\otimes} Q^{(32)}_{2\overset{2}{\mapsto}3^-}
\end{aligned}
$$

Le générateur infinitésimal correspondant à ce descripteur est présenté dans la page 78. Dans cette matrice les éléments nuls ne sont pas représentés et les éléments diagonaux sont représentés par $\bullet$. La valeur numérique des éléments diagonaux est l'opposé de la somme des autres éléments de la ligne (la somme des éléments d'une ligne vaut zéro).

Le nombre d'états atteignables de ce modèle est égal à 24 (2 états pour la file s_1, 2 états pour la file s_2 et 6 états pour la file s_3) pour un total de 36 états dans l'espace produit. Les états globaux $\tilde{x}$ non atteignables sont les états où la somme des valeurs de $x^{(31)}$ et $x^{(32)}$ dépassent la capacité de file s_3 ($C_3 - 1$):



$x^{(11)}$	$x^{(22)}$	$x^{(31)}$	$x^{(32)}$	$x^{(11)}$	$x^{(22)}$	$x^{(31)}$	$x^{(32)}$
0	0	1	2	1	0	1	2
0	0	2	1	1	0	2	1
0	0	2	2	1	0	2	2
0	1	1	2	1	1	1	2
0	1	2	1	1	1	2	1
0	1	2	2	1	1	2	2

4.3.2 Mécanisme de Perte

Le mécanisme de perte représente une alternative au blocage d'une des files. Le départ d'un client vers l'extérieur dû au manque de place dans la prochaine file d'attente provoque la perte du client. Dans l'exemple cité précédemment, supposons que les clients sortant de la file s_2 vont à la file s_3 seulement si cette dernière n'est pas pleine, si tel n'est pas le cas les clients sont perdus. La figure 4.9 représente ce nouveau modèle. Le RAS décrivant ce modèle aura des différences au niveau de l'événement synchronisant $2 \overset{2}{\mapsto} 3$. La figure 4.10 montre le RAS pour ce nouveau modèle.

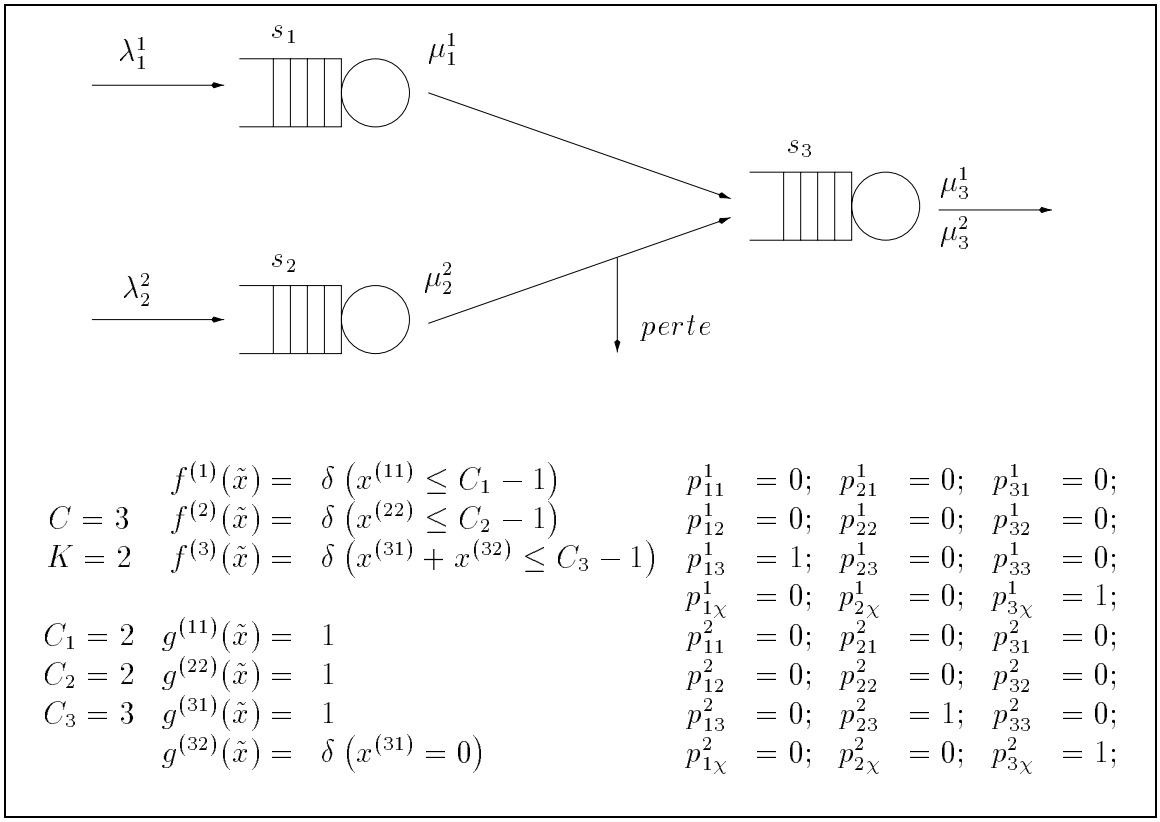


FIG. 4.9: Réseau ouvert à trois files avec blocage, perte et priorité

L'événement $2 \overset{2}{\mapsto} 3$ représentera maintenant, non seulement le mouvement d'un client de la classe 2 de la file s_2 vers la file s_3, mais aussi la *perte* d'un client. Cette perte intervient selon le nombre total de clients de la file s_2 exprimé, comme avant, par la

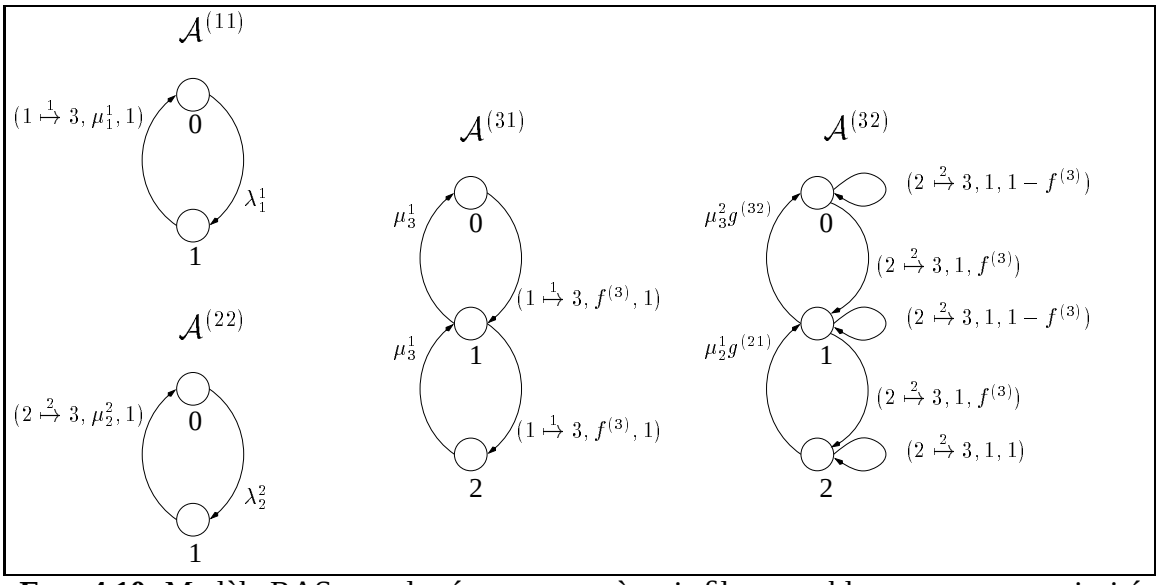


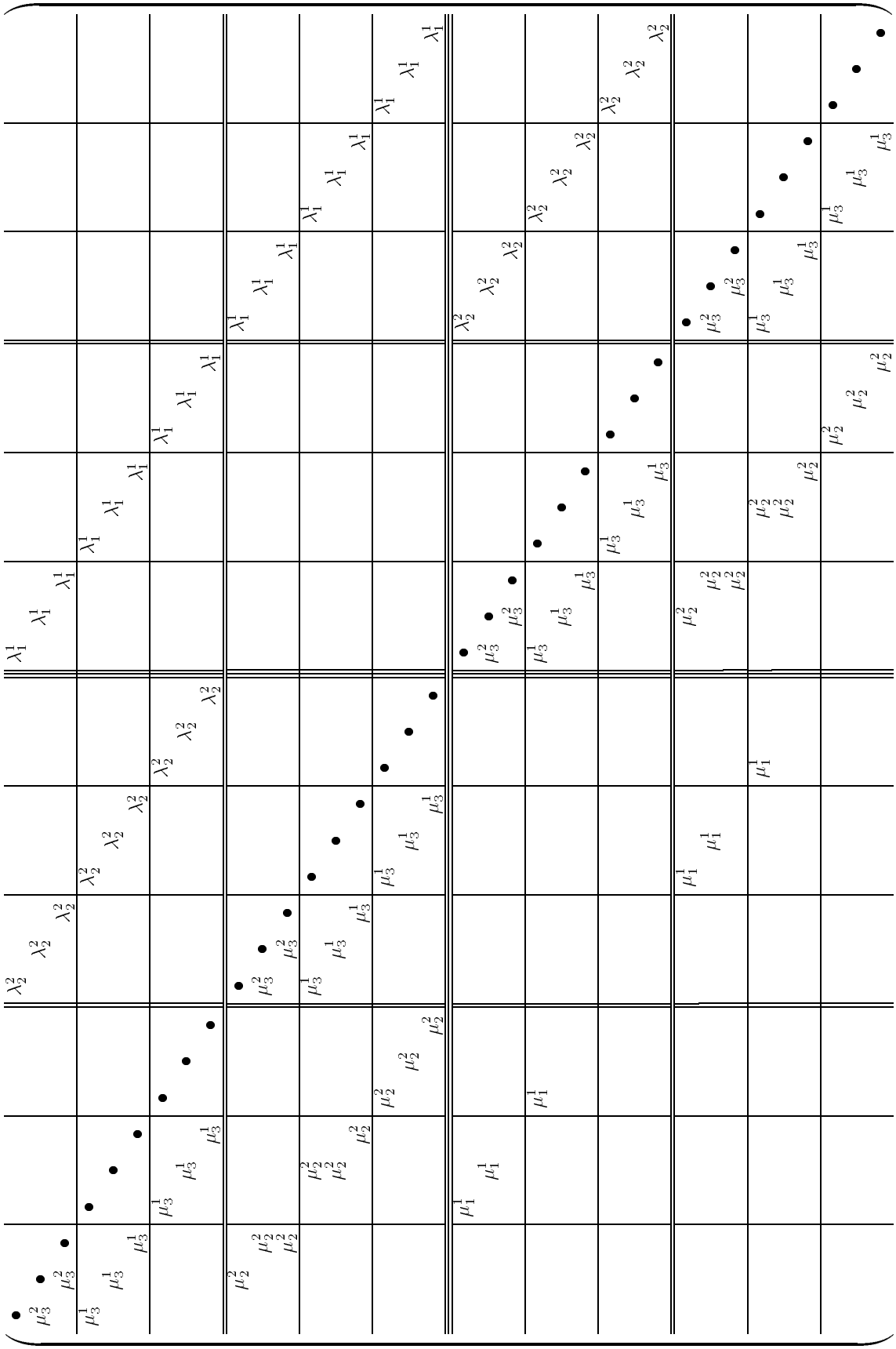
FIG. 4.10: Modèle RAS pour le réseau ouvert à trois files avec blocage, perte et priorité

fonction $f^{(3)}$. Chaque état de l'automate $\mathcal{A}^{(32)}$ possède deux possibilités de transitions synchronisées avec l'événement $2 \overset{2}{\mapsto} 3$. Si la file s_3 n'est pas pleine ($f^{(3)}$), l'événement représente l'arrivée d'un client sorti de file s_2. Si la file s_3 est pleine ($1-f^{(3)}$), l'événement représente la perte d'un client sorti de file s_2. Dans les deux cas, le taux des triplets est égal à 1, car l'automate $\mathcal{A}^{(32)}$ est un automate esclave pour l'événement $2 \overset{2}{\mapsto} 3$. Notons que pour le dernier état de l'automate $\mathcal{A}^{(32)}$ (état 2) la fonction $f^{(3)}$ est toujours évaluée à 0 (s'il y a deux clients de la classe 2 dans la file s_3, cette file est pleine). De ce fait, seulement un type de synchronisation peut se passer: la perte d'un client sorti de la file s_2.

L'inclusion de ce mécanisme de perte, comme il a été dit avant, ne change que les informations concernant l'événement $2 \overset{2}{\mapsto} 3$. Les nouvelles matrices correspondant à cet événement sont:

$$
\begin{aligned}
Q^{(11)}_{2\overset{2}{\mapsto}3^+} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & Q^{(11)}_{2\overset{2}{\mapsto}3^-} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\
Q^{(22)}_{2\overset{2}{\mapsto}3^+} &= \begin{pmatrix} 0 & 0 \\ \mu_2^2 & 0 \end{pmatrix} & Q^{(22)}_{2\overset{2}{\mapsto}3^-} &= \begin{pmatrix} 0 & 0 \\ 0 & -\mu_2^2 \end{pmatrix} \\
Q^{(31)}_{2\overset{2}{\mapsto}3^+} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & Q^{(31)}_{2\overset{2}{\mapsto}3^-} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
Q^{(32)}_{2\overset{2}{\mapsto}3^+} &= \begin{pmatrix} 1-f^{(3)} & f^{(3)} & 0 \\ 0 & 1-f^{(3)} & f^{(3)} \\ 0 & 0 & 1 \end{pmatrix} & Q^{(32)}_{2\overset{2}{\mapsto}3^-} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

L'espace produit, ainsi que les états atteignables ne changent pas non plus. Le changement des matrices fait que le descripteur de ce nouveau modèle est égal à la matrice présenté dans la page 81.



4.3.3 Mécanisme de Routage Variable

Pour décrire ce mécanisme on va adopter un exemple différent des précédentes. Prenons un réseau ouvert avec N files. Chaque file s_i ($i \in [1..N]$) possède une capacité $C_i - 1$ et un taux de service μ_i. Tous les clients arrivent de l'extérieur vers la file s_1 avec un taux d'arrivée λ. Après être servis dans la file s_1 chaque client se dirige vers une des $N-1$ files, dites *files secondaires*. Le choix de la file secondaire est fait selon leur nombre de clients. Si deux ou plusieurs files secondaires possèdent le même nombre minimal de clients, celle avec le plus petit indice est choisie. Si toutes les files secondaires sont pleines, le client sortant de la file s_1 abandonne le système (perte). Après être servis par l'une des files secondaires, les clients abandonnent le système. La figure 4.11 présente de façon générale ce modèle.

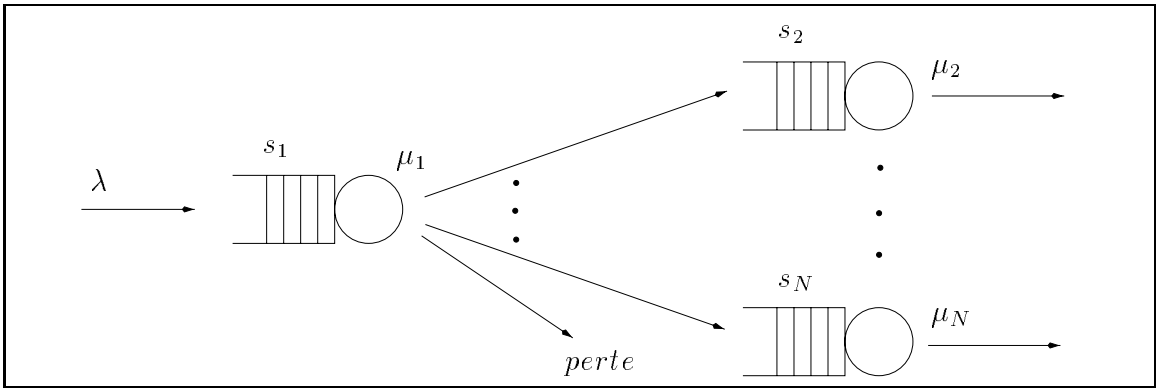


FIG. 4.11: Réseau ouvert avec routage variable

Le choix de la file avec le plus petit nombre de clients est défini par un ensemble de fonctions $f^{(i)}$ qui signifient que *la file s_i est choisie pour recevoir le prochain client si elle est la moins pleine ou si elle est l'une des moins pleines et a le plus petit indice*. Les fonctions $f^{(i)}$ sont définis par:

$$f^{(i)}(\tilde{x}) = \prod_{j=2, j\neq i}^{N} \delta\left[x^{(i)} < x^{(j)}\right] \quad + \quad \prod_{j=2, j\neq i}^{N} \delta\left[(x^{(i)} \leq x^{(j)}) et (i < j)\right]$$

Cet exemple est modélisé par un automate pour chaque file s_i. L'automate de la première file (s_1) est décrit dans la figure 4.12. Les étiquettes des arcs représentant l'arrivée des clients de l'extérieur ont seulement un taux local égal à λ. Le départ d'un client est représenté par un événement synchronisant nommé s. Cet événement représentera une synchronisation avec tous les autres automates du modèle. Cependant, dans seulement un de ces automates la synchronisation causera un changement de son état local.

La figure 4.13 représente de façon générale les automates $\mathcal{A}^{(i)}$ ($i \in [1..N]$). Dans ces automates le départ d'un client est une transition avec un seul taux local (μ_i). L'événement s représentant le départ d'un client de la file s_1 peut correspondre soit à l'arrivée d'un client dans la file s_i, si $f^{(i)}$ est égal à 1, soit au maintien de l'état local courant (boucle), si $f^{(i)}$ est égal à 0 ($1 - f^{(i)}$ est égal à 1). Notons que si la file est pleine (l'automate $\mathcal{A}^{(i)}$

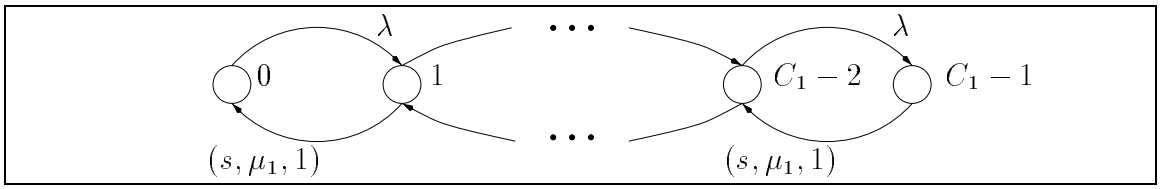


FIG. 4.12: Automate $\mathcal{A}^{(1)}$ représentant la file s_1

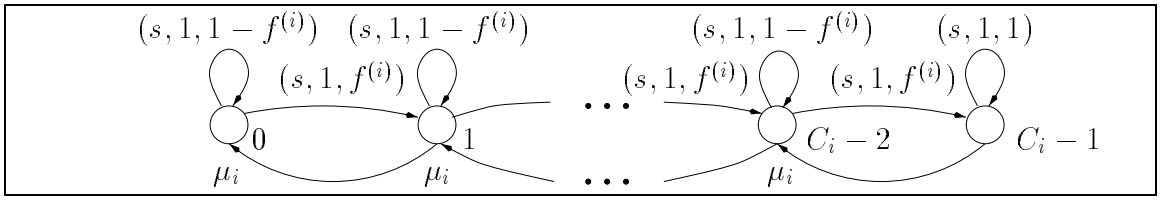


FIG. 4.13: Automate $\mathcal{A}^{(i)}$ représentant la file s_i ($i \in [2..N]$)

est dans l'état $C_i - 1$), la seule option de synchronisation est le maintien de l'état local de l'automate.

Dans cet exemple tous les états sont atteignables, donc la fonction d'atteignabilité (F) est:

$$F(\tilde{x}) = 1$$

Le nombre total d'états de ce modèle est égal au produit de la taille de chaque automate:

$$\prod_{i=1}^{N} C_i$$

Pour illustrer cet exemple, supposons un système avec trois files secondaires ($N = 4$) avec des capacités 2, 1 et 1 ($C_2 = 3$, $C_3 = 2$ et $C_4 = 2$) et la première file (s_1) avec capacité 2 ($C_1 = 3$). La figure 4.14 représente ce système. Le réseaux pour cet exemple est présenté dans la figure 4.15.

Les matrices locales de ce modèle sont:

$$Q_l^{(1)} = \begin{pmatrix} -\lambda & \lambda & 0 \\ 0 & -\lambda & \lambda \\ 0 & 0 & 0 \end{pmatrix} \quad Q_l^{(2)} = \begin{pmatrix} 0 & 0 & 0 \\ \mu_2 & -\mu_2 & 0 \\ 0 & \mu_2 & -\mu_2 \end{pmatrix}$$

$$Q_l^{(3)} = \begin{pmatrix} 0 & 0 \\ \mu_3 & -\mu_3 \end{pmatrix} \quad Q_l^{(4)} = \begin{pmatrix} 0 & 0 \\ \mu_4 & -\mu_4 \end{pmatrix}$$

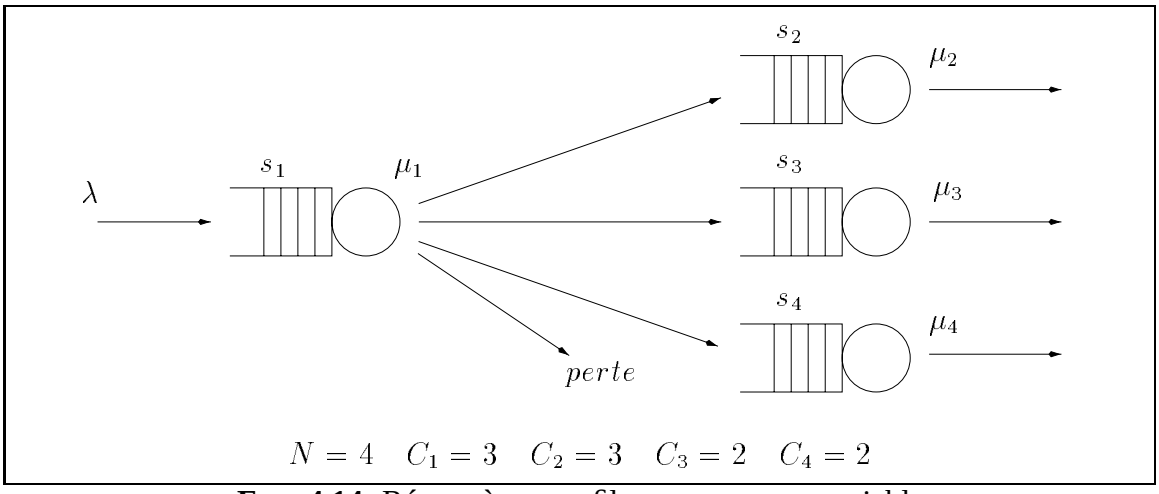


FIG. 4.14: Réseau à quatre files avec routage variable

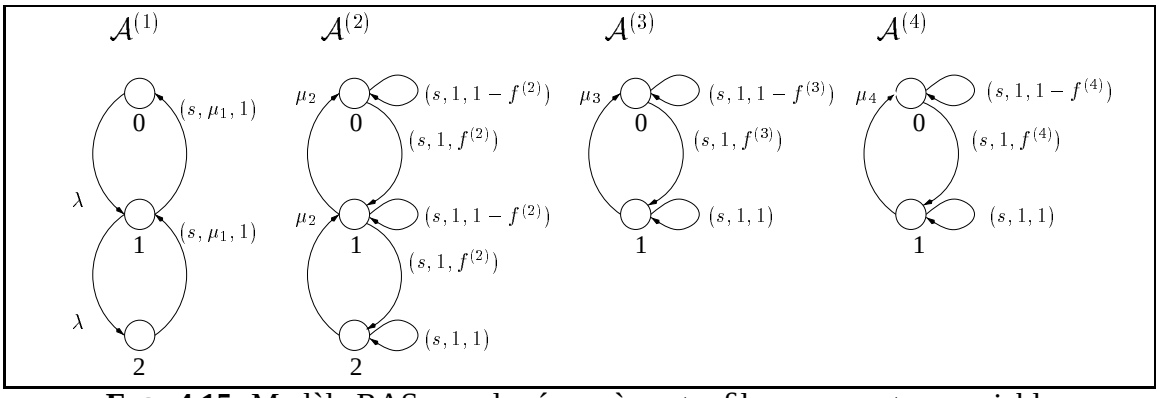


FIG. 4.15: Modèle RAS pour le réseau à quatre files avec routage variable

Les matrices correspondant à l'événement s sont:

$$Q_{s^+}^{(1)} = \begin{pmatrix} 0 & 0 & 0 \\ \mu_0 & 0 & 0 \\ 0 & \mu_0 & 0 \end{pmatrix} \qquad Q_{s^-}^{(1)} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\mu_0 & 0 \\ 0 & 0 & -\mu_0 \end{pmatrix}$$

$$Q_{s^+}^{(2)} = \begin{pmatrix} 1-f^{(2)} & f^{(2)} & 0 \\ 0 & 1-f^{(2)} & f^{(2)} \\ 0 & 0 & 1 \end{pmatrix} \qquad Q_{s^-}^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$Q_{s^+}^{(3)} = \begin{pmatrix} 1-f^{(3)} & f^{(3)} \\ 0 & 1 \end{pmatrix} \qquad Q_{s^-}^{(3)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$Q_{s^+}^{(4)} = \begin{pmatrix} 1-f^{(4)} & f^{(4)} \\ 0 & 1 \end{pmatrix} \qquad Q_{s^-}^{(4)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Le descripteur de ce modèle est:

$$\begin{aligned} Q &= Q_l^{(1)} \bigoplus_g Q_l^{(2)} \bigoplus_g Q_l^{(3)} \bigoplus_g Q_l^{(4)} \\ &+ Q_{s^+}^{(1)} \bigotimes_g Q_{s^+}^{(2)} \bigotimes_g Q_{s^+}^{(3)} \bigotimes_g Q_{s^+}^{(4)} \\ &+ Q_{s^-}^{(1)} \bigotimes_g Q_{s^-}^{(2)} \bigotimes_g Q_{s^-}^{(3)} \bigotimes_g Q_{s^-}^{(4)} \end{aligned}$$

Le générateur infinitésimal correspondant à ce descripteur est présenté dans la page 86. La représentation de cette matrice est identique à celle des matrices précédentes. Ce modèle possède 36 états dans son espace produit. Tous les états globaux sont atteignables.

Nous pensons que ces exemples auront permis au lecteur de mieux comprendre la modélisation avec les RAS ainsi que la construction des descripteurs. Le chapitre suivant montre l'efficacité numérique que l'on peut tirer d'une telle représentation. Les exemples présentés ici seront utilisés dans les expériences numériques des deux prochains chapitres.

$$
\left(
\begin{array}{*{36}{c}}
\bullet &&&&& &&&&& && \lambda \\
\mu_4 & \bullet &&&&& &&&&& && \lambda \\
\mu_3 && \bullet &&&&& &&&&& && \lambda \\
& \mu_3 & \mu_4 & \bullet &&&&& &&&&& && \lambda \\
\mu_2 &&&& \bullet &&&&& &&&&& && \lambda \\
& \mu_2 &&& \mu_4 & \bullet &&&&& &&&&& && \lambda \\
&& \mu_2 && \mu_3 && \bullet &&&&& &&&&& && \lambda \\
&&& \mu_2 && \mu_3 & \mu_4 & \bullet &&&&& &&&&& && \lambda \\
&&&& \mu_2 &&&& \bullet &&&&& &&&&& && \lambda \\
&&&&& \mu_2 &&& \mu_4 & \bullet &&&&& &&&&& && \lambda \\
&&&&& & \mu_2 && \mu_3 && \bullet &&&&& &&&&& && \lambda \\
&&&&& && \mu_2 && \mu_3 & \mu_4 & \bullet &&&&& &&&&& && \lambda \\
&&&& \mu_1 &&&&& &&& \bullet &&&&& &&&&& && \lambda \\
&&&&& \mu_1 &&&&& && \mu_4 & \bullet &&&&& &&&&& && \lambda \\
&&&&& & \mu_1 &&&&& & \mu_3 && \bullet &&&&& &&&&& && \lambda \\
&&&&& && \mu_1 &&&&& & \mu_3 & \mu_4 & \bullet &&&&& &&&&& && \lambda \\
&&&&& & \mu_1 &&&&& & \mu_2 &&&& \bullet &&&&& &&&&& && \lambda \\
&&&&& && \mu_1 &&&&& & \mu_2 &&& \mu_4 & \bullet &&&&& &&&&& && \lambda \\
&&&&& && \mu_1 &&&&& && \mu_2 && \mu_3 && \bullet &&&&& &&&&& && \lambda \\
&&&&& && \mu_1 &&&&& &&& \mu_2 && \mu_3 & \mu_4 & \bullet &&&&& &&&&& && \lambda \\
&&&&& &&&& \mu_1 &&&&& && \mu_2 &&&& \bullet &&&&& &&&&& && \lambda \\
&&&&& &&&&& \mu_1 &&&&& && \mu_2 &&& \mu_4 & \bullet &&&&& &&&&& && \lambda \\
&&&&& &&&&& \mu_1 &&&&& &&& \mu_2 && \mu_3 && \bullet &&&&& &&&&& && \lambda \\
&&&&& &&&&& \mu_1 &&&&& &&&& \mu_2 && \mu_3 & \mu_4 & \bullet &&&&& &&&&& && \lambda \\
&&&&& &&&&& &&&&& &&&& \mu_1 &&&&& \bullet \\
&&&&& &&&&& &&&&& &&&&& \mu_1 &&&& \mu_4 & \bullet \\
&&&&& &&&&& &&&&& &&&&& & \mu_1 &&& \mu_3 && \bullet \\
&&&&& &&&&& &&&&& &&&&& && \mu_1 &&& \mu_3 & \mu_4 & \bullet \\
&&&&& &&&&& &&&&& &&&&& & \mu_1 &&&&& && \bullet \\
&&&&& &&&&& &&&&& &&&&& && \mu_1 &&&&& & \mu_4 & \bullet \\
&&&&& &&&&& &&&&& &&&&& && \mu_1 &&&&& & \mu_3 && \bullet \\
&&&&& &&&&& &&&&& &&&&& && \mu_1 &&&&& && \mu_3 & \mu_4 & \bullet \\
&&&&& &&&&& &&&&& &&&&& &&&& \mu_1 &&&&& &&& \bullet \\
&&&&& &&&&& &&&&& &&&&& &&&&& \mu_1 &&&&& && \mu_4 & \bullet \\
&&&&& &&&&& &&&&& &&&&& &&&&& \mu_1 &&&&& && \mu_3 && \bullet \\
&&&&& &&&&& &&&&& &&&&& &&&&& \mu_1 &&&&& &&& \mu_3 & \mu_4 & \bullet
\end{array}
\right)
$$

Deuxième partie

Méthodes Numériques

Chapitre 5

Multiplication Vecteur-Descripteur

Les cibles principales des RAS sont les problèmes à grands espaces d'états et pour ces problèmes les méthodes itératives sont les plus adéquates. La multiplication d'un vecteur de probabilités par le générateur est l'opération fondamentale de toutes les méthodes itératives [5, 99, 33]. Dans ce chapitre on présente des considérations sur l'efficacité du produit vecteur-descripteur, ainsi que les options algorithmiques capables d'optimiser cette tâche.

La multiplication d'un vecteur v par un descripteur Q (équation 3.2, page 63) est

$$vQ = v \sum_{j=1}^{(N+2E)} \left[\bigotimes_{i=1}^{N}{}_{g} Q_j^{(i)} \right] = \sum_{j=1}^{(N+2E)} \left[v \bigotimes_{i=1}^{N}{}_{g} Q_j^{(i)} \right]$$

donc, l'opération de base qui nous intéresse est

$$v \bigotimes_{i=1}^{N}{}_{g} Q^{(i)}$$

où les indices j ont été omis pour les matrices $Q_j^{(i)}$ afin de simplifier la notation. De cette façon, les algorithmes présentés dans ce chapitre font la multiplication d'un vecteur par un terme produit tensoriel généralisé. Ce terme est composé d'une suite de N matrices notées $Q^{(i)}$ avec $i \in [1..N]$, chacune reliée à un automate $\mathcal{A}^{(i)}$.

La première section de ce chapitre rappelle [5, 81] l'algorithme de base de la multiplication d'un vecteur par un produit tensoriel. Cet algorithme est employé pour les produits tensoriels qui n'ont pas d'éléments fonctionnels (produits tensoriels classiques). La deuxième section montre les contraintes à observer lors de l'occurrence des éléments fonctionnels dans le terme produit tensoriel (produits tensoriels généralisés). Finalement, la troisième section présente les algorithmes alternatifs pour optimiser le traitement du descripteur dans sa totalité.

5.1 Cas Sans Éléments Fonctionnels

Le cas le plus simple de multiplication vecteur-produit tensoriel est le cas où les matrices ne contiennent pas d'éléments fonctionnels. Dans ce cas, on doit calculer:

$$v \bigotimes_{i=1}^{N} Q^{(i)}$$

5.1.1 Multiplication des Facteurs Normaux

Selon la propriété de décomposition des produits tensoriels (équation 2.16, page 23) tout produit tensoriel de N matrices est équivalent au produit de N facteurs normaux. En utilisant cette propriété pour le terme $\bigotimes_{i=1}^{N} Q^{(i)}$:

$$\begin{aligned} Q^{(1)} \otimes Q^{(2)} \otimes \ldots \otimes Q^{(N-1)} \otimes Q^{(N)} = \\ & Q^{(1)} \otimes I_{nright_1} \\ & \times I_{nleft_2} \otimes Q^{(2)} \otimes I_{nright_2} \\ & \times \cdots \\ & \times I_{nleft_{N-1}} \otimes Q^{(N-1)} \otimes I_{nright_{N-1}} \\ & \times I_{nleft_N} \otimes Q^{(N)} \end{aligned}$$

On rappelle les définitions des suites finies de matrices[1] introduites dans le chapitre 2:

✍ **Soit**

n_i la dimension de la i-ème matrice d'une suite;

$nleft_i$ le produit des dimensions de toutes les matrices à gauche de la i-ème matrice d'une suite, *i.e.*, $\prod_{k=1}^{i-1} n_k$ (cas particulier: $nleft_1 = 1$);

$nright_i$ le produit des dimensions de toutes les matrices à droite de la i-ème matrice d'une suite, *i.e.*, $\prod_{k=i+1}^{N} n_k$ (cas particulier: $nright_N = 1$);

$\bar{n}_i$ le produit des dimensions de toutes les matrices sauf la i-ème matrice d'une suite, *i.e.*, $\prod_{k=1,k\neq i}^{N} n_k$ ($\bar{n}_i = nleft_i\ nright_i$);

Pour calculer la multiplication d'un vecteur v par le terme $\bigotimes_{i=1}^{N} Q^{(i)}$ il est nécessaire et suffisant de savoir multiplier un vecteur par un facteur normal. Le vecteur v doit être multiplié par le premier facteur normal, le résultat est multiplié par le deuxième facteur normal et ainsi de suite jusqu'au dernier des facteurs normaux. Ceci est possible grâce à la propriété d'associativité de la multiplication (traditionnelle) de matrices. De plus, la propriété de commutativité entre facteurs normaux (équation 2.19, page 26), permet la multiplication des facteurs normaux dans un ordre quelconque.

[1] Dans le reste de ce document les *suites finies de matrices* seront appelées seulement *suites de matrices*. Car, seules les suites finies seront abordées.

Multiplication du Dernier Facteur Normal

$$v \times I_{nleft_N} \otimes Q^{(N)}$$

La matrice $I_{nleft_N} \otimes Q^{(N)}$ étant une matrice bloc diagonale (figure 5.1), il suffit de multiplier $nleft_N$ tranches de taille n_N du vecteur v par la matrice $Q^{(N)}$. L'algorithme pour cette multiplication (algorithme 5.1) est présenté page 92. Cet algorithme correspond à la construction des tranches de taille n_N du vecteur v et leur multiplication par $Q^{(N)}$ comme décrit par la figure 5.2.

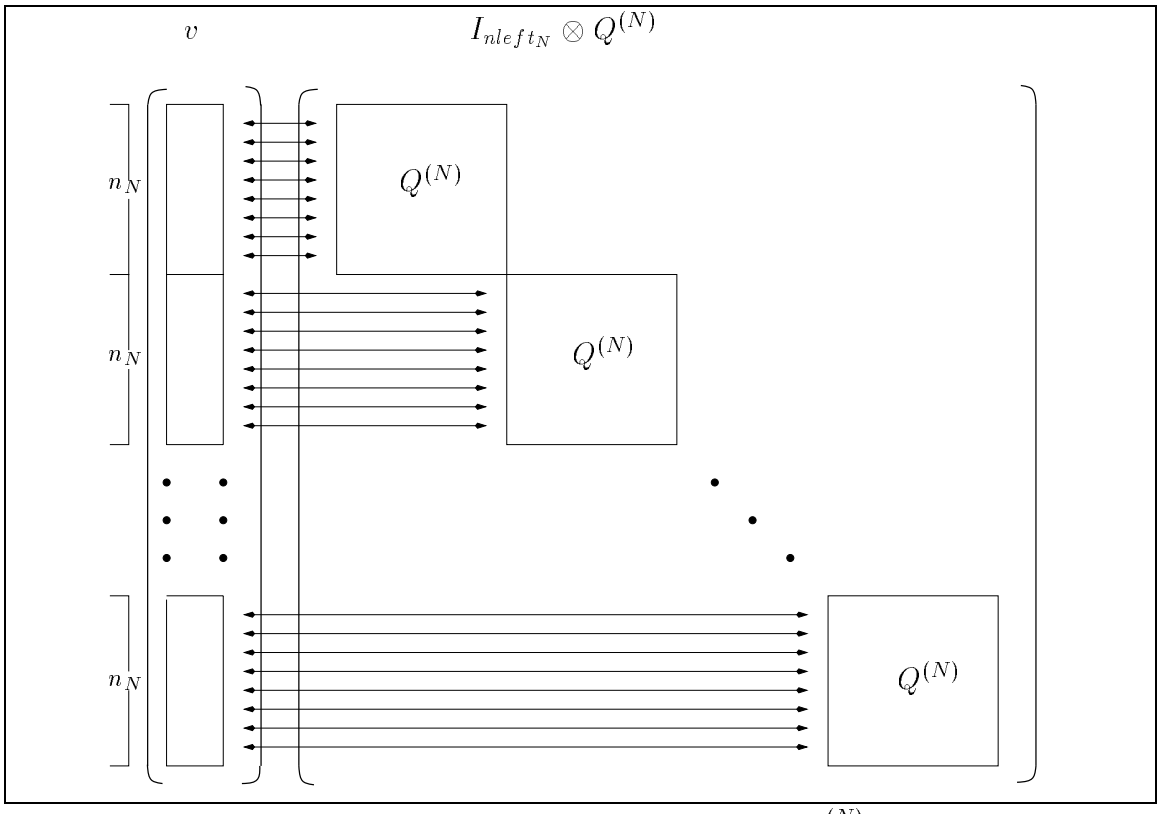


FIG. 5.1: Multiplication $v \times I_{nleft_N} \otimes Q^{(N)}$

Multiplication du Premier Facteur Normal

$$v \times Q^{(1)} \otimes I_{nright_1}$$

Le premier facteur normal, grâce à la propriété de pseudo-commutativité (équation 2.17, page 23), est égal à:

$$P_\sigma \times I_{nright_1} \otimes Q^{(1)} {P_\sigma}^T$$

où σ est la permutation qui passe de $\{1, 2, 3, \ldots, N\}$ à $\{2, 3, \ldots, N, 1\}$.

Algorithme 5.1

```
1   base = 0;
2   for k = 1, 2, ... , nleft_N
3   do  index = base + 1;
4       for l = 1, 2, ... , n_N
5       do  z_in[l] = v[index];
6           index = index + 1;
7       end do
8       multiply   z_out = z_in × Q^(N);
9       index = base + 1;
10      for l = 1, 2, ... , n_N
11      do  v[index] = z_out[l];
12          index = index + 1;
13      end do
14      base = base + n_N;
15  end do
```

Algorithme 5.1: Multiplication $v \times I_{nleft_N} \otimes Q^{(N)}$

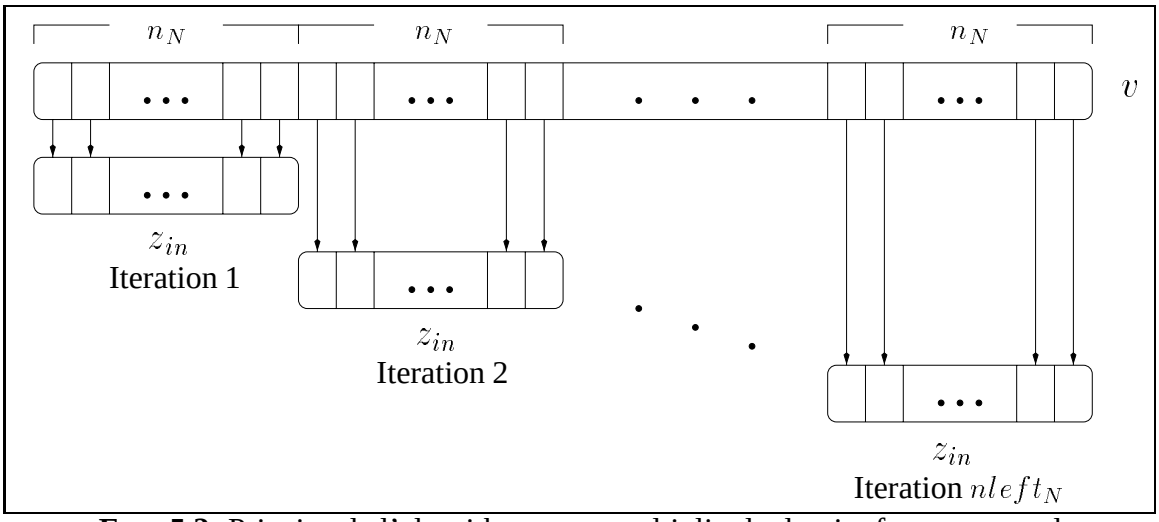


FIG. 5.2: Principe de l'algorithme pour multiplier le dernier facteur normal

Cette formulation peut être calculée de façon analogue au cas précédent (multiplication du dernier facteur normal). La multiplication d'un vecteur v est exécutée en trois étapes:

- la multiplication de v par la matrice de permutation P_σ;
- la multiplication de $(v \times P_\sigma)$ par le facteur normal commuté $I_{nright_1} \otimes Q^{(1)}$; et
- la multiplication du résultat des étapes précédentes par la matrice de permutation ${P_\sigma}^T$.

La première étape correspond à une permutation du vecteur v. Il est possible d'exécuter cette permutation lors de l'extraction de tranches du vecteur v, *i.e.*, dans les remplissages des vecteurs z_{in} de l'algorithme (lignes 3 à 7 de l'algorithme 5.1). Dans le cas

non permuté (multiplication du dernier facteur normal), le vecteur z_{in} est rempli avec des tranches successives de taille n_N. La permutation nécessaire pour le premier facteur normal, en revanche, équivaut à accéder au vecteur en prenant un élément à chaque intervalle de taille $nright_1$. La figure 5.3 représente le processus de permutation, qu'il faut comparer au processus équivalent pour le cas sans permutation (décrit dans la figure 5.2). La raison de cette permutation peut être compris en observant le format de la matrice $Q^{(1)} \otimes I_{nright_1}$:

$$Q^{(1)} \otimes I_{nright_1} = \begin{pmatrix} q^{(1)}_{1,1} I_{nright_1} & q^{(1)}_{1,2} I_{nright_1} & \cdots & q^{(1)}_{1,n_1} I_{nright_1} \\ q^{(1)}_{2,1} I_{nright_1} & q^{(1)}_{2,2} I_{nright_1} & \cdots & q^{(1)}_{2,n_1} I_{nright_1} \\ \vdots & \vdots & \ddots & \vdots \\ q^{(1)}_{n_1,1} I_{nright_1} & q^{(1)}_{n_1 1,2} I_{nright_1} & \cdots & q^{(1)}_{n_1 1,n_1} I_{nright_1} \end{pmatrix}$$

La deuxième étape est identique au cas précédent, *i.e.*, la multiplication répétée des tranches du vecteur v par la matrice $Q^{(1)}$. La troisième étape est la permutation symétrique à la première et peut, donc, être exécutée lors du stockage des éléments du temporaire z_{out} (lignes 9 à 13 de l'algorithme 5.1). L'algorithme pour cette multiplication est numéroté 5.2.

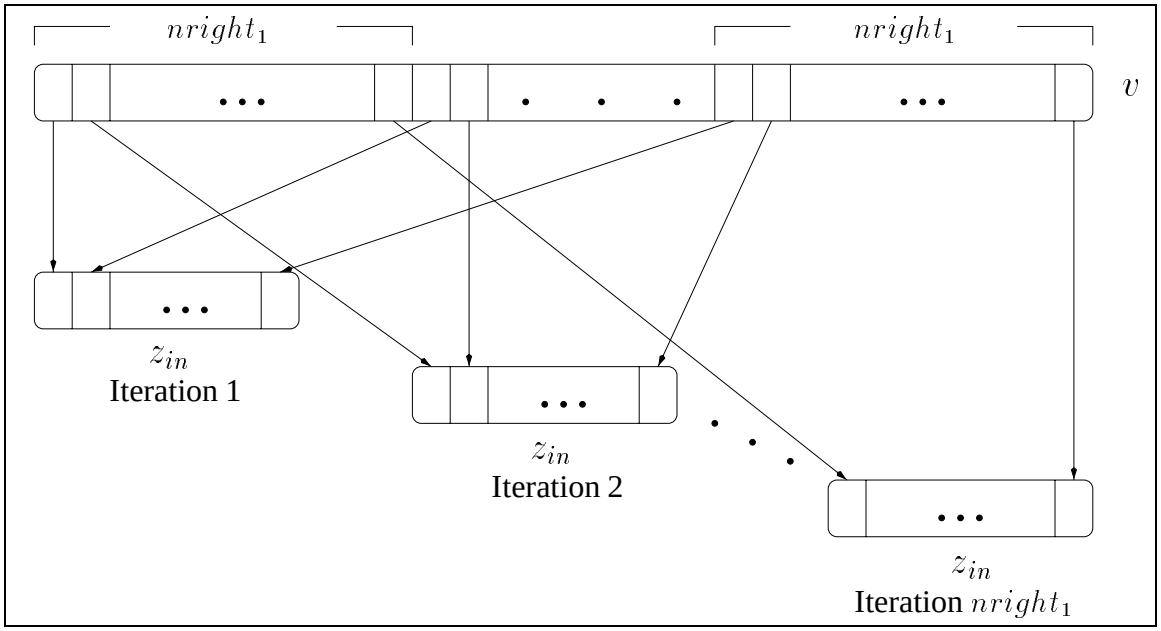


FIG. 5.3: Permutations exécutés lors de la multiplication du premier facteur normal

Multiplication Complète

Les autres facteurs normaux (du deuxième jusqu'à l'avant-dernier) sont traités comme des combinaisons des deux cas précédents. La technique de base consiste toujours à ap-

Algorithme 5.2

```
1   base = 0;
2   for j = 1, 2, ... , nright_1
3   do  index = base + j;
4       for l = 1, 2, ... , n_1
5       do  z_in[l] = v[index];
6           index = index + nright_1;
7       end do
8       multiply   z_out = z_in × Q^(1);
9       index = base + j;
10      for l = 1, 2, ... , n_1
11      do  v[index] = z_out[l];
12          index = index + nright_1;
13      end do
14  end do
```

Algorithme 5.2: Multiplication $v \times Q^{(1)} \otimes I_{nright_1}$

pliquer la propriété de pseudo-commutativité (équation 2.17, page 23) au facteur normal:

$$I_{nleft_i} \otimes Q^{(i)} \otimes I_{nright_i} = P_\sigma \times \left(I_{nleft_i} \otimes I_{nright_i} \otimes Q^{(i)}\right) {P_\sigma}^T$$

où σ est la permutation qui passe de $\{1, \ldots, i-1, i, i+1, \ldots, N\}$ à $\{1, \ldots, i-1, i+1, \ldots, N, i\}$.

Ceci amène à multiplier toujours des facteurs normaux de la forme:

$$I_{nleft_i} \otimes I_{nright_i} \otimes Q^{(i)} = I_{\bar{n}_i} \otimes Q^{(i)}$$

Les permutations sont faites implicitement selon la position de la matrice $Q^{(i)}$. L'algorithme 5.3 résout la multiplication d'un vecteur v par un produit tensoriel $\otimes_{i=1}^{N} Q^{(i)}$. Dans cet algorithme les facteurs normaux sont traités du premier jusqu'au dernier. Pourtant, selon la propriété de commutativité de facteurs normaux (équation 2.19, page 26), n'importe quel autre ordre aurait pu être employé.

Complexité

La complexité du produit d'un vecteur par un terme produit tensoriel classique est obtenue en observant le nombre de multiplications vecteur-matrice exécuté (ligne 10 de l'algorithme 5.3). À chaque boucle en i de l'algorithme $nleft_i \times nright_i$ produits vecteur-matrice sont exécutés avec des matrices de taille n_i. En supposant les matrices $Q^{(i)}$ pleines, le nombre de multiplications pour chaque produit vecteur-matrice est égal à

Algorithme 5.3

```
1  for i = 1, 2, ..., N
2  do  base = 0;
3      for k = 1, 2, ..., nleft_i
4      do  for j = 1, 2, ..., nright_i
5          do  index = base + j;
6              for l = 1, 2, ..., n_i
7              do  z_in[l] = v[index];
8                  index = index + nright_i;
9              end do
10             multiply    z_out = z_in × Q^(i);
11             index = base + j;
12             for l = 1, 2, ..., n_i
13             do  v[index] = z_out[l];
14                 index = index + nright_i;
15             end do
16         end do
17         base = base + (nright_i × n_i);
18     end do
19 end do
```

Algorithme 5.3: Multiplication $v \times \otimes_{i=1}^{N} Q^{(i)}$

$(n_i)^2$. La complexité de l'algorithme 5.3 est[2]:

$$\sum_{i=1}^{N} \left(\bar{n}_i \times (n_i)^2\right) = \prod_{i=1}^{N} n_i \times \sum_{i=1}^{N} n_i \tag{5.1}$$

Ceci est à comparer avec une multiplication $v \bigotimes_{i=1}^{N} Q^{(i)}$ qui consiste d'abord à calculer $Q = \bigotimes_{i=1}^{N} Q^{(i)}$, puis à multiplier par le vecteur v dont la complexité est de l'ordre de $(\prod_{i=1}^{N} n_i)^2$. Si les matrices $Q^{(i)}$ sont stockées dans un format creux, le nombre de multiplications pour chaque produit vecteur-matrice est, en général, bien inférieur à $(n_i)^2$. Dans ce cas, si nz_i est le nombre d'éléments non nuls de la matrice $Q^{(i)}$, la complexité de l'algorithme 5.3 est:

$$\sum_{i=1}^{N} \left(\bar{n}_i \times nz_i\right) = \prod_{i=1}^{N} n_i \times \sum_{i=1}^{N} \frac{nz_i}{n_i} \tag{5.2}$$

Pour avoir une idée plus précise des coûts de calcul de cet algorithme, établissons une comparaison de cette complexité avec la multiplication d'un vecteur par une matrice unique stockée en format creux. La matrice équivalente à un terme produit tensoriel possédera $\prod_{i=1}^{N} nz_i$ éléments non nuls, donc la complexité sera de cet ordre. La comparaison

[2]Rappelons que par la définition des RAS, ainsi que pour la définition des suites de matrices, $\bar{n}_i \times n_i = nleft_i \times nright_i \times n_i = \prod_{i=1}^{N} n_i$.

entre deux formules aussi différentes est difficile. Fixons un limite de remplissage des matrices $Q^{(i)}$:

$$nz_i = n_i(N)^{\frac{1}{N-1}}$$

Pour ce nombre d'éléments non nuls les complexités de l'algorithme 5.3 et la multiplication par une matrice creuse sont égales. Pour des valeurs de nz_i inférieures à cette borne la multiplication par une matrice creuse sera plus performante et pour des valeurs supérieures l'avantage sera à l'algorithme proposé[3]. Il faut rappeler que ces considérations sont applicables à la comparaison d'un seul terme produit tensoriel classique (sans éléments fonctionnels). Les cas pratiques, avec des descripteurs composés par une somme de produits tensoriels généralisés sont beaucoup plus complexes et seulement l'expérimentation numérique peut apporter des éléments d'information certains à propos des coûts d'exécution.

5.1.2 Résolution de Systèmes Triangulaires

Dans cette section nous faisons une parenthèse pour montrer que la résolution d'un système linéaire tringulaire (inférieur ou supérieur) peut se faire en utilisant un schéma algorithmique identique à celui de la multiplication vecteur-matrice lorsque la matrice est un produit tensoriel. L'algorithme 5.3 implémente la multiplication

$$v \times \otimes_{i=1}^{N} Q^{(i)}.$$

Un algorithme de même structure peut être employé pour déterminer la valeur du vecteur y solution du système

$$y \times \otimes_{i=1}^{N} Q^{(i)} = v,$$

si, et seulement si, les matrices $Q^{(i)}$ sont triangulaires. Prenons d'abord le cas où les matrices $Q^{(i)}$ sont triangulaires supérieures et appelons ces matrices $U^{(i)}$. La solution du système

$$y \times \otimes_{i=1}^{N} U^{(i)} = v$$

est équivalente à évaluer la multiplication

$$v \times \left(\otimes_{i=1}^{N} U^{(i)}\right)^{-1}.$$

Avec la propriété de compatibilité avec l'inversion de matrices (équation 2.9, page 21), nous pouvons re-écrire la matrice $\left(\otimes_{i=1}^{N} U^{(i)}\right)^{-1}$:

$$\otimes_{i=1}^{N} \left(U^{(i)}\right)^{-1}.$$

[3]Pour ces comparaisons, le coût de génération d'une matrice à partir d'une expression tensorielle n'est pas pris en compte.

Suivant la propriété de décomposition en facteurs normaux (équation 2.16, page 23), nous pouvons décomposer le produit tensoriel $\otimes_{i=1}^{N} \left(U^{(i)}\right)^{-1}$:

$$\begin{aligned}\otimes_{i=1}^{N} \left(U^{(i)}\right)^{-1} = &\left(U^{(1)}\right)^{-1} \otimes I_{nright_1} \\ &\times I_{nleft_2} \otimes \left(U^{(2)}\right)^{-1} \otimes I_{nright_2} \\ &\times \cdots \\ &\times I_{nleft_{N-1}} \otimes \left(U^{(N-1)}\right)^{-1} \otimes I_{nright_{N-1}} \\ &\times I_{nleft_N} \otimes \left(U^{(N)}\right)^{-1}\end{aligned}$$

De plus, la propriété de commutativité entre facteurs normaux (équation 2.19, page 26) nous autorise le choix de l'ordre suivant:

$$\begin{aligned}\otimes_{i=1}^{N} \left(U^{(i)}\right)^{-1} = &I_{nleft_N} \otimes \left(U^{(N)}\right)^{-1} \\ &\times I_{nleft_{N-1}} \otimes \left(U^{(N-1)}\right)^{-1} \otimes I_{nright_{N-1}} \\ &\times \cdots \\ &\times I_{nleft_2} \otimes \left(U^{(2)}\right)^{-1} \otimes I_{nright_2} \\ &\times \left(U^{(1)}\right)^{-1} \otimes I_{nright_1}\end{aligned}$$

L'évaluation de la multiplication

$$\begin{aligned}y \quad &\times I_{nleft_N} \otimes \left(U^{(N)}\right)^{-1} \\ &\times I_{nleft_{N-1}} \otimes \left(U^{(N-1)}\right)^{-1} \otimes I_{nright_{N-1}} \\ &\times \cdots \\ &\times I_{nleft_2} \otimes \left(U^{(2)}\right)^{-1} \otimes I_{nright_2} \\ &\times \left(U^{(1)}\right)^{-1} \otimes I_{nright_1}\end{aligned}$$

se fait avec l'algorithme décrit précédemment (section 5.1).

D'abord nous multiplions le vecteur y par le facteur normal $I_{nleft_N} \otimes \left(U^{(N)}\right)^{-1}$. Ce facteur normal étant une matrice composée de blocs diagonaux qui sont des matrices triangulaires supérieures, des tranches du vecteur y sont multipliés par $\left(U^{(N)}\right)^{-1}$ (figure 5.4). La multiplication d'une tranche du vecteur y par l'inverse d'une matrice triangulaire supérieure équivaut à une solution en remontée ((*backward substitution*) [43]. Ceci nous permet d'effectuer la multiplication de ce facteur normal grace à la résolution de $\bar{n}_N$ systèmes triangulaires.

Ensuite, comme dans le cas précédent (section 5.1), les autres facteurs normaux sont permutés de façon à avoir le même format que le facteur normal $I_{nleft_N} \otimes \left(U^{(N)}\right)^{-1}$. Notons que tous les facteurs normaux permutés deviennent des matrices triangulaires supérieures.

L'algorithme 5.4 représente de façon analogue à l'algorithme 5.3 la multiplication d'un vecteur y par l'inverse d'une matrice triangulaire supérieure décrite comme le produit tensoriel de N matrices triangulaires supérieures[4].

[4]Grace à à la propriété de commutativité entre facteurs normaux (équation 2.19, page 26), l'ordre dans lequel les facteurs normaux sont résolus n'est pas important. Nous adoptons l'ordre du dernier facteur normal ($I_{nleft_N} \otimes \left(U^{(N)}\right)^{-1}$) jusqu'au premier ($\left(U^{(1)}\right)^{-1} \otimes I_{nright_1}$) de façon arbitraire.

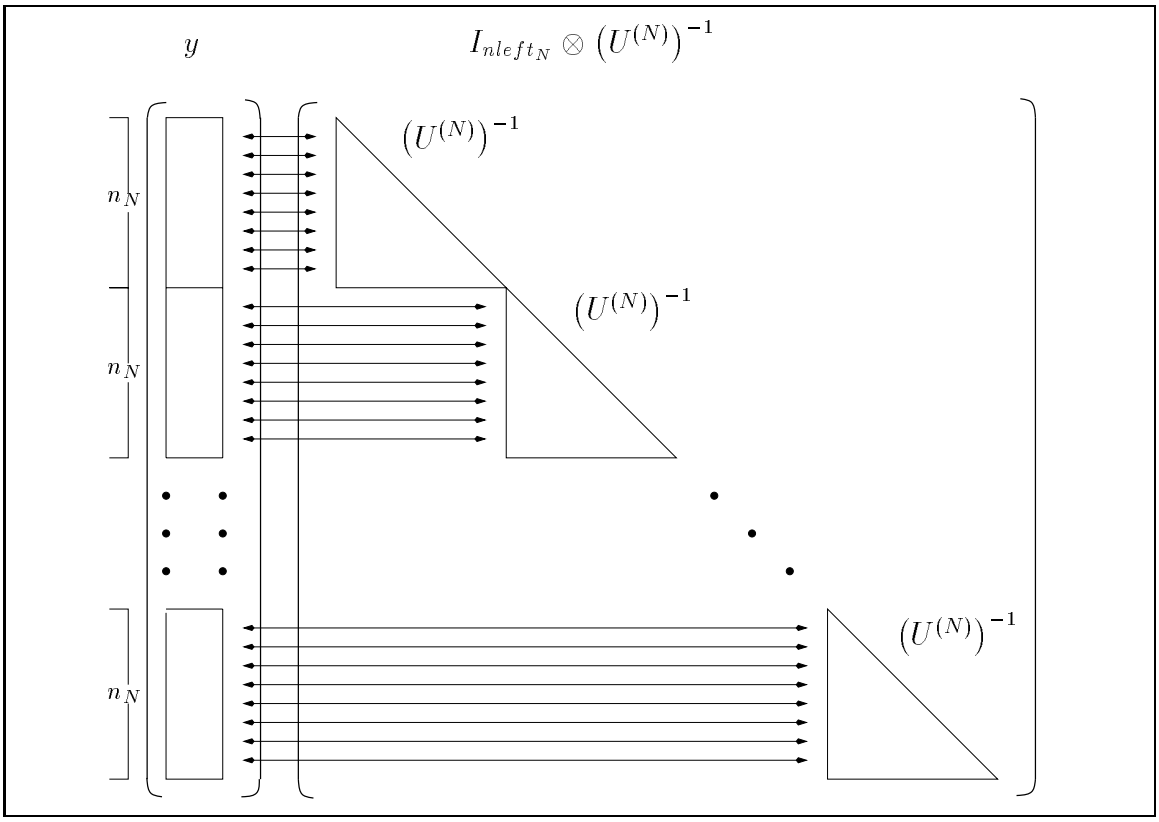


FIG. 5.4: Matrice $I_{nleft_N} \otimes \left(U^{(N)}\right)^{-1}$

Il est facile de vérifier que le procédé analogue est valide pour des produits tensoriels de matrices triangulaires inférieures. L'une des applications de cet algorithme est discutée dans la section 6.2.2.

5.2 Traitement des Dépendances Fonctionnelles

La multiplication d'un produit tensoriel généralisé peut se faire de deux façons distinctes selon le type de *dépendances fonctionnelles* existant entre les matrices. Définissons un *graphe de dépendances fonctionnelles* entre les matrices d'un produit tensoriel généralisé[5] $\otimes_{i=1}^{N} Q^{(i)}(\dots)$. Ce graphe est orienté et chaque matrice $Q^{(i)}(\dots)$ est représentée par un sommet. Chaque arc d'un sommet i vers un sommet j représente la dépendance fonctionnelle de la matrice $Q^{(i)}(\dots)$ de l'état de l'automate[6] $\mathcal{A}^{(j)}$. La figure 5.5 montre quelques exemples de graphes de dépendances fonctionnelles.

D'après les propriétés de décomposition en facteurs normaux des produits tensoriels généralisés (équations 2.35 et 2.36), il est toujours possible d'obtenir un ordre σ pour

[5]La notation $Q^{(i)}(\dots)$ est utilisée pour décrire de manière générique des matrices comportant différentes dépendances fonctionnelles

[6]Les matrices $Q^{(i)}$ d'un terme produit tensoriel dans un descripteur représentent chacune un automate $\mathcal{A}^{(i)}$ du modèle RAS. La définition des dépendances fonctionnelles, ainsi que des éléments fonctionnels est faite dans la section 3.1.1 dans la page 51.

Algorithme 5.4

```
1  for i = 1, 2, ..., N
2  do base = 0;
3      for k = 1, 2, ..., nleft_i
4      do for j = 1, 2, ..., nright_i
5          do index = base + j;
6              for l = 1, 2, ..., n_i
7              do z_in[l] = y[index];
8                  index = index + nright_i;
9              end do
10             solve      z_out = z_in × (U^(i))^-1;
11             index = base + j;
12             for l = 1, 2, ..., n_i
13             do y[index] = z_out[l];
14                 index = index + nright_i;
15             end do
16         end do
17         base = base + (nright_i × n_i);
18     end do
19 end do
```

Algorithme 5.4: Multiplication $y \times \left(\otimes_{i=1}^{N} U^{(i)}\right)^{-1}$

multiplier les facteurs normaux d'un terme $\otimes_{i=1}^{N} Q^{(i)}(\dots)$ si et seulement s'il n'y a pas de cycles dans son graphe de dépendances fonctionnelles. L'existence de cycles empêche l'application directe des propriétés de décomposition en facteurs normaux.

Dans cette section la multiplication des produits tensoriels généralisés sans cycle de dépendances fonctionnelles est exposée. Ensuite le traitement nécessaire pour éliminer ces cycles, et donc se ramener au cas précédent, est présenté.

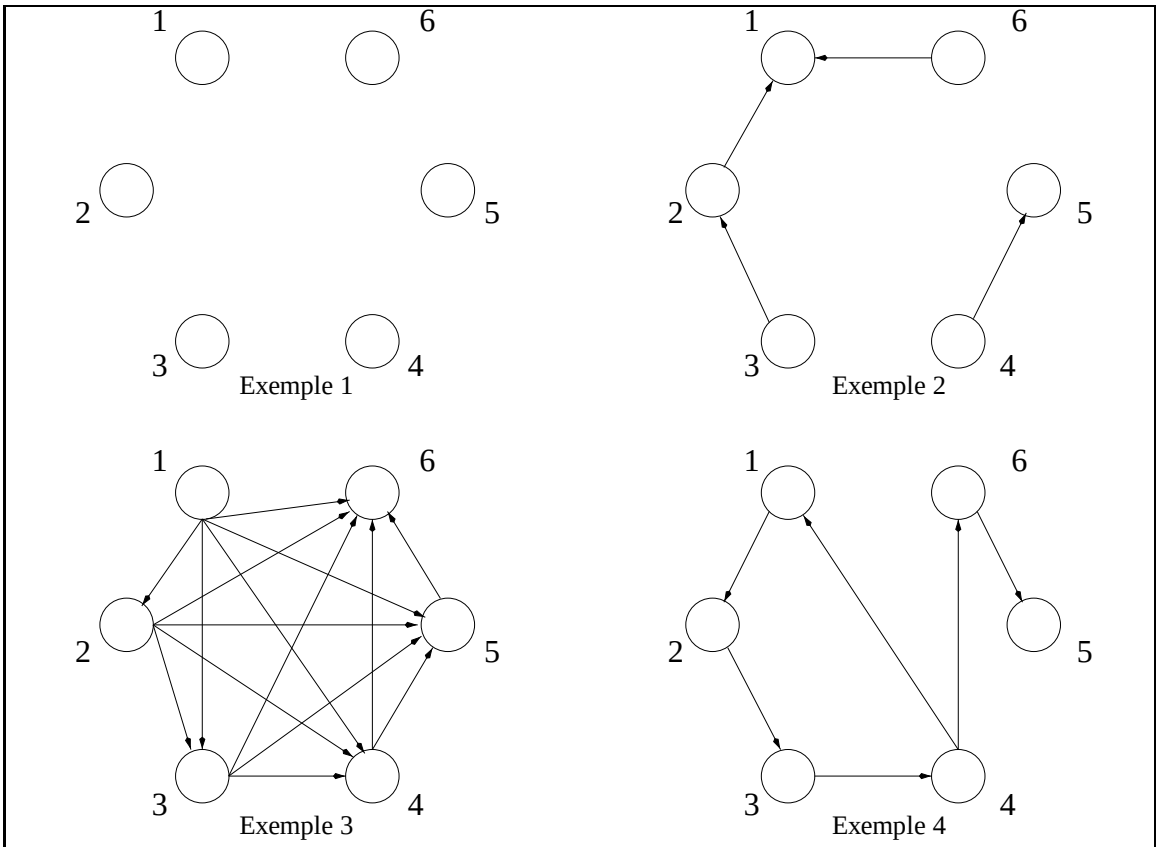
5.2.1 Cas sans Cycle

La multiplication d'un vecteur v par un produit tensoriel $\bigotimes_{g\,i=1}^{N} Q^{(i)}(\mathcal{A}^{(i+1)}, \dots, \mathcal{A}^{(N)})$ (sans cycle) est faite de façon similaire au cas sans éléments fonctionnels. Deux modifications doivent êtres faites à la multiplication implémentée par l'algorithme 5.3:

- calculer un ordre σ dans lequel les facteurs normaux doivent être multipliés;
- évaluer les éléments fonctionnels des matrices avant chacune de leur multiplication.

Établissement de l'ordre des Facteurs Normaux

Rappelons que selon les propriétés de décomposition en facteurs normaux des produits tensoriels généralisés (équations 2.35 et 2.36), le facteur normal d'une matrice $Q^{(i)}$ doit



Exemples

1. $Q^{(1)} \underset{g}{\otimes} Q^{(2)} \underset{g}{\otimes} Q^{(3)} \underset{g}{\otimes} Q^{(4)} \underset{g}{\otimes} Q^{(5)} \underset{g}{\otimes} Q^{(6)}$
2. $Q^{(1)} \underset{g}{\otimes} Q^{(2)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} Q^{(3)}(\mathcal{A}^{(2)}) \underset{g}{\otimes} Q^{(4)}(\mathcal{A}^{(5)}) \underset{g}{\otimes} Q^{(5)} \underset{g}{\otimes} Q^{(6)}(\mathcal{A}^{(1)})$
3. $\underset{g}{\bigotimes_{i=1}^{6}} Q^{(i)}(\mathcal{A}^{(i+1)}, \ldots, \mathcal{A}^{(6)})$
4. $Q^{(1)}(\mathcal{A}^{(2)}) \underset{g}{\otimes} Q^{(2)}(\mathcal{A}^{(3)}) \underset{g}{\otimes} Q^{(3)}(\mathcal{A}^{(4)}) \underset{g}{\otimes} Q^{(4)}(\mathcal{A}^{(1)}, \mathcal{A}^{(6)}) \underset{g}{\otimes} Q^{(5)} \underset{g}{\otimes} Q^{(6)}(\mathcal{A}^{(5)})$

FIG. 5.5: Exemples de Graphes de Dépendances Fonctionnelles

toujours précéder les facteurs normaux des matrices $Q^{(j)}(\ldots, \mathcal{A}^{(i)}, \ldots)$, *i.e.*, les facteurs normaux des matrices qui dépendent de l'automate $\mathcal{A}^{(i)}$. Ceci définit un ordre partiel entre les facteurs normaux.

L'établissement de l'ordre des facteurs normaux est fait par un parcours du graphe de dépendances fonctionnelles des matrices $Q^{(i)}(\ldots)$. Ce parcours permet d'extraire un ordre total compatible avec l'ordre partiel défini par ce graphe par un algorithme de tri topologique [11]:

- le sommet sans arc entrant (représentant une matrice qui n'est paramètre d'aucune autre matrice) est classée en premier;
- le sommet classé en premier (et ses arcs sortants) sont enlevés du graphe;
- le sommet sans arc entrant (représentant une matrice que n'est paramètre d'aucune

autre matrice sauf celle déjà classée) est classé en deuxième;

- le sommet classé en deuxième (et ses arcs sortants) sont enlevés du graphe;
- l'algorithme se poursuit en classant un sommet sans arc entrant et en l'enlevant du graphe jusqu'à épuisement des sommets.

Par exemple, les produits

$$v \times \bigotimes_{i=1}^{N}{}_g \, Q^{(i)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(i-1)}) \quad \text{(appelé cas 1) et}$$

$$v \times \bigotimes_{i=1}^{N}{}_g \, Q^{(i)}(\mathcal{A}^{(i+1)}, \ldots, \mathcal{A}^{(N)}) \quad \text{(appelé cas 2)}$$

sont des cas où il faut appliquer directement une des propriétés de décomposition en facteurs normaux (soit 2.36 pour le cas 1, soit 2.35 pour le cas 2). Ces application donnent respectivement:

$$v \times \prod_{i=N}^{1} \left[I_{nleft_i} \underset{g}{\otimes} Q^{(i)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(i-1)}) \underset{g}{\otimes} I_{nright_i} \right] \quad \text{(pour le cas 1) et}$$

$$v \times \prod_{i=1}^{N} \left[I_{nleft_i} \underset{g}{\otimes} Q^{(i)}(\mathcal{A}^{(i+1)}, \ldots, \mathcal{A}^{(N)}) \underset{g}{\otimes} I_{nright_i} \right] \quad \text{(pour le cas 2).}$$

Notons que, entre ces deux cas, la seule différence est l'ordre de multiplication des facteurs normaux[7]. L'impact de ce changement d'ordre sur le schéma d'algorithme présenté (algorithme 5.3) est le suivant:

- Pour le cas 1, la première ligne de l'algorithme 5.3 est modifiée comme indiqué dans l'algorithme 5.5;
- Le cas 2, au contraire, doit être traité selon l'algorithme 5.6.

Il est utile de remarquer que les ordres exprimés dans les algorithmes 5.5 et 5.6 sont obligatoires, *i.e.*, la seule décomposition en facteurs normaux valide est celle utilisée (de N à 1 pour le cas 1 et de 1 à N pour le cas 2). Ceci n'est pas le cas de la multiplication des produits tensoriel sans fonctions (algorithme 5.3) où n'importe quel ordre peut être employé.

Le cas général reste l'application alternée des propriétés 2.35 et 2.36 qui permet l'établissement d'un ordre pour la multiplication des facteur normaux. Cette ordre va être noté par une permutation σ sur l'intervalle $[1..N]$. Pour cela rappelons les notations suivantes (déjà définies au chapitre 2):

[7] Il faut rappeler que la multiplication de facteurs normaux avec éléments fonctionnels n'est pas commutative

Algorithme 5.5

```
1  for i = N, N − 1, . . . , 1
2  do ···
⋮
```

Algorithme 5.5: Multiplication $v \times \bigotimes_{g\, i=1}^{N} Q^{(i)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(i-1)})$

Algorithme 5.6

```
1  for i = 1, 2, . . . , N
2  do ···
⋮
```

Algorithme 5.6: Multiplication $v \times \bigotimes_{g\, i=1}^{N} Q^{(i)}(\mathcal{A}^{(i+1)}, \ldots, \mathcal{A}^{(N)})$

Rappelons

σ une permutation nommée σ sur l'intervalle $[1..N]$, une permutation σ établie un nouveau ordre pour une suite de N matrices;

$\sigma(i)$ le rang de la matrice $Q^{(i)}$ dans l'ordre représenté par la permutation σ;

σ_k l'indice de la matrice placée au rang k de l'ordre représenté par la permutation σ (si $\sigma_k = i$, $\sigma(i) = k$);

Le produit $v \times \bigotimes_{g\, i=1}^{N} Q^{(i)}(\ldots)$ doit être traité par

$$v \times \prod_{i=1}^{N} \left[I_{nleft_{\sigma_i}} \underset{g}{\otimes} Q^{(\sigma_i)}(\ldots) \underset{g}{\otimes} I_{nright_{\sigma_i}} \right]$$

Le changement dans l'algorithme 5.3 est indiqué dans l'algorithme 5.7. Notons que ce changement demande le calcul d'un ordre de traitement (permutation σ) qui n'est pas inclus dans l'algorithme de multiplication du facteur normal.

Algorithme 5.7

```
1  for i = σ1, σ2, . . . , σN
2  do ···
⋮
```

Algorithme 5.7: Multiplication $v \times \bigotimes_{g\, i=1}^{N} Q^{(i)}(\ldots)$

L'ordre dans lequel les facteurs normaux doivent être multipliés sera décrite par la permutation σ et sera appelé *ordre de décomposition*. Or cet ordre représentera l'ordre dans lequel le produit tensoriel sera décomposé en facteurs normaux selon les propriétés 2.35

et 2.36. Nous voulons attirer l'attention du lecteur sur cette dénomination, car d'autres ordres seront présentés dans la prochaine section à des fins d'optimisations.

Pour certains produits tensoriels l'*ordre de décomposition* peut ne pas être unique[8]. Par exemple, deux matrices dans la série peuvent être constantes. Dans ce cas, n'importe laquelle des matrices peut être traitée avant l'autre. Ceci est dû au fait que le produit de facteurs normaux de deux matrices constantes sont commutatifs (propriété décrite dans la section 2.1.3). La règle générale est que deux (ou plusieurs) matrices n'ayant pas de dépendances fonctionnelles directes ou indirectes entre elles peuvent changer librement de position. Cette règle généralise l'absence d'un ordre précis pour la multiplication des facteurs normaux d'un produit tensoriel classique (cas sans fonctions).

Évaluation des Éléments Fonctionnels

Après le calcul de l'*ordre de décomposition*, la deuxième préoccupation pour la multiplication des produits tensoriels généralisés est l'évaluation des éléments fonctionnels de chaque matrice $Q^{(i)}(\dots)$ avant leur multiplication par une tranche z_{in} du vecteur v (correspondant à la ligne 10 de l'algorithme 5.3).

À chaque exécution de la multiplication d'une tranche z_{in} du vecteur v par la matrice $Q^{(i)}(\dots)$, cette matrice doit être évaluée pour les états des automates arguments de $Q^{(i)}$. Les indices k et j représentent, respectivement, les sous vecteurs des états des automates à gauche et à droite[9] de l'automate $\mathcal{A}^{(i)}$.

Pour l'évaluation des éléments fonctionnels, il est nécessaire d'inclure le calcul des états locaux des automates arguments de $\mathcal{A}^{(i)}$. Comme ces arguments sont un sous ensemble à priori arbitraire du RAS, ceci amène à calculer les états locaux de tous automates autres que $\mathcal{A}^{(i)}$. Ensuite il faut faire l'évaluation de $Q^{(i)}(\dots)$ selon ces états locaux.

Examinons d'abord le calcul des états locaux des automates à gauche de l'automate $\mathcal{A}^{(i)}$. L'indice k de l'algorithme est le rang dans l'espace d'état $\prod_{j=1}^{i-1} S^{(j)}$, *i.e.*, des sous vecteurs d'états locaux du type $(x^{(1)}, x^{(2)}, \dots, x^{(i-1)})$. Il est possible de regarder un sous vecteur d'états locaux comme un nombre en base variable, *i.e.*, un nombre pour lequel chacun des chiffres a sa propre base. L'état local de l'automate $\mathcal{A}^{(j)}$ ($x^{(j)}$) représentera un *chiffre* qui peut varier dans l'intervalle $[1..n_k]$. Ceci permet d'obtenir le rang k dans l'espace d'état $\prod_{j=1}^{i-1} S^{(j)}$ d'un sous vecteur $(x^{(1)}, x^{(2)}, \dots, x^{(i-1)})$ par la formule:

$$k = \sum_{j=1}^{i-1} \left(x^{(j)} \times \prod_{l=1}^{j-1} (n_l) \right)$$

De façon analogue, on peut obtenir le sous vecteur $(x^{(1)}, x^{(2)}, \dots, x^{(i-1)})$ à partir d'un indice k donné par une suite de divisions entières.

[8] La discussion sur les *ordres de décomposition* en facteurs normaux possibles pour un produit tensoriel généralisé est présentée dans la section 2.2.6, page 42.

[9] Nous allons appeler *à gauche* d'une matrice d'indice i, toutes matrices d'un même terme qui ont des indices l inférieurs à i ($l \in [1..i-1]$). De façon analogue, nous allons appeler *à droite* d'une matrice d'indice i, toutes matrices d'un même terme qui ont des indices l supérieurs à i ($l \in [i+1..N]$).

Un algorithme qui implante ces décisions est très coûteux (autant de divisions entières que le nombre d'automates à gauche de $\mathcal{A}^{(i)}$). Une solution moins coûteuse peut être employée pour cet algorithme. La première remarque utile est que toutes les combinaisons d'états locaux vont être nécessaires. De plus, les combinaisons seront prises exactement dans l'ordre lexicographique (la valeur de l'indice k varie de 1 jusqu'à $nleft_i$). La deuxième remarque intéressante est qu'il est peu coûteux de passer d'un sous vecteur d'états locaux au sous vecteur suivant selon un ordre lexicographique. Ceci peut être fait par un algorithme simple d'incrémentation de +1 d'un nombre en base variable.

Prenons par exemple, les états locaux de trois automates ($\mathcal{A}^{(1)}$, $\mathcal{A}^{(2)}$ et $\mathcal{A}^{(3)}$) avec tailles $n_1 = 2$, $n_2 = 4$ et $n_3 = 3$. Les sous vecteurs d'états locaux ordonnés selon l'ordre lexicographique sont:

rang (k)	$x^{(1)}$ (k_1)	$x^{(2)}$ (k_2)	$x^{(3)}$ (k_3)	rang (k)	$x^{(1)}$ (k_1)	$x^{(2)}$ (k_2)	$x^{(3)}$ (k_3)	rang (k)	$x^{(1)}$ (k_1)	$x^{(2)}$ (k_2)	$x^{(3)}$ (k_3)	rang (k)	$x^{(1)}$ (k_1)	$x^{(2)}$ (k_2)	$x^{(3)}$ (k_3)
1	1	1	1	7	1	3	1	13	2	1	1	19	2	3	1
2	1	1	2	8	1	3	2	14	2	1	2	20	2	3	2
3	1	1	3	9	1	3	3	15	2	1	3	21	2	3	3
4	1	2	1	10	1	4	1	16	2	2	1	22	2	4	1
5	1	2	2	11	1	4	2	17	2	2	2	23	2	4	2
6	1	2	3	12	1	4	3	18	2	2	3	24	2	4	3

Le changement de $\{1, 2, 2\}$ (de rang 5), vers le prochain sous vecteur dans l'ordre lexicographique (de rang 6) nécessite l'incrément de l'état du dernier état local (de 2 vers 3). Le changement de cette nouvelle combinaison ($\{1, 2, 3\}$ - rang 6) vers la prochaine (de rang 7) ne correspond pas à une incrémentation du dernier état local (il est déjà à sa valeur maximale $n_3 = 3$), donc l'avant dernier état local doit être incrémenté, ce qui donne $\{1, 3, 1\}$.

De façon générale, pour parcourir les sous vecteurs avec les états locaux des automates à gauche de $Q^{(i)}$ on initialise:

$$k_1 = 1 \quad k_2 = 1 \quad \cdots \quad k_{i-2} = 1 \quad k_{i-1} = 1$$

À chaque incrément de k, correspond une addition de +1 sur le nombre en base variable, soit un incrément de l'état local k_{i-1}, avec une propagation de retenue si nécessaire. La suite des valeurs obtenues est:

$$\begin{array}{llllllllllll}
k_1 = 1 & k_2 = 1 & \cdots & k_{i-2} = 1 & k_{i-1} = 1 \\
k_1 = 1 & k_2 = 1 & \cdots & k_{i-2} = 1 & k_{i-1} = 2 \\
\vdots & \vdots & & \vdots & \vdots \\
k_1 = 1 & k_2 = 1 & \cdots & k_{i-2} = 1 & k_{i-1} = n_{i-1} \\
k_1 = 1 & k_2 = 1 & \cdots & k_{i-2} = 2 & k_{i-1} = 1 \\
\vdots & \vdots & & \vdots & \vdots \\
k_1 = n_1 & k_2 = n_2 & \cdots & k_{i-2} = n_{i-2} & k_{i-1} = n_{i-1}
\end{array}$$

Ce même raisonnement peut s'appliquer au calcul des états correspondant aux automates à droite de $\mathcal{A}^{(i)}$ selon la valeur de j.

L'algorithme 5.8 implémente les modifications nécessaires au traitement de termes avec éléments fonctionnels par rapport à l'algorithme 5.3. Les six opérations qui ont été ajoutées sont:

- ligne 1: le parcours des facteurs normaux dans l'ordre de décomposition σ;
- ligne 3: l'initialisation du sous vecteur avec les états locaux des automates à gauche de la matrice $Q^{(i)}(\dots)$;
- ligne 6: l'initialisation du sous vecteur avec les états locaux des automates à droite de la matrice $Q^{(i)}(\dots)$;
- ligne 13: l'évaluation de la matrice $Q^{(i)}(\dots)$ avec les arguments *calculés* (états locaux des automates);
- ligne 19: l'incrément du sous vecteur avec les états locaux des automates à droite de la matrice $Q^{(i)}(\dots)$;
- ligne 22: l'incrément du sous vecteur avec les états locaux des automates à gauche de la matrice $Q^{(i)}(\dots)$.

On peut remarquer que les sous vecteurs d'états locaux des automates à gauche et à droite de la matrice $Q^{(i)}(\dots)$ sont redondants par rapport aux indices k et j, respectivement. Cette redondance a pour but d'éviter le calcul des sous vecteurs à chaque incrément de k et j, qui est trop coûteux. De ce point de vue, les initialisations (lignes 3 et 6 de l'algorithme 5.8) et les incréments (lignes 22 et 19) sont analogues aux initialisations et incréments sur k et j exécutés par les contrôles de boucles dans les lignes 4 et 5, respectivement.

L'algorithme 5.8 n'apporte pas de changement dans la taille des boucles par rapport à l'algorithme 5.3. La multiplication d'une tranche du vecteur v par les matrices $Q^{(i)}(\dots)$ reste aussi identique. Ceci nous permet de dire que la complexité de l'algorithme 5.8 reste du même ordre que celle de l'algorithme 5.3. Les seuls coûts additionnels correspondent aux six opérations incluses. Le parcours des facteurs normaux dans l'ordre exprimé par σ est très peu important car il ne fait qu'ajouter une indirection aux accès sur i. L'incrément du sous vecteur avec les états locaux des automates à droite de la matrice $Q^{(i)}(\dots)$ et l'évaluation de la matrice $Q^{(i)}$ représentent un coût important, surtout du fait que ces opérations doivent être faites dans la boucle la plus interne (boucle de 1 jusqu'à $nright_i$). Dans la section des optimisations algorithmiques (section 5.3), ces coûts sont étudiés et des solutions pour réduire leur impact sont proposées.

5.2.2 Traitement des Cycles

Le traitement d'un terme où le graphe de dépendances fonctionnelles a des cycles est fait par décomposition du terme en plusieurs termes libres de cette restriction. Ensuite, pour chacun de ces termes l'algorithme 5.8 de la section précédente peut être employé.

Idée de Base

Un des cas le plus évident où il y a des cycles est le produit $A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A})$. Pour ce cas il n'existe pas de propriété permettant la décomposition directe en facteurs normaux.

Algorithme 5.8

```
1  for i = σ(1), σ(2), ... , σ(N)
2  do  base = 0;
3      initialize   k_1 = 1, k_2 = 1, ... , k_{i-1} = 1;
4      for k = 1, 2, ... , nleft_i
5      do  for j = 1, 2, ... , nright_i
6          do  initialize   j_{i+1} = 1, j_{i+2} = 1, ... , j_N = 1;
7              index = base + j;
8              for l = 1, 2, ... , n_i
9              do  z_in[l] = v[index];
10                 index = index + nright_i;
11             end do
12             evaluate   Q^(i)(a^(1)_{k_1}, ... , a^(N)_{j_N});
13             multiply   z_out = z_in × Q^(i);
14             index = base + j;
15             for l = 1, 2, ... , n_i
16             do  v[index] = z_out[l];
17                 index = index + nright_i;
18             end do
19             next   j_{i+1}, j_{i+2}, ... , j_N;
20         end do
21         base = base + ( nright_i × n_i);
22         next   k_1, k_2, ... , k_{i-1};
23     end do
24 end do
```

Algorithme 5.8: Multiplication $v \times \bigotimes_{g\, i=1}^{N} Q^{(i)}(\dots)$

Pourtant il est possible d'appliquer la propriété de décomposition en produits tensoriels classiques (équation 2.31 dans la page 30) de la façon suivante:

$$\begin{aligned} A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A}) = \sum_{k=1}^{n_A} \ell_k(A(\mathcal{B})) \underset{g}{\otimes} B(a_k) = & \ell_1(A(\mathcal{B})) \otimes B(a_1) \\ & + \ell_2(A(\mathcal{B})) \otimes B(a_2) \\ & \vdots \\ & + \ell_{n_A}(A(\mathcal{B})) \otimes B(a_{n_A}) \end{aligned}$$

La multiplication d'un vecteur v par un terme $\sum_{k=1}^{n_A} \ell_k(A(\mathcal{B})) \otimes B(a_k)$ devient la somme des multiplications par chacune des décompositions, *i.e.*:

$$v \times \sum_{k=1}^{n_A} \ell_k(A(\mathcal{B})) \otimes B(a_k) = \sum_{k=1}^{n_A} \left(v \times \ell_k(A(\mathcal{B})) \otimes B(a_k)\right)$$

Le terme $v \times \ell_k(A(\mathcal{B})) \otimes B(a_k)$ devient un produit tensoriel valide pour l'application de l'algorithme 5.8, car la matrice $B(a_k)$ n'est plus dépendante de l'état de l'automate $\mathcal{A}$. L'application de la décomposition en facteurs normaux de ce terme pour l'algorithme 5.8 est:

$$v \ \times \ \ell_k(A(\mathcal{B})) \underset{g}{\otimes} I_{n_B} \ \times \ I_{n_A} \underset{g}{\otimes} B(a_k)$$

Notons que la matrice $B(\mathcal{A})$ aurait aussi pu être choisie pour être décomposée en lignes avec un résultat également satisfaisant:

$$\begin{aligned} A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A}) = \sum_{k=1}^{n_B} A(b_k) \underset{g}{\otimes} \ell_k(B(\mathcal{A})) = & A(b_1) \underset{g}{\otimes} \ell_1(B(\mathcal{A})) \\ & + A(b_2) \underset{g}{\otimes} \ell_2(B(\mathcal{A})) \\ & \vdots \\ & + A(b_{n_B}) \underset{g}{\otimes} \ell_{n_B}(B(\mathcal{A})) \end{aligned}$$

Généralisation pour les Suites finies de Matrices

Ce même raisonnement peut être appliqué pour des cas plus complexes où plusieurs matrices doivent être *décomposées* en matrices lignes. Examinons la multiplication d'un vecteur v par un terme $\bigotimes_{g\,i=1}^{N} Q^{(i)}(\dots)$ avec des cycles de dépendances fonctionnelles. Supposons, sans perte de généralité, que l'ensemble des T premières matrices soit choisi[10]. Les matrices de cet ensemble sont décomposées en lignes. La décomposition se fait de la façon suivante:

$$\begin{aligned} \bigotimes_{i=1}^{N}{}_g \, Q^{(i)}(\dots) &= \bigotimes_{i=1}^{T}{}_g \, Q^{(i)}(\dots) \bigotimes_{i=T+1}^{N}{}_g \, Q^{(i)}(\dots) \\ &= \bigotimes_{i=1}^{T}{}_g \left(\sum_{k_i=1}^{n_i} \ell_k(Q^{(i)}(\dots)) \right) \bigotimes_{i=T+1}^{N}{}_g \, Q^{(i)}(\dots) \end{aligned}$$

Par la propriété de distributivité avec la somme (équation 2.23, page 29):

$$\bigotimes_{i=1}^{T}{}_g \left(\sum_{k_i=1}^{n_i} \ell_k(Q^{(i)}(\dots)) \right) \bigotimes_{i=T+1}^{N}{}_g \, Q^{(i)}(\dots)$$

$$\sum_{k_1=1}^{n_1} \cdots \sum_{k_T=1}^{n_T} \bigotimes_{i=1}^{T}{}_g \, \ell_{k_i} \left(Q^{(i)}(a_{k_1}^{(1)}, \dots, a_{k_T}^{(T)}, \dots) \right) \bigotimes_{i=T+1}^{N}{}_g \, Q^{(i)}(a_{k_1}^{(1)}, \dots, a_{k_T}^{(T)}, \dots)$$

Chacun des termes de la somme donne lieu à un graphe de dépendances fonctionnelles sans cycle.

[10] Les matrices à décomposer peuvent être choisies à n'importe quelle position de la suite. Pour un confort de description on va supposer que ces matrices sont les T premières de la suite. Le paragraphe suivant va expliciter le choix des matrices réellement utilisé.

Choix des Matrices à Décomposer

Le problème de fond reste le choix des matrices à décomposer. Une méthode pour résoudre ce problème est de calculer le *transversal de cycle* du graphe de dépendances. Un transversal de cycle est un ensemble de sommets du graphe de dépendances tel que, en enlevant les arcs incidents à ces sommets, le graphe n'ait plus de cycle[11]. Enlever les arcs entrant d'un sommet revient à supprimer les paramètres modélisés par cet arc. Cette suppression se traduit dans la formule tensorielle par la décomposition des matrices en ligne ($A = \sum_{k=1}^{n_a} \ell_k(A)$). En effet, ne conserver qu'une ligne k d'une matrice A revient à imposer que l'automate soit dans l'état correspondant à cette ligne (a_k). Le paramètre devient constant et égal à a_k et n'est donc plus identifié comme un paramètre variable (une dépendance fonctionnelle).

La détermination des matrices d'un transversal est faite d'abord par identification des matrices appartenant à un cycle. La technique utilisée (algorithme de tri topologique) pour déterminer un ordre de décomposition σ sur le graphe de dépendances sert aussi à identifier les matrices appartenant à un cycle. En essayant de placer les matrices dans cet ordre, on se rend compte que les matrices appartenant à un cycle ne peuvent jamais être rangées. Une fois établi l'ensemble de matrices appartenant à un ou plusieurs cycles, des heuristiques peuvent être employées pour déterminer un transversal des cycles.

Pour un cycle il est toujours possible de trouver plusieurs transversals. Par exemple, dans le produit $A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A})$ trois transversals peuvent être obtenus:

– la matrice $A(\mathcal{B})$;

– la matrice $B(\mathcal{A})$;

– les deux matrices.

Évidement, le choix du transversal doit être fait en gardant à l'esprit que la décomposition des matrices va impliquer dans la génération d'autant de nouveaux termes tensoriels que le produit des tailles de chacune des matrices du transversal. Pour cet exemple, il est évident qu'il n'est pas nécessaire de décomposer les deux matrices, car ceci aurait pour effet la génération de $n_A \times n_B$ nouveaux termes. Donc, le choix du transversal doit se porter en faveur de la matrice avec la plus petite dimension.

Il faut remarquer que l'idée intuitive de chercher le transversal de cycles avec le plus petit nombre d'automates n'est pas forcément le meilleur choix. Le nombre de nouveaux termes générés est égal au produit des tailles de chaque matrice à décomposer. Le transversal à choisir doit minimiser ce produit.

Prenons un exemple où quatre matrices (A, B, C et D) font partie des cycles. Les interdépendances entre les matrices peuvent amener à des transversals parmi les possibilités suivantes:

A, B, C, D, AB, AC, AD, BC, BD, CD, ABC, ABD, BCD, $ABCD$.

Bien sûr, un transversal avec une seule matrice, si possible avec la matrice de plus petite dimension, sera toujours le meilleur choix. Il est aussi évident que le transversal

[11] Les définitions traditionnelles des transversals et de transversals de cycles [11, 86] ont un point de vue plus général. Pourtant pour simplifier la compréhension des techniques employées dans le cadre de cette thèse la définition fournie est suffisante.

contenant toutes les matrices ne sera jamais un bon choix, car pour *casser* un cycle il suffit d'enlever tous les sommets sauf un. Ceci nous permet aussi d'affirmer que parmi les possibilités raisonnables, la combinaison qui retient toutes les matrices sauf celle de plus grande dimension est la borne supérieure de notre liste de possibilités. Dans le cas des quatre matrices, supposons encore que les dimensions sont définies:

$$n_A < n_B < n_C < n_D$$

Il est possible de savoir que le meilleur choix, si valide, sera le transversal égal à A. Également les trois possibilités ABD, BCD et $ABCD$ peuvent être oubliées car elles sont sans doute plus mauvaises que l'option sûrement valide ABC. Entre les autres possibilités aucune affirmation à priori ne peut être faite. Par exemple, il est impossible de savoir à priori si le choix AB sera plus avantageux que le choix D.

Étant donné les coûts induits par la recherche d'un transversal optimal, l'utilisation d'heuristiques de la théorie des graphes n'est pas intéressante. La solution exhaustive qui consiste à établir un ordre parmi tous les transversals possibles est la seule façon sûre de trouver le meilleur transversal. Cette solution, aussi naïve qu'elle puisse paraître, est valide dans notre contexte car le nombre d'automates est en général faible (inférieur à 20). Elle présente encore plus d'avantages si on considère des optimisations algorithmiques simples comme ne générer que les termes où les lignes des matrices décomposées ne sont pas nulles. Cette optimisation implique de considérer le nombre de termes nouveaux, non plus comme le produit des dimensions de chaque matrice du transversal, mais comme le produit du nombre de lignes ayant au moins un élément non nul dans ce même ensemble.

Complexité

La complexité de la multiplication d'un vecteur v par un produit tensoriel $\bigotimes_{g\,i=1}^{N} Q^{(i)}(\dots)$ reste essentiellement une fonction du nombre de termes sans cycle qui ont dû être générés. La complexité du produit de chaque terme sans cycle reste toujours exprimée par l'équation 5.2. Cette valeur est multiplié par le nombre de nouveaux termes générés, dont une borne supérieure est:

$$\prod_{k \in \mathcal{T}} n_k \tag{5.3}$$

avec $\mathcal{T}$ l'ensemble des indices des matrices choisies pour le transversal

Notons cependant que le nombre d'éléments non nuls dans les matrices lignes des nouveaux termes est généralement très petit par rapport au nombre total des éléments non nuls des matrices originelles (avant décomposition).

5.3 Optimisations Algorithmiques

Les algorithmes présentés effectuent la multiplication d'un vecteur par un terme produit tensoriel. Cependant le descripteur d'un RAS est une somme de produits tensoriels.

Cette section propose des améliorations numériques de certains cas pratiques et propose trois optimisations dans le traitement d'un produit vecteur descripteur.

5.3.1 Pré-calcul de la Diagonale du Descripteur

La première optimisation proposée est basée sur l'observation du nombre de multiplications dans un produit vecteur descripteur par rapport au nombre de multiplications dans un produit vecteur matrice creuse équivalent.

Prenons un descripteur de RAS où il n'existe pas d'événements synchronisants. Ce descripteur sera composé d'une seule somme tensorielle, c'est à dire, une somme de facteurs normaux. Un facteur normal $I_{nleft_i} \underset{g}{\otimes} Q_l^{(i)}(\dots) \underset{g}{\otimes} I_{nright_i}$ nécessite un nombre de multiplications égal à:

$$\bar{n}_i \times nz_i$$

Le nombre total de multiplications du descripteur est égal à:

$$\sum_{i=1}^{N} (\bar{n}_i \times nz_i)$$

Le nombre de multiplications à effectuer pour le produit d'un vecteur par une matrice creuse est égal au nombre d'éléments non nuls de cette matrice. En observant une matrice générée par une somme tensorielle $\bigoplus_{g\,i=1}^{N} Q_l^{(i)}(\dots)$, il est possible de remarquer que tous les éléments non-diagonaux sont:

- soit nuls;
- soit égaux à un élément non-diagonal d'une des matrices $Q_l^{(i)}(\dots)$.

Les éléments diagonaux, par contre, sont toujours des sommes d'éléments diagonaux de chaque matrice $Q_l^{(i)}(\dots)$. Ceci nous permet d'affirmer que le nombre d'éléments non nuls d'une matrice creuse générée par une somme tensorielle $\bigoplus_{g\,i=1}^{N} Q_l^{(i)}(\dots)$ est inférieur à

$$\sum_{i=1}^{N} (\bar{n}_i \times nz_i)$$

s'il existe au moins deux matrices $Q_l^{(i)}(\dots)$ avec des éléments diagonaux non nuls. Prenons par exemple, un modèle RAS pour le partage de ressources (section 4.1). Cet exemple avec 12 clients ($N = 12$) et 12 ressources ($P = 12$) équivaut à une matrice creuse de dimension 4096 et 53248 éléments non nuls[12]. Le nombre de multiplications pour le descripteur de ce modèle est égal à:

$$\sum_{i=1}^{12} (2048 \times 4) = 98304$$

[12]Il faut rappeler que ces paramètres ($N = 12$ et $P = 12$) transforment ce descripteur en un modèle sans fonctions, car chacun des 12 clients trouvera toujours une des 12 ressources libre.

Dans le cas où les matrices $Q_l^{(i)}(\dots)$ sont pleines, il est facile de vérifier que les 45056 multiplications faites en plus correspondent aux éléments diagonaux. Dans la matrice creuse chaque élément diagonal est multiplié une seule fois, tandis que, dans le descripteur chacun des 12 facteurs normaux a une diagonale et ces 12 diagonales sont traitées séparément. Si on enlève du nombre d'éléments non nuls de la matrice creuse les éléments diagonaux, il reste 49152 éléments. Si on enlève du nombre de multiplications pour le descripteur les multiplications des éléments diagonaux des 12 facteurs normaux (12×4096), il reste également 49152 multiplications.

Compte tenu des désavantages de manipuler les éléments diagonaux des sommes tensorielles avec l'algorithme proposé, la première optimisation pour la multiplication d'un vecteur par le descripteur est le pré-calcul de la *diagonale du descripteur*. Pour la diagonale du descripteur on entend le vecteur contenant les éléments de la diagonale de la matrice obtenue en évaluant toutes les opérations de la formule du descripteur.

Le descripteur $Q = \sum_{j=1}^{(N+2E)} \bigotimes_{g_i \in \eta} Q_j^{(i)}$ (équation 3.2) est décomposé en deux parties:

- D, une matrice diagonale avec la diagonale du descripteur;
- $\bar{Q}$, le descripteur selon la formule 3.2, mais où tous les éléments diagonaux d'une matrice de chaque terme produit tensoriel ont été mis à zéro.

Dans la pratique tous les éléments diagonaux des matrices locales ($Q_l^{(i)}$) sont mis à zéro. Ceci représente une matrice de chaque facteur normal de la partie somme tensorielle du descripteur. Dans chaque terme produit tensoriel correspondant à l'événement synchronisant e du descripteur, les éléments diagonaux des matrices correspondant à l'automate maître de cet événement ($Q_{e^+}^{(\iota^{(e)})}$ et $Q_{e^-}^{(\iota^{(e)})}$) sont mis à zéro.

Un deuxième avantage de cette optimisation n'apparaît que pour les descripteurs ayant des événements synchronisants. Chaque événement synchronisant génère deux termes produit tensoriels. Le premier terme contient les taux d'occurrence de l'événement synchronisant. Le deuxième terme contient exclusivement des éléments pour l'ajustement diagonal. Ce deuxième terme, une fois évalué, va générer uniquement des éléments sur la diagonale du descripteur. Le pré-calcul de la diagonale permet donc d'éliminer le second terme produit tensoriel de chaque événement synchronisant. Ceci réduit le nombre de termes à traiter lors de la multiplication du descripteur sans pour autant ajouter des multiplications. Les éléments diagonaux venus des termes synchronisants se rajoutent aux éléments diagonaux correspondant à la partie somme tensorielle du descripteur. Le nombre de termes produit tensoriel du descripteur est réduit de $2E + N$ à $E + N$.

Mesures Numériques

Le pré-calcul de la diagonale du descripteur a un coût, cependant ce coût ne se manifeste qu'une seule fois lors de la préparation du descripteur. Le bénéfice apporté par cette optimisation, par contre, se manifeste à chaque multiplication vecteur descripteur exécutée.

Les temps d'exécution[13] de 100 multiplications pour le modèles RAS de partage des ressources avec $N = 12$ et $P = 12$ sont:

– pour le cas non optimisé:

  8.2 secondes, dont 6.7 secondes exclusivement pour les multiplications de tranches de vecteur par $Q_j^{(i)}$;

– pour le cas optimisé avec le pré-calcul de la diagonale:

  5.2 secondes, dont 3.7 secondes pour les multiplications de tranches de vecteur par $Q_j^{(i)}$ et pour la multiplication par la diagonale pré-calculée;

Pour des descripteurs avec éléments fonctionnels, cette optimisation apporte encore plus d'avantages à cause de l'évaluation des éléments fonctionnels diagonaux. Un désavantage de cette optimisation est la nécessité de stocker les éléments diagonaux. Reprenant l'exemple de partage de ressources avec 12 clients et 12 ressources, le pré-calcul de la diagonale du descripteur change les besoins de stockage de:

– 12 matrices avec 4 éléments non nuls; à

– 12 matrices avec 2 éléments non nuls (éléments non diagonaux), plus un vecteur de 4096 éléments (diagonale pré-calculée).

Cette augmentation de l'utilisation de la mémoire reste quand même faible comparée aux besoins de stockage d'une matrice creuse.

Cette optimisation permet aussi un accès rapide à la diagonale du descripteur. Parmi les avantages de cet accès à la diagonale on a:

– l'accès facile au plus grand élément du descripteur, car, le descripteur étant une représentation d'un générateur infinitésimal, le plus grand élément en module est toujours parmi les éléments diagonaux;

– la facilité d'utilisation de certaines techniques de pré-conditionnement utilisant la diagonale.

5.3.2 Re-ordonnancement des Automates

Cette optimisation se comprend aisément en observant des cas particuliers de produits tensoriels généralisés. Prenons par exemple le produit

$$v \times \bigotimes_{i=1}^{N}{}_{g} Q^{(i)}(\mathcal{A}^{(i+1)}, \ldots, \mathcal{A}^{(N)})$$

[13]Tous les temps d'exécution ont été mesurés avec une précision du dixième de seconde dans le logiciel PEPS (voir chapitre 7) sur une station IBM RS6000 avec système AIX 4.2 et 96 Mega octets de mémoire vive.

Les facteurs normaux de la décomposition doivent être traités dans l'ordre suivant:

$$\begin{aligned}
&Q^{(1)}(\mathcal{A}^{(2)},\dots,\mathcal{A}^{(N)}) \underset{g}{\otimes} I_{nright_1} \\
&\times I_{n_1} \underset{g}{\otimes} Q^{(2)}(\mathcal{A}^{(3)},\dots,\mathcal{A}^{(N)}) \underset{g}{\otimes} I_{nright_2} \\
&\times \cdots \\
&\times I_{nleft_{N-1}} \underset{g}{\otimes} Q^{(N-1)}(\mathcal{A}^{(N)}) \underset{g}{\otimes} I_{n_N} \\
&\times I_{nleft_N} \underset{g}{\otimes} Q^{(N)}
\end{aligned}$$

De plus, on peut observer que chaque matrice $Q^{(i)}(\dots)$ ne dépend que des automates à sa droite. Cela nous permet d'éliminer les lignes correspondant au calcul des états locaux des automates à sa gauche (lignes 3 et 22) dans l'algorithme 5.8.

De façon analogue, pour le produit

$$v \times \bigotimes_{i=1}^{N}{}_g\, Q^{(i)}(\mathcal{A}^{(1)},\dots,\mathcal{A}^{(i-1)})$$

Les facteurs normaux de la décomposition doivent être traités dans l'ordre suivant:

$$\begin{aligned}
&I_{nleft_N} \underset{g}{\otimes} A^{(N)}(\mathcal{A}^{(1)},\dots,\mathcal{A}^{(N-1)}) \\
&\times I_{nleft_{N-1}} \underset{g}{\otimes} A^{(N-1)}(\mathcal{A}^{(1)},\dots,\mathcal{A}^{(N-2)}) \underset{g}{\otimes} I_{n_N} \\
&\times \cdots \\
&\times I_{n_1} \underset{g}{\otimes} A^{(2)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} I_{nright_2} \\
&\times A^{(1)} \underset{g}{\otimes} I_{nright_1}
\end{aligned}$$

Puisque les matrices $Q^{(i)}(\dots)$ ne dépendent que des automates à sa gauche, ce sont les lignes correspondant au calcul des états locaux des automates à sa droite (lignes 6 et 19) qui peuvent être supprimées. De plus, l'évaluation des matrices (ligne 12 de l'algorithme 5.8) peut être déplacée à l'extérieur de la boucle sur j de 1 jusqu'à $nright_i$. L'algorithme 5.9 montre les changements de ce cas particulier.

Re-ordonnancement des Termes Produit Tensoriel

Tout terme produit tensoriel sans cycle de dépendance fonctionnelle, *i.e.*, tout produit tensoriel où l'algorithme 5.8 peut être directement appliqué, peut être re-ordonné de façon à prendre la forme $\bigotimes_{g\,i=1}^{N} Q^{(i)}(\mathcal{A}^{(1)},\dots,\mathcal{A}^{(i-1)})$. La deuxième optimisation proposée est le re-ordonnancement des matrices de chaque terme produit tensoriel de façon à réduire les calculs des états locaux et le nombre d'évaluations pour les matrices $Q^{(i)}(\dots)$. Ce re-ordonnancement définit un nouvel ordre appelé *ordre de dépendance*.

Il faut remarquer que l'*ordre de dépendance* n'est pas le même que celui introduit lors du traitement de la multiplication des produits tensoriels sans cycle de dépendance fonctionnelle, alors appelé *ordre de décomposition*. L'*ordre de décomposition* établit simplement l'ordre dans lequel les facteurs normaux doivent être multipliés selon les propriétés

Algorithme 5.9

```
1  for i = N, N − 1, ... , 1
2  do  base = 0;
3      initialize   k_{1=1}, k_2 = 1, ... , k_{i−1} = 1;
4      for k = 1, 2, ... , nleft_i
5      do  for j = 1, 2, ... , nright_i
6          do  evaluate   Q^(i)(a_{k_1}^(1), ... , a_{k_{i−1}}^(i−1));
7              index = base + j;
8              for l = 1, 2, ... , n_i
9              do  z_in[l] = v[index];
10                 index = index + nright_i;
11             end do
12             multiply   z_out = z_in × Q^(i);
13             index = base + j;
14             for l = 1, 2, ... , n_i
15             do  v[index] = z_out[l];
16                 index = index + nright_i;
17             end do
18         end do
19         base = base + ( nright_i × n_i);
20         next   k_1, k_2, ... , k_{i−1};
21     end do
22 end do
```

Algorithme 5.9: Multiplication $v \times \bigotimes_{g\,i=1}^{N} Q^{(i)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(i-1)})$

de décomposition en facteurs normaux des produits tensoriels généralisés. Mais la suite de matrices garde toujours le même format, *i.e.,* la matrice représentant le premier automate est suivie de celle représentant le second et ainsi de suite. L'*ordre de dépendance* implique un changement plus important qui re-ordonne la suite de matrices elle-même. Étant donné un *ordre de dépendance* exprimé par une permutation σ, la suite de matrices est changée, *i.e.,* la matrice représentant le σ_1-ème automate est suivie de celle représentant le σ_2-ème automate et ainsi de suite. Ce changement occasionne un re-calcul des valeurs de $nleft_i$ et $nright_i$. Les matrices à gauche de la matrice $Q^{(i)}$ selon l'*ordre de dépendance* σ deviennent les matrices $Q^{(k)}$ où $\sigma_k < \sigma_i$. De la même façon, les matrices à droite de $Q^{(i)}$ deviennent les matrices $Q^{(k)}$ où $\sigma_k > \sigma_i$. Pour éviter de confondre ces deux concepts d'ordre, nous allons définir les notations suivantes:

✍ **Soit**

$\sigma left_i$ le produit des dimensions de toutes les matrices à gauche de la i-ème matrice d'une suite re-ordonnée selon un permutation σ, *i.e.,* $\prod_{k=1}^{i-1} n_{\sigma_k}$ (cas particulier: $\sigma left_1 = 1$);

$\sigma right_i$ le produit des dimensions de toutes les matrices à droite de la i-ème

matrice d'une suite re-ordonnée selon un permutation σ, *i.e.*, $\prod_{k=i+1}^{N} n_{\sigma_k}$ (cas particulier: $\sigma right_N = 1$);

Notons que la définition de $\bar{n}_i$ reste la même, *i.e.*, le produit des dimensions de toutes les matrices sauf la i-ème matrice d'une suite. Prenons un exemple avec le produit tensoriel généralisé suivant:

$$Q^{(1)}(\mathcal{A}^{(6)}) \underset{g}{\otimes} Q^{(2)}(\mathcal{A}^{(5)}) \underset{g}{\otimes} Q^{(3)}(\mathcal{A}^{(2)}) \underset{g}{\otimes} Q^{(4)}(\mathcal{A}^{(3)}) \underset{g}{\otimes} Q^{(5)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} Q^{(6)}$$

L'ordre de dépendance pour ce terme est exprimé par la permutation $\sigma = \{4, 3, 2, 5, 1, 6\}$, *i.e.*:

$$Q^{(4)}(\mathcal{A}^{(3)}) \underset{g}{\otimes} Q^{(3)}(\mathcal{A}^{(2)}) \underset{g}{\otimes} Q^{(2)}(\mathcal{A}^{(5)}) \underset{g}{\otimes} Q^{(5)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} Q^{(1)}(\mathcal{A}^{(6)}) \underset{g}{\otimes} Q^{(6)}$$

Pour cet exemple on obtient:

Ordre originel $\{1,2,3,4,5,6\}$		
i	$nleft_i$	$nright_i$
1	1	$n_2n_3n_4n_5n_6$
2	n_1	$n_3n_4n_5n_6$
3	n_1n_2	$n_4n_5n_6$
4	$n_1n_2n_3$	n_5n_6
5	$n_1n_2n_3n_4$	n_6
6	$n_1n_2n_3n_4n_6$	1

Ordre de dépendance $\{4,3,2,5,1,6\}$		
i	$\sigma left_i$	$\sigma right_i$
1	$n_4n_3n_2n_5$	n_6
2	n_4n_3	$n_5n_1n_6$
3	n_4	$n_2n_5n_1n_6$
4	1	$n_3n_2n_5n_1n_6$
5	$n_4n_3n_2$	n_1n_6
6	$n_4n_3n_2n_5n_1$	1

Il faut remarquer que l'établissement d'un *ordre de dépendance* rend inutile l'établissement d'un *ordre de décomposition*. Admettons qu'un produit tensoriel soit re-ordonné, avec une permutation σ, pour la forme $\bigotimes_{g\ i=1}^{N} Q^{(\sigma_i)}(\mathcal{A}^{(\sigma_1)}, \ldots, \mathcal{A}^{(\sigma_{i-1})})$. Alors, tous les paramètres de la matrice $Q^{(\sigma_i)}$ sont à sa gauche. L'ordre de décomposition, *i.e.*, l'ordre dans lequel les facteurs normaux doivent être multipliés est l'inverse de l'ordre de dépendance. Ceci peut être observé pour le cas du produit $\bigotimes_{g\ i=1}^{N} Q^{(i)}(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(i-1)})$ qui ne nécessite pas de re-ordonnancement de dépendance. L'ordre de décomposition de ce produit est égal à $\{N, N-1, \ldots, 2, 1\}$.

Ceci nous amène à l'algorithme 5.10 qui multiplie un terme produit tensoriel généralisé selon un ordre de dépendances défini par une permutation σ. Avant l'application de cet algorithme le vecteur d'entrée (v) doit avoir ses éléments permutés selon l'ordre lexicographique correspondant à la permutation σ (permutation P_σ). Ceci introduit un coût additionnel qui n'est pas exprimé dans dans la multiplication d'un seul terme tensoriel (algorithme 5.10), mais qui doit être considéré dans la multiplication d'un descripteur entier (somme de termes tensoriels).

Pour illustrer cette optimisation prenons par exemple le terme produit tensoriel généralisé avec les dépendances fonctionnelles suivantes:

$$Q^{(1)} \underset{g}{\otimes} Q^{(2)}(\mathcal{A}^{(3)}) \underset{g}{\otimes} Q^{(3)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} Q^{(4)} \tag{5.4}$$

Algorithme 5.10

```
1  for i = σ_N, σ_{N-1}, ..., σ_1
2  do  base = 0;
3      initialize   k_{σ_1} = 1, k_{σ_2} = 1, ..., k_{σ_{i-1}} = 1;
4      for k = 1, 2, ..., σleft_i
5      do  for j = 1, 2, ..., σright_i
6          do  evaluate   Q^(i)(a^(σ_1)_{k_{σ_1}}, ..., a^(σ_{i-1})_{k_{σ_{i-1}}});
7              index = base + j;
8              for l = 1, 2, ..., n_i
9              do  z_in[l] = v[index];
10                 index = index + σright_i;
11             end do
12             multiply   z_out = z_in × Q^(i);
13             index = base + j;
14             for l = 1, 2, ..., n_i
15             do  v[index] = z_out[l];
16                 index = index + σright_i;
17             end do
18         end do
19         base = base + (σright_i × n_i);
20         next   k_{σ_1}, k_{σ_2}, ..., k_{σ_{i-1}};
21     end do
22 end do
```

Algorithme 5.10: Multiplication $v \times \bigotimes_{g\, i=1}^{N} Q^{(i)}(\dots)$

Ce terme peut être re-ordonné de la façon suivante, permutation $\sigma = \{1, 3, 2, 4\}$:

$$Q^{(1)} \underset{g}{\otimes} Q^{(3)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} Q^{(2)}(\mathcal{A}^{(3)}) \underset{g}{\otimes} Q^{(4)} \tag{5.5}$$

L'algorithme 5.8 (non-optimisé) appliqué au terme 5.4 exécute:

- $n_1 \times n_3 \times n_4 = \bar{n}_2$ évaluations de la matrice $Q^{(2)}(\mathcal{A}^{(3)})$;
- $n_1 \times n_2 \times n_4 = \bar{n}_3$ évaluations de la matrice $Q^{(3)}(\mathcal{A}^{(1)})$.

Le terme 5.5, par contre peut être multiplié avec l'algorithme 5.10. Le nombre d'évaluations de matrices fonctionnelles devient:

- $n_1 \times n_3$ évaluations de la matrice $Q^{(2)}(\mathcal{A}^{(3)})$;
- n_1 évaluations de la matrice $Q^{(3)}(\mathcal{A}^{(1)})$.

Pour chaque terme produit tensoriel, un ordre de dépendance est choisi. Il faut rappeler que pour certains produits tensoriels il est possible d'établir plusieurs ordres valides (voir la discussion présentée dans la section 2.2.6, page 42). Le meilleur ordre sera, en principe, celui qui minimise le nombre d'évaluations de matrices fonctionnelles. Bien qu'on

puisse établir certaines heuristiques pour automatiser le choix d'un ordre de dépendances optimale, le choix du meilleur ordre reste toujours un problème ouvert. Or, le coût de chacune des évaluations peut être très distinct. En effet, le coût de l'évaluation d'une matrice dépend à la fois du nombre d'éléments fonctionnels et de leur expression mathématique (certains éléments fonctionnels peuvent être très coûteux à évaluer).

Re-ordonnancement de Facteurs Normaux

Une autre possibilité d'optimisation analogue au re-ordonnancement des termes produits tensoriels est le re-ordonnancement des facteurs normaux. Chaque terme produit tensoriel est décomposé en facteurs normaux qui sont, eux aussi, des termes produits tensoriels plus simples. Les graphes de dépendance fonctionnelle des termes produits tensoriels peuvent être assez compliqués et plusieurs ordres peuvent être choisis. Les facteurs normaux étant plus simples, l'ordre dans lequel on minimise les évaluations des matrices $Q^{(i)}(\dots)$ est évident, car seulement la matrice $Q^{(i)}(\dots)$ est fonctionnelle (les autres matrices sont des identités).

Reprenons l'exemple du terme produit tensoriel décomposable en facteurs normaux:

$$\begin{aligned} Q^{(1)} \underset{g}{\otimes} Q^{(2)}(\mathcal{A}^{(3)}) \underset{g}{\otimes} Q^{(3)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} Q^{(4)} = \\ I_{n_1} \underset{g}{\otimes} Q^{(2)}(\mathcal{A}^{(3)}) \underset{g}{\otimes} I_{n_3} \underset{g}{\otimes} I_{n_4} \\ \times I_{n_1} \underset{g}{\otimes} I_{n_2} \underset{g}{\otimes} Q^{(3)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} I_{n_4} \\ \times Q^{(1)} \underset{g}{\otimes} I_{n_2} \underset{g}{\otimes} I_{n_3} \underset{g}{\otimes} I_{n_4} \\ \times I_{n_1} \underset{g}{\otimes} I_{n_2} \underset{g}{\otimes} I_{n_3} \underset{g}{\otimes} Q^{(4)} \end{aligned}$$

Le premier facteur normal peut être re-ordonné de la façon suivante[14]:

$$I_{n_3} \underset{g}{\otimes} Q^{(2)}(\mathcal{A}^{(3)}) \underset{g}{\otimes} I_{n_1} \underset{g}{\otimes} I_{n_4}$$

L'application de l'algorithme 5.10 à ce facteur normal (re-ordonné) fait que le nombre d'évaluations de la matrice $Q^{(2)}(\mathcal{A}^{(3)})$ devient n_3.

Le deuxième facteur normal peut être re-ordonné de la façon suivante:

$$I_{n_1} \underset{g}{\otimes} Q^{(3)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} I_{n_2} \underset{g}{\otimes} I_{n_4}$$

L'application de l'algorithme 5.10 à ce facteur normal (re-ordonné) fait que le nombre d'évaluations de la matrice $Q^{(3)}(\mathcal{A}^{(1)})$ devient n_1. Les deux derniers facteurs normaux, n'ayant que des matrices constantes, n'ont pas besoin d'être re-ordonnés.

Le tableau 5.1 montre le nombre d'évaluations des matrices fonctionnelles pour le cas non-optimisé, pour l'optimisation de re-ordonnancement des termes produit tensoriels et pour l'optimisation de re-ordonnancement des facteurs normaux. Dans ce tableau est aussi indiqué le nombre de permutations de vecteurs nécessaires dans chacun des cas.

[14]Rappelons qu'un tel re-ordonnancement nécessite deux permutations de vecteurs.

Le traitement d'un terme (produit tensoriel ou facteur normal) permuté dans l'ordre de dépendance σ implique deux permutations. La première permute le vecteur à multiplier (vecteur d'entrée multiplié par P_σ). La deuxième permutation correspond à revenir dans l'ordre originel (vecteur de sortie multiplié par P_σ^T). Cependant une seule permutation est compatibilisée, car on peut enchaîner les traitements de deux termes correspondants à des ordres distincts en faisant une seule permutation entre eux.

$$Q^{(1)} \underset{g}{\otimes} Q^{(2)}(\mathcal{A}^{(3)}) \underset{g}{\otimes} Q^{(3)}(\mathcal{A}^{(1)}) \underset{g}{\otimes} Q^{(4)}$$

Re-ordonnancement	évaluations de $Q^{(2)}(\mathcal{A}^{(3)})$	évaluations de $Q^{(3)}(\mathcal{A}^{(1)})$	permutations de vecteur
aucun	$n_1 \times n_3 \times n_4$	$n_1 \times n_2 \times n_4$	aucune
du produit tensoriel	$n_1 \times n_3$	n_1	une
du facteur normal	n_3	n_1	deux

TAB. 5.1: Comparaison entre Optimisations de Re-ordonnancement

Mesures Numériques

Ces deux optimisations peuvent, dans certains cas, avoir des temps d'exécution plus grands que la solution non-optimisée. Le coût des permutations de vecteur doit être comparé aux gains apportés par cette optimisation.

Pour analyser l'impact de ces optimisations dans le temps d'exécution, on va partager le temps d'exécution en trois tâches différentes:

- La multiplication des tranches du vecteur par les matrices $Q^{(i)}$ et par la diagonale du descripteur[15], appelée *mult*;

  ligne 13 de l'algorithme 5.8 et ligne 12 de l'algorithme 5.10;

- Le calcul des états locaux et l'évaluation des matrices $Q^{(i)}$, appelée *éval*;

  lignes 3, 6, 12, 19 et 22 de l'algorithme 5.8 et lignes 3, 6 et 20 de l'algorithme 5.10;

- Les permutations de vecteur qui doivent être faites lors d'un re-ordonnancement sur produits tensoriels ou sur facteurs normaux, appelée *permut*.

Les temps d'exécution[16] nécessaire pour une multiplication vecteur descripteur sont affichés.

Pour tous les exemples on va appeler les options algorithmiques:

- Méthode A, l'application de l'algorithme 5.8 sans aucun re-ordonnancement;
- Méthode B, l'application de l'algorithme 5.10 avec re-ordonnancements à chaque terme produit tensoriel, si nécessaire;

[15]L'exécution de la multiplication par la diagonale du descripteur correspond à l'optimisation décrite dans la section 5.3.1. Dans tous les mesures présentées cette optimisation est utilisée.

[16]Tous les temps ont été mesurés avec une précision du dixième de seconde dans le logiciel PEPS (voir chapitre 7) sur une station IBM RS6000 avec système AIX 4.2 et 96 Mega octets de mémoire vive.

– Méthode C, l'application de l'algorithme 5.10 avec re-ordonnancements à chaque facteur normal, si nécessaire;

Prenons comme exemple le RAS décrit dans la section 4.1 modélisant le partage de $P = 1, 8, 15, 16$ ressources entre $N = 16$ clients. La taille de l'espace d'état de tous ces modèles est de $2^{16} = 64536$, bien que le nombre d'états atteignables soit plus petit et dépendant du nombre de ressources P. Il faut remarquer que le premier exemple, $N = 16$ et $P = 16$ est un cas sans éléments fonctionnels. Le tableau 5.2 présente les temps d'exécution pour ces quatre modèles[17].

Modèle	Méthode	total	*mult*	*éval*	*permut*
$N = 16\ P = 16$	A	1.6s	1.0s	0.0s	0.0s
	B	1.6s	1.0s	0.0s	0.0s
	C	1.6s	1.0s	0.0s	0.0s
$N = 16\ P = 15$	A	17.2s	1.0s	15.7s	0.0s
	B	23.3s	1.0s	15.7s	6.1s
	C	23.3s	1.0s	15.7s	6.1s
$N = 16\ P = 8$	A	17.2s	1.0s	15.7s	0.0s
	B	23.3s	1.0s	15.7s	6.1s
	C	23.3s	1.0s	15.7s	6.1s
$N = 16\ P = 1$	A	17.2s	1.0s	15.7s	0.0s
	B	23.3s	1.0s	15.7s	6.1s
	C	23.3s	1.0s	15.7s	6.1s

TAB. 5.2: Mesures pour le Modèle de Partage de Ressources

Pour cet exemple, sans événements synchronisants, la première observation est l'équivalence entre les méthodes B et C. Les RAS sans événements synchronisants sont des sommes tensorielles, *i.e.*, sommes de facteurs normaux, donc les termes produit tensoriels sont déjà des facteurs normaux. La deuxième observation intéressante est de voir que le re-ordonnancement des automates, ajoutant 6.1 secondes pour les aux permutations, n'apporte aucun gain au niveau de l'évaluation des fonctions. En regardant les dépendances fonctionnelles de ce modèle, la raison de ce phénomène apparaît. Les éléments fonctionnels, ayant comme paramètres l'état de tous les automates, forcent les termes produits tensoriels à avoir la forme:

$$I_{n_1} \underset{g}{\otimes} \cdots \underset{g}{\otimes} I_{n_{i-1}} \underset{g}{\otimes} I_{n_{i+1}} \underset{g}{\otimes} \cdots \underset{g}{\otimes} I_{n_N} \underset{g}{\otimes} Q_l^{(i)}$$

Ceci oblige l'évaluation de la matrice $Q_l^{(i)}$ pour un nombre de $\bar{n}_i$ fois, exactement comme dans la méthode A.

Prenons maintenant un deuxième exemple, le RAS décrit dans la section 4.3.2 qui modélise un réseaux de trois files d'attente visitées par deux classes de clients. Prenons les capacités de files $C_1 = 10$, $C_2 = 40$ et $C_3 = 50$. Définissons aussi deux variantes du modèle RAS présenté:

[17]La différence entre le total et la somme de *mult*, *éval* et *permut* représente les coûts de gestion des boucles des algorithmes ainsi que les coûts d'extraction des tranches du vecteur.

– inclusion d'une nouvelle fonction pour le taux de service de la file s_2 (μ_2);
– inclusion d'une nouvelle fonction très coûteuse pour le taux de service de la file s_2 (μ_2);

La première variante, remplace la valeur constante μ_2 du taux de service de la file s_2 par une fonction f définie par:

$$f = \frac{\mu_2}{1 + a^{(3_2)}}$$

Cette fonction représente un ralentissement de de la file s_2 selon le nombre de clients de la classe 2 dans la file s_3. Le principal effet de ce changement est la définition d'une nouvelle dépendance fonctionnelle de la matrice contenant le taux μ_2 envers l'état local de l'automate $\mathcal{A}^{(3_2)}$.

La deuxième variante représente le même ralentissement pour le taux de service de la file s_2, mais dans ce cas le nouveau taux est exprimé par une fonction plus coûteuse à évaluer:

$$f = \mu_2 \times \sum_{i=0}^{C_3-1} \frac{1}{1+i} \delta(a^{(3_2)} = i)$$

Il faut remarquer qu'il s'agit d'une fonction retournant exactement le même résultat que dans la variation précédente. Seulement ici la fonction est tabulée, donc plus laborieuse à évaluer. Le but de cette variante est d'amplifier les coûts d'évaluation des matrices $Q^{(i)}(\dots)$. Le tableau 5.3 présente les temps d'exécution pour ces trois modèles[18].

Modèle	Méthode	total	*mult*	*éval*	*permut*
originel	A	64.8s	5.3s	57.5s	0.0s
	B	21.1s	5.3s	0.1s	13.7s
	C	18.7s	5.3s	0.1s	11.6s
variante 1	A	79.6s	5.3s	72.3s	0.0s
	B	22.6s	5.3s	1.6s	13.7s
	C	20.0s	5.3s	0.2s	12.6s
variante 2	A	259.4s	5.3s	252.1s	0.0s
	B	40.6s	5.3s	19.6s	13.7s
	C	20.4s	5.3s	0.6s	12.6s

TAB. 5.3: Mesures pour les Modèles de Trois Files d'Attente

Pour le modèle originel, ainsi que pour les variantes il est clair que la méthode A est bien plus mauvaise que les méthodes B et C. La réduction du nombre d'évaluations de fonction, justifie largement les temps d'exécution des permutations. Les variantes, ayant un plus grand nombre de fonctions, montrent plus nettement le bénéfice apporté par les optimisations de re-ordonnancement.

[18]La différence entre le total et la somme de *mult*, *éval* et *permut* représente les coûts de gestion des boucles des algorithmes ainsi que les coûts d'extraction des tranches du vecteur.

L'analyse de ces exemples doit se faire en regardant pour chacun des modèles les termes produits tensoriels en question. Le modèle originel possède trois termes constants (correspondant aux transitions locales des automates $\mathcal{A}^{(1)}$, $\mathcal{A}^{(2)}$ et $\mathcal{A}^{(3_1)}$) et les trois termes fonctionnels suivants:

$$\begin{array}{ccccccc} I_{n_1} & \underset{g}{\otimes} & I_{n_2} & \underset{g}{\otimes} & I_{n_{3_1}} & \underset{g}{\otimes} & Q_l^{(3_2)}(\mathcal{A}^{(3_1)}) \\ Q_{s_1}^{(1)} & \underset{g}{\otimes} & I_{n_2} & \underset{g}{\otimes} & Q_{s_1}^{(3_1)}(\mathcal{A}^{(3_2)}) & \underset{g}{\otimes} & I_{n_{3_2}} \\ I_{n_1} & \underset{g}{\otimes} & Q_{s_2}^{(2)} & \underset{g}{\otimes} & I_{n_{3_1}} & \underset{g}{\otimes} & Q_{s_1}^{(3_2)}(\mathcal{A}^{(3_1)}) \end{array}$$

Tous les termes fonctionnels demandent une seule permutation par terme produit tensoriel. Ceci est vrai pour la méthode B par définition et pour la méthode C parce que les termes possèdent une seule matrice fonctionnelle.

Les modèles des deux variantes ont les mêmes dépendances fonctionnelles. Les termes correspondant aux transitions locales des automates $\mathcal{A}^{(1)}$, $\mathcal{A}^{(2)}$ et $\mathcal{A}^{(3_1)}$ sont toujours constants. Les termes fonctionnels dans les modèles des deux variantes sont:

$$\begin{array}{ccccccc} I_{n_1} & \underset{g}{\otimes} & I_{n_2} & \underset{g}{\otimes} & I_{n_{3_1}} & \underset{g}{\otimes} & Q_l^{(3_2)}(\mathcal{A}^{(3_1)}) \\ Q_{s_1}^{(1)} & \underset{g}{\otimes} & I_{n_2} & \underset{g}{\otimes} & Q_{s_1}^{(3_1)}(\mathcal{A}^{(3_2)}) & \underset{g}{\otimes} & I_{n_{3_2}} \\ I_{n_1} & \underset{g}{\otimes} & Q_{s_2}^{(2)}\mathcal{A}^{(3_2)} & \underset{g}{\otimes} & I_{n_{3_1}} & \underset{g}{\otimes} & Q_{s_1}^{(3_2)}(\mathcal{A}^{(3_1)}) \end{array}$$

On observe que le dernier terme fonctionnel, possédant deux matrices fonctionnelles, va demander une seule permutation pour la méthode B (par définition), tandis que la méthode C demandera deux permutations pour ce même terme.

L'inclusion d'une fonction pour le taux de service de la file s_2 (variante 1) augmente le nombre d'évaluations pour la méthode B et augmente le nombre de permutations pour la méthode C. Les changements du modèle originel à la première variante ne sont pas très significatifs, car la fonction introduite n'est pas chère à calculer. La deuxième variante, qui introduit une fonction bien plus chère à évaluer, montre des avantages beaucoup plus importants dans l'utilisation de la méthode C.

5.3.3 Groupement d'Automates

Cette optimisation est basée sur la réduction d'un RAS vers un RAS *équivalent* avec moins d'automates. La notion d'équivalence est faite avec une transformation algébrique du générateur infinitésimal au quel le descripteur du RAS correspond.

Étant donné un RAS bien défini, son descripteur (équation 3.1) est:

$$Q = \bigoplus_{i=1}^{N}{}_g\, Q_l^{(i)} + \sum_{e=1}^{E} \left[\bigotimes_{i=1}^{N}{}_g\, Q_{e^+}^{(i)} + \bigotimes_{i=1}^{N}{}_g\, Q_{e^-}^{(i)} \right] \tag{5.6}$$

Maintenant définissons B groupes nommés $b_1, \ldots, b_B$ et, sans perte de généralité[19] supposons que le groupe b_k est composé par les indices $[c_k + 1..c_{k+1}]$, avec $c_1 = 0$ et

[19]La définition de groupes n'est pas forcément faite avec des automates d'indices contigus, mais pour le confort des notations on va décrire seulement cette possibilité.

$c_{B+1} = N$. Cela nous donne les groupes d'indices $b_1 = [1..c_2]$, $b_2 = [c_2 + 1..c_3]$, ..., $b_B = [c_B + 1..N]$. Avec la propriété d'associativité des sommes et produits tensoriels généralisés (équation 2.24), le descripteur peut être redéfini par:

$$Q = \bigoplus_{i=1}^{B}{}_g \left(\bigoplus_{j=c_k+1}^{c_{k+1}}{}_g Q_l^{(j)} \right) + \sum_{e=1}^{E} \left[\bigotimes_{i=1}^{B}{}_g \left(\bigotimes_{j=c_k+1}^{c_{k+1}}{}_g Q_{e^+}^{(j)} \right) + \bigotimes_{i=1}^{B}{}_g \left(\bigotimes_{j=c_k+1}^{c_{k+1}}{}_g Q_{e^-}^{(j)} \right) \right]$$

Définissons les matrices:

- $R_l^{(k)} = \bigoplus_{g\ j=c_k+1}^{c_{k+1}} Q_l^{(j)}$;
- $R_{e^+}^{(k)} = \bigotimes_{g\ j=c_k+1}^{c_{k+1}} Q_{e^+}^{(j)}$;
- $R_{e^-}^{(k)} = \bigotimes_{g\ j=c_k+1}^{c_{k+1}} Q_{e^-}^{(j)}$.

Ces matrices sont par définition les matrices correspondant aux *automates groupés*, appelés $\mathcal{G}^{(k)}$ d'un RAS équivalent à celui défini en 5.6, *i.e.*:

$$Q = \bigoplus_{i=1}^{B}{}_g R_l^{(k)} + \sum_{e=1}^{E} \left[\bigotimes_{i=1}^{B}{}_g R_{e^+}^{(k)} + \bigotimes_{i=1}^{B}{}_g R_{e^-}^{(k)} \right] \tag{5.7}$$

L'espace produit d'états de chaque automate $\mathcal{G}^{(k)}$ est $\prod_{j=c_k+1}^{c_{k+1}} n_j$. Cette formulation, purement algébrique, est la base de l'optimisation par groupement d'automates.

Trois avantages peuvent venir de cette optimisation:

- Élimination d'événements synchronisants;
- Élimination d'éléments fonctionnels;
- Réduction de l'espace d'état;

Élimination d'Événements Synchronisants

L'élimination d'événements synchronisants se produit lorsqu'un événement synchronisant s ne concerne que les automates d'un groupe b_k quelconque, *i.e.*:

$$\forall i \in [1..N] \text{ et } i \notin b_k \text{ on a } Q_{s^+}^{(i)} = I_{n_i} \text{ et } Q_{s^-}^{(i)} = I_{n_i}$$

De ce fait, l'événement s deviens local à l'automate groupé $\mathcal{G}^{(k)}$. La valeur de $R_l^{(k)}$ est changée pour simplifier le descripteur en utilisant comme nouvelle valeur:

$$R_l^{(k)} = R_l^{(k)} + R_{s^+}^{(k)} + R_{s^-}^{(k)}$$

Le descripteur devient:

$$Q = \bigoplus_{i=1}^{B}{}_g R_l^{(k)} + \sum_{e=1, e\neq s}^{E} \left[\bigotimes_{i=1}^{B}{}_g R_{e^+}^{(k)} + \bigotimes_{i=1}^{B}{}_g R_{e^-}^{(k)} \right]$$

Élimination d'Éléments Fonctionnels

L'élimination d'éléments fonctionnels suit le même raisonnement, se manifestant lorsqu'un élément fonctionnel, se trouvant dans une matrice $Q_j^{(i)}$ appartenant à un groupe b_k, ne dépend que des automates $\mathcal{A}^{(l)}$ appartenant au même groupe b_k. Ceci nous permet d'évaluer les éléments fonctionnels internes à un groupe et à les remplacer par des éléments constants. Cependant, si certains éléments fonctionnels ne peuvent pas être évalués[20], le groupement, soit par une somme, soit par un produit tensoriel, peuvent causer la génération de nouveaux éléments fonctionnels. Ce phénomène peut s'avérer très désavantageux, car la combinaison (somme ou multiplication) de plusieurs éléments fonctionnels peut éventuellement causer une explosion du nombre d'éléments fonctionnels dans le descripteur, ainsi que l'augmentation de la complexité des fonctions à évaluer[21]. Les effets numériques de ce phénomène sont discutés dans la présentation de mesures à la fin de cette section.

Réduction de l'Espace d'États

Finalement la réduction de l'espace d'état se manifeste lorsqu'un automate groupé $\mathcal{G}^{(k)}$ possède un nombre d'états atteignables plus petit que son espace d'états de taille $\prod_{i \in b_k} n_i$. De façon usuelle, cela est dû à l'évaluation de certaines transitions fonctionnelles à zéro ou bien à l'existence de certains états locaux incompatibles à cause d'un événement synchronisant. Après les deux pas antérieurs (élimination d'événements synchronisants et d'éléments fonctionnels), une analyse d'atteignabilité doit être faite pour calculer l'espace d'états atteignables de chaque automate groupé[22]. Si des états non-atteignables sont trouvés pour l'automate $\mathcal{G}^{(k)}$, une réduction des matrices $R_j^{(k)}$ doit être faite.

Mesures Numériques

Les mesures numériques doivent impérativement prendre en compte deux aspects fondamentaux de cette optimisation:

- la variation des temps d'exécution;
- la variation des demandes en mémoire pour le stockage du descripteur.

On peut voir cette optimisation comme des options intermédiaires entre l'utilisation d'un descripteur RAS non-groupé et la solution avec une matrice creuse équivalente. Si un RAS est groupé dans un seul automate, le résultat sera une seule matrice de transition locale, car tous les événements synchronisants et tous les éléments fonctionnels seront internes à cet unique groupe.

[20] L'évaluation d'un élément fonctionnel n'est faite que si tous les paramètres sont connus, *i.e.*, si tous les automates paramètres appartiennent à son groupe. Les évaluations partielles (avec seulement quelques paramètres, mais non sa totalité) ne sont pas considérées comme des évaluations.

[21] L'augmentation de la complexité des fonctions peut être moins importante si certaines techniques simples de calcul formel sont employées.

[22] Dans la pratique, les états atteignables d'un RAS sont connus à l'avance, permettant ainsi l'analyse d'atteignabilité d'un groupe par une simple projection.

Les exemples mesurés sont des modèles de partage de ressources (section 4.1) et des modèles de trois files d'attente avec deux classes distinctes de clients (4.3.2). Les optimisations précédentes (pré-calcul de la diagonale et re-ordonnancement d'automates) sont employés dans toutes les expériences.

Les premiers modèles mesurés sont les modèles de partage de ressources avec les paramètres suivants:

- nombre de clients $N = 12, 16, 20$;
- nombre de ressources $P = N - 1, 1$;
- nombre de groupes $B = N, N/2, N/4, 2, 1$ (pour le modèle avec $N = 12$, $N/4$ est remplacé par $N/3 = 4$).

Le tableau 5.4 présente les résultats obtenus pour ces expériences[23]. Les temps d'une multiplication sont présentés en secondes (colonne sec) et les besoins pour le stockage du descripteur sont présentés en Kilo octets (colonne Kb). Dans ce tableau, les modèles avec un nombre de ressources égal à $N - 1$ ne peuvent pas avoir un espace d'états réduit. Les modèles avec nombre de ressources égal à un ($P = 1$) sont présentés en deux versions. La première, avec groupement d'automates et réduction à l'espace d'états atteignables. La deuxième, avec groupement d'automates et sans réduction à l'espace d'états atteignables (lignes avec le symbole ❖).

Modèle	$B = 12$		$B = 6$		$B = 4$		$B = 2$		$B = 1$	
$N = 12$	sec	Kb	sec	Kb	sec	Kb	sec	Kb	sec	Kb
$P = 11$	0.8	33	0.5	33	0.2	34	0.1	50	0.0	1 087
$P = 1$ ❖	0.8	33	0.5	33	0.3	33	0.2	42	0.0	518
$P = 1$	0.8	33	0.1	6	0.0	2	0.0	1	0.0	0
Modèle	$B = 16$		$B = 8$		$B = 4$		$B = 2$		$B = 1$	
$N = 16$	sec	Kb	sec	Kb	sec	Kb	sec	Kb	sec	Kb
$P = 15$	20.8	513	12.1	514	3.3	518	1.1	593	0.4	22 527
$P = 1$ ❖	20.8	513	10.4	513	2.9	513	0.7	564	0.0	8 342
$P = 1$	20.8	513	1.6	52	0.1	52	0.0	1	0.0	1
Modèle	$B = 20$		$B = 10$		$B = 5$		$B = 2$		$B = 1$	
$N = 20$	sec	Kb	sec	Kb	sec	Kb	sec	Kb	sec	Kb
$P = 19$	496.7	8 193	283.0	8 194	73.8	8 200	19.8	8 640	—-	—-
$P = 1$ ❖	496.7	8 193	293.2	8 194	67.3	8 197	11.7	8 440	—-	—-
$P = 1$	496.7	8 193	20.7	462	0.6	25	0.0	2	0.0	1

TAB. 5.4: Résultats pour le Modèle de Partage de Ressources

L'observation de ces résultats montre que les gains en temps d'exécution peuvent être très importants dès les premiers groupements. L'augmentation des besoins en mémoire, par contre, n'est pas très prononcée avant le groupement total dans un seul bloc. En plus, ce phénomène peut être réduit par l'élimination des états non-atteignables.

[23]Tous les temps ont été mesurés avec une précision du dixième de seconde dans le logiciel PEPS (voir chapitre 7) sur une station IBM RS6000 avec système AIX 4.2 et 96 Mega octets de mémoire vive.

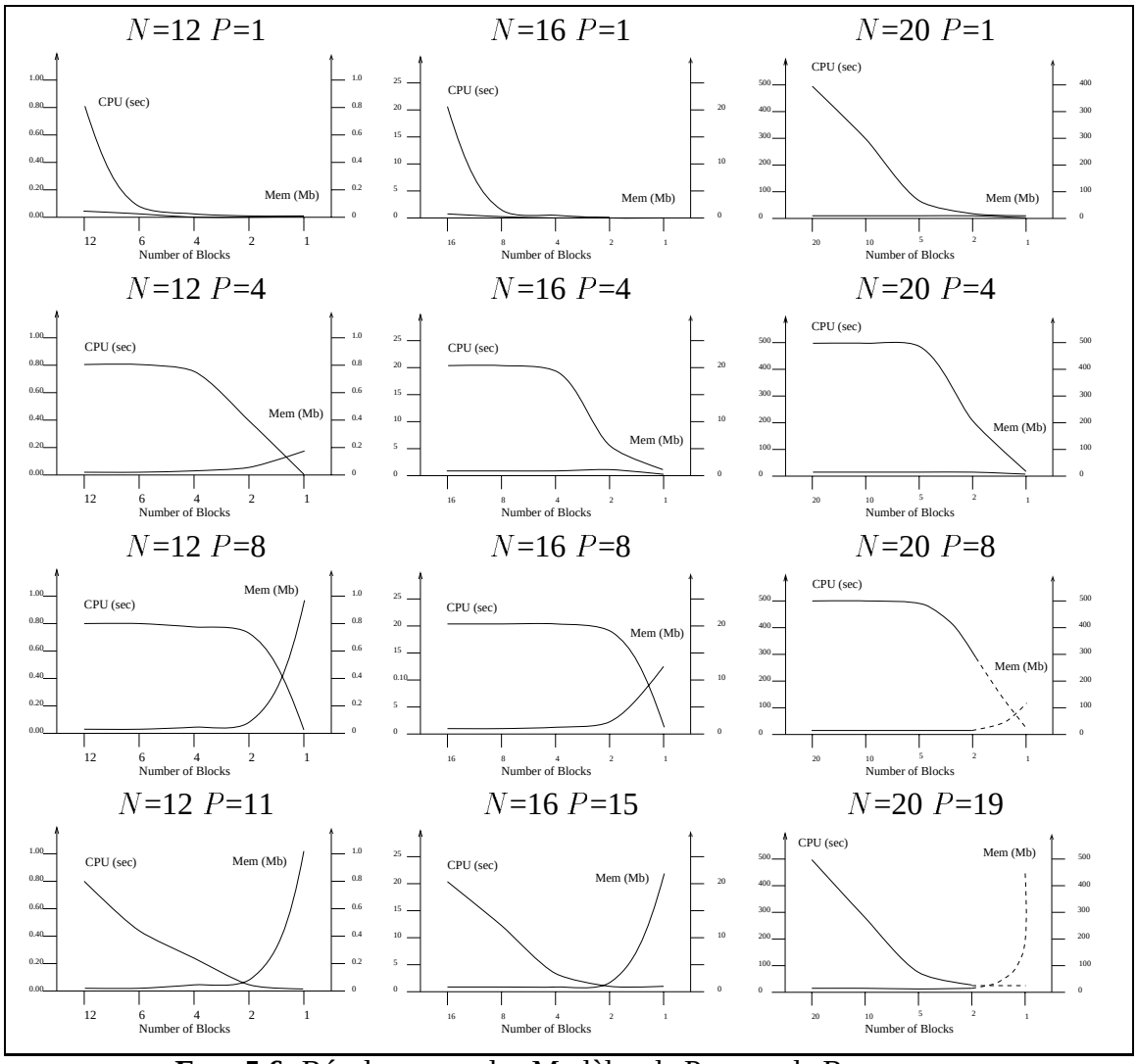


FIG. 5.6: Résultats pour les Modèles de Partage de Ressources

Intuitivement, il est intéressant de grouper les automates le plus possible avant l'explosion des besoins en mémoire. Cette approche permet d'obtenir à la fois des gains relatifs à une approche matrice creuse et les gains relatifs à une approche descripteur. On propose un deuxième jeu d'expériences avec une gamme plus large de nombre de ressources: $P = N - 1, 8, 4, 1$. La figure 5.6 montre les courbes pour ces expériences. Dans ces courbes l'axe horizontal représente la variation du nombre de blocs de N (sans groupement) jusqu'à 1 (une seule matrice creuse)[24]. Les axes verticaux indiquent à la fois les temps d'exécution en secondes et les besoins de stockage en Mega octets. Pour les modèles qui n'ont pu être générés sur notre plateforme des lignes en pointillés indiquent des estimations.

L'observation des courbes de la figure 5.6 montre clairement, dans le croisement des courbes, le nombre idéal de groupes, *i.e.*, le groupement avec des petits temps d'exécution

[24]Dans toutes les expériences le groupement inclut la réduction de l'espace d'états.

avant l'explosion des besoins de stockage. Pour les modèles très creux ($P = 1$ et $P = 4$) la multiplication par une seule matrice creuse est justifiée, car les besoins en mémoire n'arrivent jamais à poser de problème. Pour les modèles assez pleins ($P = N - 1$) on peut pousser le groupement assez loin pour obtenir des gains significatifs sans arriver à des coûts prohibitifs en mémoire.

Le point faible de cette optimisation apparaît dans les cas où quasiment aucune réduction importante du temps d'exécution peut être faite sans tomber dans un modèle trop gourmand au niveau mémoire. Ceci est le cas des modèles $N = 12$, $P = 8$ et $N = 16$, $P = 8$ où, ni l'élimination des éléments fonctionnels, ni la réduction de l'espace d'états ne suffit.

L'observation des courbes de temps d'exécution permet l'identification de certains points d'inflexion. Notamment, dans cet exemple de partage de ressources, il existe un premier point d'inflexion qui se manifeste quand les éléments fonctionnels peuvent être remplacés par des constantes. Par exemple dans un modèle avec $N = 12$ et $P = 4$, les groupes de trois automates ($B = 4$) n'éliminent pas d'éléments fonctionnels, tandis que des groupes plus grands, disons avec six automates ($B = 2$) éliminent quelques éléments fonctionnels. Un deuxième point naturel d'inflexion est quand tous les automates sont groupés, faisant disparaître alors tout besoin d'évaluation d'éléments fonctionnels et toute permutation de vecteur.

Le second type de modèle à être mesuré est le RAS décrivant un réseau de trois files d'attente visitées par deux classes de clients (section 4.3.2). La capacité des files varie selon:

- $C_1 = C_2 = 5, 10, 20$;
- $C_3 = 10, 20, 30, 50$.

Tous les modèles ayant quatre automates, on teste les possibilités de groupement selon trois possibilités:

- groupement selon les classes de client, *i.e.*, grouper l'automate $\mathcal{A}^{(1)}$ avec l'automate $\mathcal{A}^{(3_1)}$ et $\mathcal{A}^{(2)}$ avec l'automate $\mathcal{A}^{(3_2)}$;
- groupement selon les files d'attente, *i.e.*, grouper l'automate $\mathcal{A}^{(1)}$ avec l'automate $\mathcal{A}^{(2)}$ et $\mathcal{A}^{(3_1)}$ avec l'automate $\mathcal{A}^{(3_2)}$;
- groupement total des quatre automates.

Le tableau 5.5 présente les résultats pour les modèles des réseaux de trois files d'attente. Les temps d'une multiplication sont présentés en secondes (colonne sec) et les besoins pour le stockage du descripteur sont présentés en Kilo octets (colonne Kb). Dans ce tableau, tous les modèles groupés ont eu, si possible, une réduction de l'espace d'états.

Les gains des modèles de files d'attente ne sont pas aussi impressionnants que ceux des modèles de partage de ressources, mais il faut remarquer que le modèle originel possède déjà un petit nombre d'automates (quatre).

La première observation importante est que le groupement par classe de clients (colonne $(1, 3_1)(2, 3_2)$) augmente les temps d'exécution. Ce groupement élimine les événements synchronisants, mais, en évaluant les produits tensoriels des termes synchronisants, un grand nombre de nouvelles fonctions sont générées. L'évaluation des produits tensoriels peut augmenter la complexité de la multiplication à cause de la génération d'un

Modèles			$(1)(2)(3_1)(3_2)$		$(1,3_1)(2,3_2)$		$(1,2)(3_1,3_2)$		$(1,2,3_1,3_2)$	
C_1	C_2	C_3	sec	Kb	sec	Kb	sec	Kb	sec	Kb
5	5	10	0.2	21	0.7	27	0.0	16	0.0	87
5	5	20	0.2	82	0.7	94	0.0	67	0.0	376
10	10	10	0.2	81	0.9	94	0.0	63	0.0	395
10	10	20	0.7	317	3.2	346	0.2	201	0.1	1 464
10	10	30	1.6	709	10.2	754	0.5	429	0.4	3 096
10	10	50	4.5	1 963	29.3	2 039	1.4	1 153	1.2	8 156
20	20	50	17.4	7 824	88.4	7 989	6.6	4 186	—-	—-

TAB. 5.5: Résultats pour les Modèles de Trois Files d'Attente

nombre plus grand d'éléments non nuls à multiplier. En particulier l'existence d'éléments fonctionnels non éliminés, donc multipliés dans les produits tensoriels s'avère très nocive au groupement d'automates. L'élimination des événements synchronisants, décrits par produits tensoriels, peut être intéressante dans certains cas, mais une attention particulière doit être accordée à la génération des nouveaux éléments non nuls, surtout s'il s'agit d'éléments fonctionnels.

Les gains obtenus par le deuxième groupement sont dûs en partie à l'élimination des éléments fonctionnels et à la réduction de l'espace d'états. Tous les éléments fonctionnels sont dans les matrices correspondant aux automates $\mathcal{A}^{(3_1)}$ et $\mathcal{A}^{(3_2)}$ et ont ces mêmes automates comme paramètres. Ceci permet l'élimination de toutes les fonctions et par conséquence un gain énorme dans les temps d'exécution. En plus, la fonction d'atteignabilité des modèles de trois files d'attente étant, elle aussi, une fonction des automates $\mathcal{A}^{(3_1)}$ et $\mathcal{A}^{(3_2)}$, tous les états non-atteignables du modèle peuvent être éliminés.

Finalement, il est possible d'observer que les avantages apportés par un groupement total sont comparables à ceux du deuxième groupement. Par contre, les coûts en mémoire des modèles totalement groupés sont bien plus élevés. Par exemple, pour le dernier modèle ($C_1 = C_2 = 20\ C_3 = 50$) les besoins en mémoire ont excédé les possibilités de notre plateforme.

Nous avons présenté une technique efficace de traiter les produits vecteur-descripteur. Particulièrement le traitement de descripteurs avec fonctions a beaucoup progressé par rapport aux techniques utilisées antérieurement par les RAS [5]. Pour les produits tensoriels généralisés l'analyse du graphe de dépendances fonctionnelles permet, dans certains cas, de garder une complexité comparable à celle des produits tensoriels classiques. De plus les techniques d'optimisation réduisent, voir éliminent, l'impact des éléments fonctionnels dans le coût de multiplication d'un descripteur. Tout ceci nous permet de confirmer que les méthodes itératives sont les méthodes les plus adéquates pour les générateurs infinitésimaux disponibles dans le format d'un descripteur RAS.

Chapitre 6

Méthodes Itératives de Résolution des RAS

Dans ce chapitre on décrit les méthodes itératives pour obtenir une solution stationnaire pour les RAS. Puisque le descripteur d'un RAS est un générateur infinitésimal, la solution stationnaire des RAS se ramène à l'une des solutions d'un système linéaire de rang $n-1$ (n étant la taille du problème)[1]. La solution qui nous intéresse est un vecteur de probabilités, donc normalisée de façon à ce que la somme des éléments soit égale à un. Le système linéaire à résoudre est:

$$xQ = 0$$

Les méthodes de résolution présentées ici ont été implementées dans le logiciel PEPS (voir chapitre 7) d'où tous les résultats numériques ont été pris. Les méthodes présentées dans ce chapitre sont:

- Méthode de la puissance;
- Méthode d'Arnoldi;
- Méthode GMRES;

L'opération de base de toutes les méthodes est la multiplication d'un vecteur de probabilités par un descripteur RAS. Selon le chapitre 5, il est possible de multiplier un vecteur v quelconque par un descripteur sans calculer explicitement la matrice de transition[2]. De manière générale, toutes les méthodes de résolution appliquées aux matrices en format explicite (creux ou plein) peuvent aussi être employées pour les matrices en format descripteur.

[1] Toutes les solutions d'un système linéaire homogène de rang $n-1$ sont égales à une constante multiplicative près.

[2] On entend comme format explicite de stockage pour une matrice un format où une définition explicite (non-algébrique) de chaque élément est faite. Les formats explicites habituels sont les formats plein (un tableau bi-dimensionel de nombres réels) ou un format creux (un format particulier où seulement les éléments non nuls et leur positions sont stockées). Le format utilisé pour le descripteur RAS n'est pas explicite, car il s'agit d'une formule incluant les sommes traditionnelles et les produits tensoriels de matrices.

Après la présentation des méthodes de résolution, on discute l'application des techniques de pré-conditionnement pour ces méthodes. Les techniques employées pour générer une matrice de pré-conditionnement, donc pour approcher un inverse du descripteur, dérivent de l'utilisation de:

– la diagonale du descripteur comme une approximation du descripteur;

– une décomposition LU de parties choisies du descripteur selon un format tensoriel;

– l'approximation de l'inverse du descripteur par un polynôme.

La prochaine section décrit les méthodes itératives de base (puissance, Arnoldi et GMRES). Ensuite les techniques de pré-conditionnement sont décrites et leur adaptation à la solution des RAS est présentée. Quelques mesures numériques sur des exemples pratiques sont présentées pour illustrer les gains apportés par les méthodes proposées. Ces résultats ont été extraits de l'annexe B qui présente la totalité des expériences avec chacune des méthodes et leurs variantes pour un ensemble d'exemples.

6.1 Méthodes Itératives de Base

Les méthodes de base, non pré-conditionnées, sont présentées ici. Les techniques particulières à chaque méthode, ainsi que des considérations pratiques de leur implémentation dans le cadre du logiciel PEPS (chapitre 7) sont présentés. Les références de base pour les méthodes présentés dans cette section sont les livres de Saad [93], Stewart [100] et Lascaux/Théodor [59].

6.1.1 Méthode de la Puissance

La méthode de la puissance [59, 100] est la solution itérative la plus simple pour un système linéaire définie par:

$$0 = xQ$$

Cette solution doit être précédé de l'uniformisation du descripteur, *i.e.*, la transformation d'un générateur infinitésimal Q en matrice stochastique P. Le descripteur doit être normalisé et une matrice identité doit être ajoutée:

$$P = I_{n_Q} + \frac{Q}{\alpha}$$

où α est le plus grand élément en module du descripteur Q.

Cette transformation du système linéaire à résoudre correspond aux équivalences:

$$\begin{aligned} 0 &= xQ \\ \Longleftrightarrow 0 &= x\frac{Q}{\alpha} \\ \Longleftrightarrow x &= x + x\frac{Q}{\alpha} \\ \Longleftrightarrow x &= x(I_{n_Q} + \frac{Q}{\alpha}) \end{aligned}$$

Ce système est résolu par la méthode de la puissance avec le schéma itératif suivant:

$$x^{(k+1)} = x^{(k)} \left(I_{n_Q} + \frac{Q}{\alpha} \right)$$

où $x^{(k)}$ représente la solution après la k-ème itération.

L'algorithme 6.1 représente l'application de la méthode de la puissance pour un descripteur Q en partant d'un vecteur initial v et en arrivant à la solution x.

Algorithme 6.1

```
1  x^(0) = v
2  Q = I_nQ + Q/α
3  for k = 0, 1, ... ∞
4  do multiply   x^(k+1) = x^(k) Q
5      if  convergence test
6          stop                    (goto 8)
7  end do
8  x = x^(k+1)
```

Algorithme 6.1: Méthode de la Puissance

Le test de convergence exprimé dans la ligne 4 de l'algorithme 6.1 peut être exécuté de plusieurs façons. L'option la moins coûteuse est la comparaison entre les valeurs des vecteurs $x^{(k)}$ et $x^{(k+1)}$ qui correspondent aux résultats de la dernière et de l'avant-dernière itération. Cependant, cette option n'est pas forcément intéressante pour des modèles avec une convergence très lente. Une discussion complète des possibilités est présenté dans le chapitre 3 du livre de Stewart [100]. Dans le logiciel PEPS (chapitre 7), nous avons adopté une comparaison simple des vecteurs des deux dernières itérations, et selon le choix de l'utilisateur une multiplication additionnelle est exécutée pour savoir la valeur exacte du résidu de la solution trouvée. Ceci permet de savoir si le vecteur obtenu est la solution selon la précision acceptée, ou si la méthode a stagne.

Considérations d'Implémentation

La normalisation du descripteur correspond à la recherche de la valeur α et à la division des matrices du descripteur par cette valeur. La recherche de la plus grande valeur (α) est faite en comparant tous les éléments de la diagonale du descripteur. La division des matrices doit être faite pour chaque terme de l'addition du descripteur, *i.e.*:

$$\frac{Q}{\alpha} = \frac{D}{\alpha} + \sum_{j=1}^{(N+E)} \frac{\bigotimes_{g_i=1}^{N} \bar{Q}_j^{(i)}}{\alpha}$$

Dans la pratique, la normalisation doit diviser par α toutes les matrices locales $Q_l^{(i)}$ et une matrice de chaque terme synchronisant $Q_e^{(i)}$[3]. La normalisation des matrices locales

[3]Il faut rappeler que seulement un terme synchronisant, celui des matrices positives, de chaque événement est retenu après le pré calcul de la diagonale

représente la normalisation d'une matrice de chaque facteur normal (terme de la somme tensorielle). La normalisation d'une parmi les matrices de chaque terme synchronisant représente la normalisation d'une matrice de chaque produit tensoriel.

Il faut remarquer que le format du descripteur permet une optimisation du schéma itératif pour la méthode de la puissance. Après le pré-calcul de la diagonale (section 5.3.1), le descripteur est composé de deux facteurs: les éléments diagonaux en un format vecteur et les éléments non diagonaux en un format tensoriel. Le schéma d'itération est égal à:

$$x^{(k+1)} = x^{(k)} \left(I_{n_Q} + \frac{D}{\alpha} + \sum_{j=1}^{(N+E)} \frac{\bigotimes_{g\,i=1}^{N} \bar{Q}_j^{(i)}}{\alpha} \right)$$

Il est possible d'ajouter la matrice identité avec la diagonale normalisée dans un seul vecteur avant l'application de la méthode de la puissance. Ceci évite une somme de vecteurs dans le schéma d'itération.

La résolution d'un RAS avec la méthode de la puissance peut demander trois vecteurs de probabilité. À chaque pas d'itération un vecteur initial ($x^{(k)}$) doit être gardé pour multiplier chacun des termes produit tensoriel qui sont additionnés dans le descripteur. Un vecteur doit accumuler le résultat ($x^{(k+1)}$) de chaque multiplication par un terme produit tensoriel. Un troisième vecteur temporaire doit garder les pas intermédiaires de la multiplication de chaque facteur normal où un produit tensoriel est décomposé. Il faut remarquer que ce vecteur temporaire n'est pas nécessaire pour les descripteurs sans événements synchronisants. En effet, ces descripteurs n'ayant qu'une somme tensorielle, ne font pas de multiplication par des termes produit tensoriel (décomposables en plusieurs facteurs normaux).

6.1.2 Méthode d'Arnoldi

La méthode d'Arnoldi [4, 93, 100] est une méthode de projection orthogonale sur un sous-espace engendré par un espace de Krylov [113] de taille m, noté $\mathcal{K}_m$. Un espace de Krylov $\mathcal{K}_m$ est défini par une matrice carré A et un vecteur v_1 de norme 1:

$$\mathcal{K}_m(A, v_1) = span\left\{v_1, v_1A, v_1A^2, v_1A^3, \ldots, v_1A^{m-1}\right\}$$

Le plus grand avantage de la méthode d'Arnoldi est d'accélérer la convergence (par rapport à la méthode de la puissance) en utilisant seulement des multiplications vecteur-matrice. Chacun des vecteurs de l'espace de Krylov est généré par des multiplications successives d'un vecteur initial v par la matrice A. Ces vecteurs sont stockés dans des vecteurs appelés v_j avec $j \in [1..m]$, ***i.e.***:

$$v_1; \quad v_2 = v_1A; \quad v_3 = v_1A^2 = v_2A; \quad \ldots \quad v_m = v_1A^{m-1} = v_{m-1}A;$$

✍ **Soit**

v_j le j-ème vecteur de l'espace $\mathcal{K}_m(A, v_1)$;

V_m la matrice de dimension $n_A \times m$ où chaque colonne j est le vecteur v_j ($j \in [1..m]$) de l'espace $\mathcal{K}_m(A, v_1)$.

La méthode d'Arnoldi est utilisée pour deux types de problèmes:

– la détermination des plus grandes valeurs propres (et leur vecteurs propres correspondants);

– la solution des systèmes linéaires du type[4] $xA = b$.

Dans une solution itérative des RAS, nous sommes intéressés par un cas particulier de la deuxième option, où $b = 0$. Dans ce chapitre nous allons employer la méthode d'Arnoldi pour approcher la solution du système linéaire:

$$xA = 0 \quad \text{ou} \quad x = x + xA.$$

Prenons H_m une matrice carrée de dimension m avec tous les éléments non nuls situés au dessus de la deuxième diagonale inférieure, *i.e.*, une matrice d'Hessenberg supérieure [73]. Les éléments non nuls d'une matrice d'Hessenberg supérieure sont les éléments $h_{i,j}$ où $i \in [1..min(j+1, m)]$ et $j \in [1..m]$. Par exemple:

$$H_6 = \begin{pmatrix} h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} & h_{1,5} & h_{1,6} \\ h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} & h_{2,5} & h_{2,6} \\ 0 & h_{3,2} & h_{3,3} & h_{3,4} & h_{3,5} & h_{3,6} \\ 0 & 0 & h_{4,3} & h_{4,4} & h_{4,5} & h_{4,6} \\ 0 & 0 & 0 & h_{5,4} & h_{5,5} & h_{5,6} \\ 0 & 0 & 0 & 0 & h_{6,5} & h_{6,6} \end{pmatrix}$$

La méthode d'Arnoldi est fondée sur la création d'une matrice H_m qui représente la transformation linéaire A pour l'espace $\mathcal{K}_m(A, v_1)$ avec la base V_m, *i.e.*, $H_m = V_m^T A V_m$. De cette façon, les valeurs et les vecteurs propres de H_m (en général une matrice plus petite que A) peuvent être des approximations aux valeurs et aux vecteurs propres de A.

Nous allons décrire la méthode d'Arnoldi pour résoudre le système $xA = 0$ à travers de trois étapes:

– le choix d'un vecteur initial v_1 de norme 1;

– la construction de l'espace $\mathcal{K}_m(A, v_1)$ et de la matrice H_m;

– la détermination de la solution approchée.

Le premier pas de la méthode d'Arnoldi est l'obtention du vecteur v_1 (de norme 1) utilisé pour construire l'espace $\mathcal{K}_m(A, v_1)$. Partons d'une estimation initiale de la solution exprimé par le vecteur v. Le résidu initial β est obtenu par:

$$\beta = \|vA\|_2$$

Le vecteur v_1 est le vecteur initial de résidu normalisé, *i.e.*:

$$v_1 = \frac{vA}{\beta}$$

[4]La solution des systèmes $xA = b$ et $Ax = b$ sont analogues. Dans ce document seulement le premier cas est montré, pourtant tous les techniques et méthodes présentées sont applicables aux deux cas (avec les adaptations nécessaires selon algèbre matricielle).

Le deuxième pas de la méthode d'Arnoldi est le calcul des vecteurs v_j ($j \in [1..m]$), par des multiplications successives. Ensuite, chaque vecteur v_j est orthonormalisé par rapport à tous les vecteurs v_i déjà calculés ($i \in [1..j-1]$). L'orthogonalisation est exécutée par une procédure de Gram-Schmidt [43, 97] et en même temps les éléments non nuls de la matrice H_m sont définis.

La construction des vecteurs v_j orthonormaux et de la matrice H_m est implémentée par l'algorithme 6.2. Cet algorithme a comme paramètres:

- v_1, un vecteur initial de norme 1;
- A, la matrice représentant le système à résoudre;
- m, la taille de l'espace de Krylov.

Si lors de l'orthonormalisation, un des vecteurs v_j devient nul, la méthode doit s'arrêter (ligne 9 de l'algorithme 6.2). Or, si tel est le cas, la combinaison linéaire des vecteurs de l'espace $\mathcal{K}_j(A, v_1)$ (un espace plus petit que $\mathcal{K}_m(A, v_1)$) est déjà la solution exacte recherchée. La littérature [93] appelle cette situation de *lucky breakdown*, car l'algorithme n'est plus applicable à cause d'une solution exacte trouvée par hasard.

Algorithme 6.2

```
1   for j = 1, 2, ... , m
2   do  multiply    w = v_j A
3       for i = 1, 2, ... , j
4       do  h_{i,j} = v_i^T w          (produit scalaire)
5           w = w - h_{i,j} v_i
6       end do
7       h_{j+1,j} = ||w||_2
8       if    h_{j+1,j} = 0
9             stop                     (sortie de la boucle)
10      else
11            v_{j+1} = w / h_{j+1,j}
12  end do
```

Algorithme 6.2: Arnoldi - Génération de l'espace $\mathcal{K}_m(A, v_1)$ et de la matrice H_m

Le troisième pas de la méthode d'Arnoldi est la détermination de la solution approchée d'après le vecteur initial v et les informations générés par la procédure d'Arnoldi (algorithme 6.2). La solution approchée est obtenue en calculant le résidu à être ajouté orthogonal au espace V_m. Ceci est fait en déterminant le vecteur y du système:

$$H_m y = (\beta e_1)$$

où e_1 est le vecteur canonique $(1, 0, 0, \ldots, 0, 0)$ et β est la norme du résidu initial ($\|vA\|_2$). Ensuite la solution approchée x est calculée comme la combinaison linéaire des vecteurs v_j (matrice V_m) selon la formule:

$$x = v + yV_m$$

L'algorithme 6.3 présente la méthode d'Arnoldi appliquée à la solution d'un descripteur de RAS Q en partant d'un vecteur initial quelconque v et qui calcule le vecteur x solution de $0 = xQ$. La littérature spécialisée [93, 100] utilise la dénomination de méthode d'orthogonalisation totale pour la méthode d'Arnoldi appliquée à la solution des systèmes linéaires[5].

Algorithme 6.3

```
1   multiply    w = vQ
2   β = ‖w‖_2
3   normalize   v_1 = w/β          (le vecteur v_1 est de norme 1)
4   for j = 1, 2, ... , m
5   do  multiply    w = v_j Q
6       for i = 1, 2, ... , j
7       do  h_{i,j} = w v_i^T      (produit scalaire)
8           w = w − h_{i,j} v_i
9       end do
10      h_{j+1,j} = ‖w‖_2
11      if  h_{j+1,j} = 0
12          stop                   (goto 16)
13      else
14          v_{j+1} = w/h_{j+1,j}
15  end do
16  solve       y = H^{-1}(βe_1)
17  x = v + y V_m
```

Algorithme 6.3: Méthode d'Orthogonalisation Totale - Méthode d'Arnoldi

Considérations d'Implémentation - Orthogonalisation des vecteurs

La procédure de Gram-Schmidt [43] utilisée pour orthogonaliser les vecteurs v_j ($j \in [1..m]$) dans les algorithmes 6.2 et 6.3 est une version modifié de la procédure standard. Cette version modifiée présente une plus grande robustesse que la procédure de Gram-Schmidt standard, cependant même cette version modifiée présente certains problèmes d'ordre numérique [51]. L'implémentation pratique de la méthode d'Arnoldi peut devenir plus robuste d'un point de vue numérique, si l'algorithme *Householder* [110] est employé. Cette procédure est basée sur une décomposition QR de la matrice V_i, la matrice de dimension $n_A \times i$ composée des vecteurs colonnes v_j avec $j \in [1..i]$. Ceci permet une orthogonalisation moins sensible à l'accumulation des erreurs d'arrondi, mais d'un coût de calcul légèrement supérieur. La description de l'algorithme *Householder* et de la procédure de Gram-Schmidt ainsi que leur utilisation dans la méthode d'Arnoldi sont

[5]Rappelons que la méthode d'Arnoldi peut aussi être employée pour la détermination de plus grandes valeurs propres et vecteurs propres correspondants. Le lecteur peut trouver des détails de cette application dans la section 4.4 du livre de Stewart [100].

décrites en détail dans la section 6.3 du livre de Saad [93]. Dans le logiciel PEPS (chapitre 7), nous avons implémenté la méthode d'Arnoldi avec la procédure Gram-Schmidt modifiée, car cette option représente le meilleur rapport entre la robustesse et le coût de calcul.

Considérations d'Implémentation - Arnoldi avec Redémarrage

D'un point de vue pratique, la méthode d'orthogonalisation totale présente des sérieuses limitations à cause de la taille m de l'espace de Krylov ($\mathcal{K}_m(A, v_1)$). La complexité de calcul due à la procédure de Gram-Schmidt est d'ordre de m^2n (orthogonalisation de m vecteurs de taille n). L'utilisation mémoire est de l'ordre de mn (stockage de m vecteurs de taille n). L'option naturelle pour réduire la taille de l'espace de Krylov est d'exécuter des redémarrages périodiques de la procédure. Ceci correspond à appliquer la méthode d'Arnoldi pour un nombre m de pas et ensuite utiliser la solution obtenue comme un vecteur initial d'un nouvel ensemble de m pas. L'algorithme 6.4 montre la modification en conséquence de l'algorithme 6.3. Dans l'algorithme 6.4 on appelle la procédure d'Arnoldi comme décrit dans l'algorithme 6.2. Dans le logiciel PEPS (chapitre 7), nous avons implémenté la méthode d'Arnoldi exclusivement avec redémarrages.

Algorithme 6.4

```
1  multiply    w = vQ
2  β = ‖w‖_2
3  normalize   v_1 = w/β            (le vecteur v_1 est de norme 1)
4  for k = 1, 2, ... ∞
5  do call     Arnoldi              (algorithme 6.2)
6     solve    y = H^{-1}(βe_1)
7     x = v + yV_m
8     if       convergence test
9        stop                       (sortie de la boucle)
10    else
11       multiply   w = xQ
12       β = ‖w‖_2
13       normalize  v_1 = w/β
14 end do
```

Algorithme 6.4: Méthode d'Arnoldi avec Redémarrage

6.1.3 Méthode GMRES

La méthode GMRES - *Generalized Minimum Residual Method* [92, 100, 43] est aussi une méthode de projection sur un espace de Krylov [113]. De façon identique à la méthode d'Arnoldi, un espace de Krylov $\mathcal{K}_m(A, v_1)$ est engendré et une matrice H_m est calculée. Les mêmes considérations à propos des procédures d'orthogonalisation (Gram-Schmidt

et Householder) peuvent être faites. La seule différence entre la méthode d'Arnoldi et la méthode GMRES se situe dans le calcul de la solution approchée.

La solution dans la méthode d'Arnoldi (sans redémarrage) est exprimée par la ligne 16 et 17 de l'algorithme 6.3:

$$\begin{aligned} y &= H^{-1}(\beta e_1) \\ x &= v + yV_m \end{aligned}$$

Dans la méthode d'Arnoldi, le vecteur y qui nous amène à la solution est choisi de façon à ce que le résidu (yV_m) reste orthogonal à tous les vecteurs de l'espace $\mathcal{K}_m(A, v_1)$ (condition de Galerkin). En opposition, la méthode GMRES cherche un vecteur y de façon à minimiser le résidu (yV_m). Ceci reviens à dire que la méthode GMRES minimise l'équation:

$$\|xQ\|_2$$

Ceci est fait en remplaçant la ligne 16 de l'algorithme 6.3 par une procédure qui trouve le vecteur y qui minimise l'équation:

$$\left\|\beta e_1 - \bar{H}_m y\right\|_2 \tag{6.1}$$

où $\bar{H}_m$ est la matrice de dimension $(m+1) \times m$ obtenue en ajoutant à H_m la ligne:

$$[0, 0, \ldots, 0, h_{m+1,m}].$$

L'algorithme 6.5 présente l'implémentation de la méthode GMRES de façon analogue à la méthode d'orthogonalisation totale (algorithme 6.3).

Considérations d'Implémentation - Minimisation

Le point clé de l'algorithme 6.5 est le calcul du vecteur y qui minimise $\|\beta e_1 - \bar{H}_m y\|_2$, *i.e.*, la recherche d'une solution au sens des moindres carrés [16]. Une solution pour ce problème est proposée dans le livre de Saad [93] (section 6.5). Cette solution consiste à transformer la matrice $\bar{H}_m$ en une matrice triangulaire supérieure à l'aide de rotations de Givens [43]. Ces rotations correspondent à des multiplications par une matrice G_i de rang 2 de la forme[6]:

$$G_i = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & c_i & s_i & & & \\ & & & -s_i & c_i & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{pmatrix}$$

[6]Les valeurs de c_i et s_i sont choisies de façon à éliminer l'élément $h_{i+1,i}$.

Algorithme 6.5

1 **initialize** $w = vQ$
2 $\beta = \|w\|_2$
3 **normalize** $v_1 = w/\beta$ *(le vecteur v_1 est de norme 1)*
4 **for** $j = 1, 2, \ldots, m$
5 **do multiply** $w = v_j Q$
6 **for** $i = 1, 2, \ldots, j$
7 **do** $h_{i,j} = w v_i^T$ *(produit scalaire)*
8 $w = w - h_{i,j} v_i$
9 **end do**
10 $h_{j+1,j} = \|w\|_2$
11 **if** $h_{j+1,j} = 0$
12 **stop** *(goto 16)*
13 **else**
14 $v_{j+1} = w/h_{j+1,j}$
15 **end do**
16 **solve** $y \in \mathbb{R}^{n_Q} \min \|\beta e_1 - \bar{H}_m y\|_2$
17 $x = v + y V_m$

Algorithme 6.5: Méthode GMRES Totale - sans redémarrages

Par exemple, l'élimination du premier élément sous diagonal d'une matrice H_m ($h_{2,1}$) correspond à multiplier la matrice $\bar{H}_m$ et le vecteur βe_1 par la matrice de rotation G_1.

$$G_1 = \begin{pmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}$$

$$\text{où} \quad s_1 = \frac{h_{2,1}}{\sqrt{(h_{1,1})^2 + (h_{2,1})^2}} \quad \text{et} \quad c_1 = \frac{h_{1,1}}{\sqrt{(h_{1,1})^2 + (h_{2,1})^2}}$$

Le résultat obtenu est:

$$G_1 \times \bar{H}_m = \begin{pmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \cdots \\ 0 & h_{2,2} & h_{2,3} & \cdots \\ 0 & h_{3,2} & h_{3,3} & \cdots \\ 0 & 0 & h_{4,3} & \cdots \\ 0 & 0 & 0 & \ddots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad G_1 \times \beta e_1 = \begin{pmatrix} c_i \beta \\ -s_i \beta \\ 0 \\ \vdots \end{pmatrix}$$

Après une matrice G_2 est engendrée pour éliminer l'élément $h_{3,2}$ et ainsi de suite jusqu'à ce que la matrice $\left(\prod_{i=m}^{1} G_i\right) \bar{H}_m$ devienne une matrice triangulaire supérieure à

laquelle on ajoute une ligne nulle:

$$\left(\prod_{i=m}^{1} G_i\right) \bar{H}_m = \begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,m-1} & h_{1,m} \\ 0 & h_{2,2} & \cdots & h_{2,m-1} & h_{2,m} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & h_{m-1,m-1} & h_{m-1,m} \\ 0 & 0 & \cdots & 0 & h_{m,m} \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

Ceci permet la solution au sens des moindres carrés par des remontés (*backward substitutions* - solution d'une matrice triangulaire supérieure) de l'équation

$$\left\| \left(\prod_{i=m}^{1} G_i\right) \beta e_1 - \left(\prod_{i=m}^{1} G_i\right) \bar{H}_m y \right\|_2 = 0.$$

Or, le même vecteur y qui minimise l'équation 6.1 minimise l'équation

$$\left\| \left(\prod_{i=m}^{1} G_i\right) \beta e_1 - \left(\prod_{i=m}^{1} G_i\right) \bar{H}_m y \right\|_2$$

De plus, cette technique permet la résolution d'un problème pratique important, la détermination d'un point d'arrêt pour la méthode. Pendant la génération des vecteurs v_j (boucle sur j - lignes 4 à 15 de l'algorithme 6.5) aucune solution intermédiaire n'est connue. Le calcul des matrices G_i (en fait le calcul de c_i et s_i seulement) peut être fait simultanément avec le calcul des éléments de la matrice H_m. Ceci permet à toute itération sur j d'obtenir la norme du résidu avec un coût relativement bas (solution d'un système triangulaire).

Considérations d'Implémentation - GMRES avec Redémarrage

À l'instar de la méthode d'Arnoldi, la méthode de GMRES peut aussi être utilisée avec redémarrage. Le principe est toujours le même, c'est à dire, réduire les coûts de traitement et d'utilisation de mémoire en réduisant le nombre m. L'algorithme 6.6 présente l'implémentation de la méthode GMRES avec redémarrage.

La méthode GMRES totale (algorithme 6.5) arrive sûrement à une solution exacte en n (la taille du problème) itérations. La méthode GMRES avec redémarrage, au contraire, peut stagner, *i.e.*, peut ne plus réduire la norme du résidu. Ce phénomène de stagnation est le seul désavantage apporté par l'utilisation des redémarrages. Cependant, des valeurs de m proches de la limite théorique de la méthode GMRES totale (n) ne sont pas applicables pour des cas pratiques car les besoins de temps de calcul et place mémoire sont prohibitifs. Dans le logiciel PEPS (chapitre 7), nous avons implémenté la méthode de GMRES exclusivement avec redémarrages.

Algorithme 6.6

1 **multiply** $w = vQ$
2 $\beta = \|w\|_2$
3 **normalize** $v_1 = w/\beta$ *(le vecteur v_1 est de norme 1)*
4 **for** $k = 1, 2, \ldots \infty$
5 **do** **call** Arnoldi *(algorithme 6.2)*
6 **solve** $y \in \underset{min}{\mathbb{R}^{n_Q}} \|\beta e_1 - \bar{H}_m y\|_2$
7 $x = v + yV_m$
8 **if** ***convergence test***
9 **stop** *(sortie de la boucle)*
10 **else**
11 **multiply** $w = xQ$
12 $\beta = \|w\|_2$
13 **normalize** $v_1 = w/\beta$
14 **end do**

Algorithme 6.6: Méthode GMRES avec Redémarrage

6.1.4 Mesures Numériques

Les gains apportés pour les méthodes d'Arnoldi et GMRES se manifestent aussi bien sur le nombre d'itérations que sur le temps. Il faut noter, par contre, que leur utilisation en mémoire est beaucoup plus importante. En effet, la différence des besoins de place mémoire des méthodes d'Arnoldi et GMRES par rapport à ceux de la méthode de la puissance est de l'ordre de $m \times n_Q$, où m est le nombre de vecteurs intermédiaires définissant l'espace de Krylov et n_Q est la taille de l'espace d'états du modèle RAS. Nous pouvons observer ces gains sur tous les exemples présentés dans l'annexe B. Le tableau 6.1 montre un extrait de l'annexe B avec les nombres d'itérations nécessaires à chacune des méthodes (colonne *it.*) et les temps d'exécution jusqu'à convergence (colonne *sec.*). Nous prendrons pour ce tableau des exemples génériques pour essayer de couvrir les principaux phénomènes de l'application des méthodes d'Arnoldi et GMRES. Les méthodes qui n'ont pas convergée (stagnation) ont le symbole "—". Les meilleures valeurs obtenus (en temps de résolution et en nombre d'itérations) sont soulignés en gras.

Prenons trois exemples de modèles de partage de ressource (voir section 4.1):

- $N = 12, P = 12, B = 2$ - 12 clients, 12 ressources, groupés en 2 blocs et taux d'allocation et libération distincts pour tous les clients (exemple 1.2.1 de l'annexe B);
- $N = 12, P = 11, B = 2$ - 12 clients, 1, ressources, groupés en 2 blocs et taux d'allocation et libération distincts pour tous les clients (exemple 1.4.1 de l'annexe B);
- $N = 12, P = 8, B = 2$ - 12 clients, 8 ressources, groupés en 2 blocs et taux d'allocation et libération distincts pour tous les clients (exemple 1.6.1 de l'annexe B).

Modèles de Partage de Ressources										
	Puissance		Arnoldi $m = 10$		Arnoldi $m = 30$		GMRES $m = 10$		GMRES $m = 30$	
Exemples	it.	sec.	it.	sec.	it.	sec.	it.	sec.	it.	sec.
1.2.1	662	13.2	100	2.6	**62**	**2.4**	107	2.8	66	2.5
1.4.1	668	35.2	106	6.5	68	**5.0**	107	6.9	**66**	**5.0**
1.6.1	679	434.2	117	78.0	78	52.8	116	85.8	**73**	**47.8**
1.8.1	978	513.8	142	76.9	87	51.9	130	81.1	**77**	**45.7**
Modèles de Réseau Fermé avec Trois Files - 10 clients										
	Puissance		Arnoldi $m = 10$		Arnoldi $m = 30$		GMRES $m = 10$		GMRES $m = 30$	
Exemples	it.	sec.	it.	sec.	it.	sec.	it.	sec.	it.	sec.
2.1	1010	6.6	126	1.0	**76**	**0.7**	147	1.1	77	**0.7**
2.2	457	3.0	100	0.8	58	**0.5**	93	0.8	**57**	0.6
2.3	528	3.5	104	0.8	62	**0.6**	95	0.8	**60**	**0.6**
2.4	723	4.8	117	0.9	**62**	**0.6**	99	0.8	**62**	**0.6**
Modèles de Réseau Ouvert avec Trois Files										
	Puissance		Arnoldi $m = 10$		Arnoldi $m = 30$		GMRES $m = 10$		GMRES $m = 30$	
Exemples	it.	sec.	it.	sec.	it.	sec.	it.	sec.	it.	sec.
3.1.2	476	10.5	182	**5.2**	**177**	7.3	—	—	288	11.8
3.2.1	1459	36.2	349	11.9	**246**	12.1	330	**11.3**	279	13.6
3.2.2	496	12.6	**218**	**7.5**	267	13.0	—	—	298	14.2
3.3.2	525	13.9	**172**	**6.5**	180	9.7	—	—	234	12.1

TAB. 6.1: Applications des Méthodes de Base sans Pré-conditionnement

– $N = 12, P = 4, B = 2$ - 12 clients, 4 ressources, groupés en 2 blocs et taux d'allocation et libération distincts pour tous les clients (exemple 1.8.1 de l'annexe B);

Prenons quatre exemples de modèles de réseau fermé avec trois files et 10 clients (voir section 4.2):

– *routage équilibré* - probabilités de routage et taux de service définis pour avoir environ le même rapport entre le taux de visite et le temps de service de toute les files (exemple 2.1 de l'annexe B);

– *surcharge à la file* s_2 - probabilités de routage et taux de service définis pour concentrer les clients dans la deuxième file (exemple 2.2 de l'annexe B);

– *surcharge à la file* s_3 - probabilités de routage et taux de service définis pour concentrer les clients dans la troisième file (exemple 2.3 de l'annexe B);

– *faible charge à la file* s_3 - probabilités de routage définies pour concentrer les clients dans les deux premières files (exemple 2.4 de l'annexe B);

Prenons trois exemples de modèles de réseau ouvert avec trois files (voir section 4.3.2):

– $C_1 = 7, C_2 = 9, C_3 = 11$ - file s_1 avec 0 à 6 clients, file s_2 avec 0 à 8 clients, file s_3 avec 0 à 10 clients et taux d'arrivée et service provoquant la saturation de la première file (exemple 3.1.2 de l'annexe B);

– $C_1 = 8, C_2 = 8, C_3 = 12$ - files s_1 et s_2 avec 0 à 7 clients, file s_3 avec 0 à 11 clients et taux d'arrivée et service provoquant un distribution presque uniforme des clients entre les files (exemple 3.2.1 de l'annexe B);

– $C_1 = 8, C_2 = 8, C_3 = 12$ - files s_1 et s_2 avec 0 à 7 clients, file s_3 avec 0 à 11 clients et taux d'arrivée et service provoquant la saturation de la première file (exemple 3.2.2 de l'annexe B);

– $C_1 = 10, C_2 = 10, C_3 = 10$ - files s_1, s_2 et s_3 avec 0 à 9 clients et taux d'arrivée et service provoquant la saturation de la première file (exemple 3.3.2 de l'annexe B);

Le tableau 6.1 illustre les gains apportés par les méthodes d'Arnoldi et GMRES en comparaison avec la méthode de la puissance. Si le gain en temps de calcul entre ces méthodes et la méthode de la puissance est très clair, aucune affirmation peut être faite pour établir à priori laquelle de ces méthodes sera la plus performante dans un cas pratique.

Un deuxième fait à remarquer est que la taille de l'espace de Krylov n'est pas toujours corrélée avec l'accélération. Dans le tableau 6.1 nous pouvons remarquer que pour le modèle $C_1 = 8, C_2 = 8, C_3 = 12$ l'accélération est plus importante avec un espace de taille 10 ($m = 10$) qu'avec un espace de taille 30 ($m = 30$). Le tableau 6.2 montre les accélérations obtenues pour cinq modèles distincts utilisant des tailles de l'espace de Krylov allant de 2 jusqu'à 30. Seulement le nombre d'itérations est indiqué pour donner une idée de l'accélération obtenue. Il va de soit que les temps par itération augmentent au fur et à mesure qu'une taille plus grande est utilisée. Cependant cette augmentation n'est pas très significative pour ces exemples[7]. La première ligne du tableau indique le nombre d'itérations obtenues avec la méthode de la puissance.

Dans ce tableau nous reprenons deux modèles de partage de ressources:

– $N = 12, P = 12, B = 2$, taux d'allocation et libération distincts pour tous les clients (exemple 1.2.1 de l'annexe B);

– $N = 12, P = 4, B = 2$, taux d'allocation et libération distincts pour tous les clients (exemple 1.8.1 de l'annexe B).

Puis, deux modèles de réseau fermé de files d'attente:

– *routage équilibré* (exemple 2.1 de l'annexe B);

– *surcharge à la file* s_3 (exemple 2.3 de l'annexe B).

Et deux modèles de réseau ouvert de files d'attente:

– $C_1 = 8, C_2 = 8, C_3 = 12$ - taux distincts d'arrivée et service (exemple 3.2.1 de l'annexe B);

– $C_1 = 8, C_2 = 8, C_3 = 12$ - taux distincts d'arrivée et service (exemple 3.2.2 de l'annexe B).

L'analyse du tableau 6.2 permet de confirmer que l'accélération de la convergence n'est pas forcément corrélée avec la taille de l'espace de Krylov. Des fluctuations entre nombres de m voisins apparaissent dans tous les exemples. De plus, le cas de la méthode d'Arnoldi pour l'exemple 3.2.2 montre un comportement qui pourrait être expliqué par une augmentation de l'instabilité numérique au fur et à mesure de l'augmentation de m. L'accumulation des erreurs d'arrondi dans l'orthogonalisation des vecteurs génère des

[7] Pour tous les exemples, la plus grande augmentation du temps par itération trouvée entre un espace de taille 2 et 30 est de l'ordre de moins d'un dixième.

	Exemple 1.2.1		Exemple 1.8.1		Exemple 2.1		Exemple 2.3		Exemple 3.2.1		Exemple 3.2.2	
	662 it.		978 it.		1010 it.		528 it.		1459 it.		496 it.	
m	Arnoldi	GMRES	Arnoldi	GMRES	Arnoldi	GMRES	Arnoldi	GMRES	Arnoldi	GMRES	Arnoldi	GMRES
2	–	–	–	284	–	–	–	–	–	–	–	–
3	387	350	612	186	821	657	237	167	–	–	–	–
4	184	144	214	172	506	476	152	136	994	692	346	–
5	164	113	220	175	294	260	118	124	728	650	247	–
6	125	134	152	150	186	224	129	110	615	420	194	–
7	130	119	129	161	210	184	91	111	404	413	193	–
8	122	110	172	157	138	152	111	115	394	344	247	–
9	114	108	124	142	132	156	115	124	376	306	195	–
10	100	107	142	130	126	147	104	95	349	330	218	–
11	96	96	134	131	136	136	87	98	283	330	222	–
12	99	93	110	119	131	120	84	105	282	288	184	–
13	90	91	109	114	106	130	97	85	288	311	**172**	–
14	86	86	109	117	115	130	93	88	268	308	203	–
15	79	83	107	115	107	118	103	97	267	270	180	–
16	88	75	103	105	106	108	111	88	239	276	189	–
17	83	72	104	105	107	98	84	90	214	291	206	–
18	82	74	100	101	102	93	85	88	244	306	246	–
19	82	78	91	91	108	87	88	85	256	266	209	–
20	82	78	93	90	90	95	85	82	234	273	222	–
21	81	77	98	82	94	88	83	81	256	**240**	212	–
22	82	73	87	82	95	95	80	76	259	257	228	–
23	82	76	96	82	95	88	81	85	**191**	262	220	–
24	78	77	94	88	87	92	76	78	237	298	210	–
25	74	72	94	90	83	93	73	73	256	272	198	399
26	75	67	91	90	84	82	73	73	228	282	184	323
27	75	68	**81**	84	81	**75**	72	72	218	269	179	339
28	74	70	92	81	82	81	65	65	226	271	198	299
29	70	67	86	**76**	82	80	67	61	242	255	194	**260**
30	**62**	**66**	87	77	**76**	77	**62**	**60**	246	279	267	298

TAB. 6.2: Vitesse de Convergence par rapport à la Taille de l'Espace de Krylov

bruits dans la convergence parce que les valeurs numériques des vecteurs orthonormalisés ne sont pas suffisamment précis.

Un deuxième point à observer est que même des petites valeurs de m peuvent donner de très bonnes accélérations. Notamment nous avons eu le cas de la méthode GMRES appliquée à l'exemple 1.8.1 avec $m = 2$ (réduction à un tiers des itérations). Il est possible aussi de ne pas avoir de convergence avec une méthode avant d'utiliser des grandes valeurs de m. Ceci est le cas de la méthode GMRES appliquée à l'exemple 3.2.2 (réduction à 80% des itérations avec $m = 25$).

6.2 Pré-conditionnement

Les techniques de pré-conditionnement [59, 93, 100, 109] ont pour but d'accélérer la convergence des méthodes itératives. Le principe de base du pré-conditionnement d'un système linéaire du type

$$xA = b \tag{6.2}$$

est de le multiplier à droite les deux membres de l'équation par une matrice dite de pré-conditionnement (M^{-1}):

$$xAM^{-1} = bM^{-1} \tag{6.3}$$

La matrice M étant une matrice inversible et assez proche de la matrice A, l'équation 6.3 a, en général, une converge plus rapide que l'équation 6.2. Si la matrice M est identique à la matrice A, le système converge en une seule itération [100].

Appelons le pré-conditionnement obtenu en multipliant à droite les deux membres de l'équation le *pré-conditionnement externe*. Or, un type de pré-conditionnement analogue consiste à résoudre le système obtenu en multipliant à gauche la matrice A et le vecteur b:

$$xM^{-1}A = M^{-1}b$$

Pour certains cas pratiques ce type de pré-conditionnement, appelé le *pré-conditionnement interne*[8], peut s'avérer plus intéressante. Pourtant, nous n'allons pas discuter cette possibilité qui est généralement plus coûteuse à traiter que le pré-conditionnement externe [93]. Tous les développements dans la suite de cette thèse font référence exclusivement au pré-conditionnement externe.

6.2.1 Pré-conditionnement des Méthodes de Base

Les deux difficultés fondamentales de la technique de pré-conditionnement appliquée aux méthodes de base sont:

- choisir une bonne matrice M, c'est à dire, une matrice qui peut accélérer la convergence du schéma pré-conditionné;
- trouver une façon peu coûteuse pour calculer l'inverse de la matrice M.

Regardons d'abord le deuxième point et laissons la discussion du choix de la matrice M dans la section 6.2.2. Théoriquement le calcul de l'inverse de la matrice M est aussi coûteux que la solution du système elle-même. Cependant, si on regarde le schéma d'itération pré-conditionné on remarque que le calcul de l'inverse peut être évité. Prenons, par exemple la méthode de la puissance. Dans le cadre des RAS, le système à résoudre est tel que le vecteur b est nul et la matrice A est exprimée par le descripteur Q:

$$xQ = 0 \tag{6.4}$$

De cette façon, le pré-conditionnement se fait par l'équation

$$xQM^{-1} = 0 \tag{6.5}$$

[8]Dans la solution de systèmes linéaires du type $Ax = b$, la littérature appelle le pré-conditionnement externe *pré-conditionnement à gauche* [93], car il est obtenu en multipliant le système à gauche: $M^{-1}Ax = M^{-1}b$. De façon analogue le pré-conditionnement interne ($AM^{-1}x = bM^{-1}$) est appelé *pré-conditionnement à droite*. Nous n'avons pas retenu cette nomenclature car nos systèmes sont du type $xA = 0$.

Donc, le schéma d'itération pré-conditionné pour la méthode de la puissance est

$$x^{(k+1)} = x^{(k)} + x^{(k)} Q M^{-1}$$

Les opérations à faire sont:

- la multiplication du vecteur $x^{(k)}$ par la matrice Q, dont le résultat est un vecteur appelé y;
- la multiplication du vecteur y par l'inverse de la matrice M.

La technique de base consiste à remplacer la matrice M par une décomposition[9] LU:

$$x^{(k+1)} = x^{(k)} + y(LU)^{-1}$$

$$x^{(k+1)} = x^{(k)} + yU^{-1}L^{-1}$$

Sachant que les matrices L et U sont des matrices triangulaires (inférieure et supérieure respectivement), il est possible de remplacer ces multiplications par des résolutions de systèmes d'équations. La multiplication de y par U^{-1} est remplacée par la résolution du système

$$zU = y$$

où $z = yU^{-1}$. Ceci est fait par un algorithme classique de solution en remontée (*backward substitution*) [59, 85]. De façon analogue, la multiplication zL^{-1} est résolue par une solution en descente (*forward substitution*).

La méthode de la puissance pré-conditionnée est implémentée par l'algorithme 6.7, qui est une version modifiée de l'algorithme non pré-conditionné (algorithme 6.1). Il est à remarquer l'inclusion de la ligne 5, qui représente la multiplication du vecteur $y = x^{(k)}Q$ par l'inverse de la matrice M, implémentée par des résolutions de systèmes triangulaires.

Les pré-conditionnements de la méthode d'Arnoldi et de la méthode GMRES se font dans les mêmes termes que ceux de la méthode de la puissance. Les opérations de multiplication par le descripteur Q sont suivies des opérations de multiplication par l'inverse de la matrice M.

Les algorithmes 6.8 et 6.9 implémentent les versions pré-conditionnées des méthodes d'Arnoldi et GMRES avec redémarrage. Les versions non pré-conditionnées de ces deux méthodes sont décrites par les algorithmes 6.4 et 6.6.

6.2.2 Choix de la Matrice de Pré-conditionnement

L'autre problème de taille du pré-conditionnement est le choix de la matrice M qui peut accélérer la convergence. La matrice Q, un générateur infinitésimal, n'est pas inversible, donc nous sommes obligés de trouver une matrice M suffisamment proche de Q.

[9] La décomposition LU d'une matrice M quelconque consiste à obtenir deux matrices triangulaires, une supérieure (U) et une inférieure (L) de façon à ce que le produit matriciel $L \times U$ soit égal à la matrice M [85].

Algorithme 6.7

1 $x^{(0)} = v$
2 $Q = I_{n_Q} + Q/\alpha$
3 **for** $k = 0, 1, \ldots \infty$
4 **do multiply** $y = x^{(k)}\ Q$
5 **solve** $x^{(k+1)} = y\ M^{-1}$
6 **if** ***convergence test***
7 **stop** *(goto 9)*
8 **end do**
9 $x = x^{(k)}$

Algorithme 6.7: Méthode de la Puissance Pré-conditionnée

Le concept de proximité, en ce qui concerne le pré-conditionnement, n'est pas absolu et plusieurs heuristiques peuvent être adoptées. Pour les cas des matrices stockées dans un format creux standard [100] un certain nombre d'heuristiques sont largement employées. Il existe notamment les algorithmes de décomposition LU incomplète [43, 93, 100], selon des critères qui réduisent le nombre d'éléments non nuls des matrices L et U. Les trois politiques usuelles de choix des éléments des matrices L et U à garder sont:

- $ILU0$, les éléments non nuls des matrices L et U ne sont gardés que si leurs correspondants dans la matrice à décomposer Q étaient des éléments non nuls;
- $ILUk$, seulement les k plus grands éléments de chaque ligne des matrices L et U sont gardés;
- $ILUTH$, seulement les éléments des matrices L et U plus grands qu'un seuil sont conservés.

Le descripteur RAS ne peut pas utiliser directement ces options, car son format de stockage n'est pas un format creux standard, mais une formulation en terme de somme de produits tensoriels[10]. Néanmoins, un produit tensoriel $A = \otimes_{i=1}^{N} A^{(i)}$ peut profiter de ces approximations (décompositions LU incomplètes). En effet, la propriété de compatibilité du produit tensoriel avec la multiplication traditionnelle des matrices (équation 2.7, page 21) et la propriété d'associativité (équation 2.19) impliquent que:

$$A = \otimes_{i=1}^{N} A^{(i)} = \otimes_{i=1}^{N} \left[L^{(i)} U^{(i)}\right] = \otimes_{i=1}^{N} \left[L^{(i)}\right] \otimes_{i=1}^{N} \left[U^{(i)}\right] \tag{6.6}$$

✍ **Avec**

$L^{(i)}$ la matrice L d'une décomposition LU de la matrice $A^{(i)}$ d'une suite finie des matrices;

$U^{(i)}$ la matrice U d'une décomposition LU de la matrice $A^{(i)}$ d'une suite finie des matrices.

[10] Les définitions des descripteurs des RAS sont faites dans le chapitre 3.

Algorithme 6.8

```
1   multiply    y = vQ
2   solve       w = yM^{-1}
3   β = ‖w‖_2
4   normalize   v_1 = w/β                    (le vecteur v_1 est de norme 1)
5   for k = 1, 2, ... ∞
6   do  for j = 1, 2, ... , m
7       do  multiply    y = v_j A
8           solve       w = yM^{-1}
9           for i = 1, 2, ... , j
10          do  h_{i,j} = v_i^T w            (produit scalaire)
11              w = w − h_{i,j} v_i
12          end do
13          h_{j+1,j} = ‖w‖_2
14          if  h_{j+1,j} = 0
15              stop                         (goto 19)
16          else
17              v_{j+1} = w/h_{j+1,j}
18      end do
19      solve       y = H^{-1}(βe_1)
20      x = v + yV_m
21      if  convergence test
22          stop                             (sortie de la boucle)
23      else
24          multiply    y = xQ
25          solve       w = yM^{-1}
26          β = ‖w‖_2
27          normalize   v_1 = w/β
28  end do
```

Algorithme 6.8: Méthode d'Arnoldi Pré-conditionnée avec Redémarrage

Il faut remarquer que le produit tensoriel de deux matrices triangulaires supérieures est aussi une matrice triangulaire supérieure (idem pour les matrices triangulaires inférieures). Ceci nous permet de décomposer un produit tensoriel $A = \otimes_{i=1}^{N} A^{(i)}$ en décomposant ses matrices $A^{(i)}$.

De plus, un algorithme semblable à celui de la multiplication d'un vecteur par un produit tensoriel (algorithme 5.4, page 99) peut être employé pour effectuer la multiplication d'un vecteur quelconque v par l'inverse d'un terme tensoriel triangulaire ($\otimes_{i=1}^{N} \left[L^{(i)}\right]$ ou $\otimes_{i=1}^{N} \left[U^{(i)}\right]$). Le principe de base reste la résolution de chacun des systèmes triangulaires par des tranches du vecteur v (voir section 5.1.2).

Nous allons donc présenter dans les sections suivantes trois options pour déterminer une matrice M^{-1} capable d'accélérer la convergence. Les options adoptées sont:

Algorithme 6.9

```
1  multiply   y = vQ
2  solve      w = yM^{-1}
3  β = ||w||_2
4  normalize  v_1 = w/β                    (le vecteur v_1 est de norme 1)
5  for k = 1, 2, ... ∞
6  do  for j = 1, 2, ... , m
7      do  multiply   y = v_j A
8          solve      w = yM^{-1}
9          for i = 1, 2, ... , j
10         do  h_{i,j} = v_i^T w            (produit scalaire)
11             w = w − h_{i,j} v_i
12         end do
13         h_{j+1,j} = ||w||_2
14         if   h_{j+1,j} = 0
15             stop                         (goto 19)
16         else
17             v_{j+1} = w/h_{j+1,j}
18     end do
19     solve      y ∈ min_{R^{n_Q}} ||βe_1 − H̄_m y||_2
20     x = v + yV_m
21     if   convergence test
22         stop                             (sortie de la boucle)
23     else
24         multiply   y = xQ
25         solve      w = yM^{-1}
26         β = ||w||_2
27         normalize  v_1 = w/β
28 end do
```

Algorithme 6.9: Méthode GMRES Pré-conditionnée avec Redémarrage

– utiliser une approximation polynômiale de l'inverse du descripteur;

– utiliser une combinaison des décompositions LU des termes tensoriels du descripteur;

– utiliser l'inverse de la diagonale du descripteur.

Pré-conditionnement Polynômial

Le premier type de pré-conditionnement est basé sur une approche de l'inverse du générateur Q par un polynôme. L'inverse d'une matrice de la forme $(I_{n_Q} - Q)$ est donné

par la série[11] [62]:

$$(I_{n_Q} - Q)^{-1} = I_{n_Q} + Q + Q^2 + \cdots + Q^{(k-1)} + \cdots$$

Puisque nous cherchons une matrice M^{-1} qui est proche de Q^{-1}, nous allons chercher l'inverse de

$$Q \equiv I_{n_Q} - (I_{n_Q} - Q)$$

Ceci nous donne le polynôme:

$$\left[I_{n_Q} - (I_{n_Q} - Q)\right]^{-1} = I_{n_Q} + (I_{n_Q} - Q) + (I_{n_Q} - Q)^2 + \cdots + (I_{n_Q} - Q)^{(k-1)} + \cdots$$

Le choix du nombre k de termes du polynôme nous amène à un compromis entre la précision de l'inverse et le coût de calcul. En principe plus grand sera le nombre de termes, meilleur sera le pré-conditionnement. Le coût de l'opération de pré-conditionnement correspondra à k multiplications vecteur descripteur, *i.e.*, aussi cher que k pas de la méthode de puissance non pré-conditionnée.

L'implémentation du *pré-conditionnement polynômial* pour la méthode de la puissance (algorithme 6.7) correspond à remplacer la ligne 5 (opération de pré-conditionnement) par:

$$x^{(k+1)} = y\left[I_{n_Q} + (I_{n_Q} - Q) + (I_{n_Q} - Q)^2 + \cdots + (I_{n_Q} - Q)^{(k-1)}\right]$$

La qualité du pré-conditionnement polynômial peut être évaluée de façon théorique en regardant les courbes correspondant aux valeurs propres de la la matrice pré-conditionné:

$$Q \times \left[I_{n_Q} + (I_{n_Q} - Q) + (I_{n_Q} - Q)^2 + \cdots + (I_{n_Q} - Q)^{(k-1)}\right]$$

Ces valeurs propres (λ') peuvent être exprimées en fonction des valeurs propres de la matrice Q, appelées λ, par la formule:

$$\lambda' = \lambda\left[1 + (1 - \lambda) + (1 - \lambda)^2 + \cdots + (1 - \lambda)^{(k-1)}\right]$$

Cette remarque est particulièrement importante pour la méthode de la puissance, car la vitesse de convergence de cette méthode est inversement proportionnelle au module de la deuxième plus grande valeur propre[12]. La figure 6.1 représente les courbes obtenus pour des polynômes de plusieurs tailles. Ces courbes sont à comparer avec le pré-conditionnement théoriquement parfait où l'inverse exacte de Q pourra être utilisée. Il est clair que Q étant un générateur, donc une matrice singulière, il n'existe pas d'inverse exacte de la matrice Q. Pourtant cette matrice représenterait le pré-conditionneur parfait, car les valeurs propres de la matrice pré-conditionné QQ^{-1} (λ_p) seraient exprimés en fonction des valeurs propres de Q par:

$$\lambda_p = \frac{\lambda}{\lambda} = 1$$

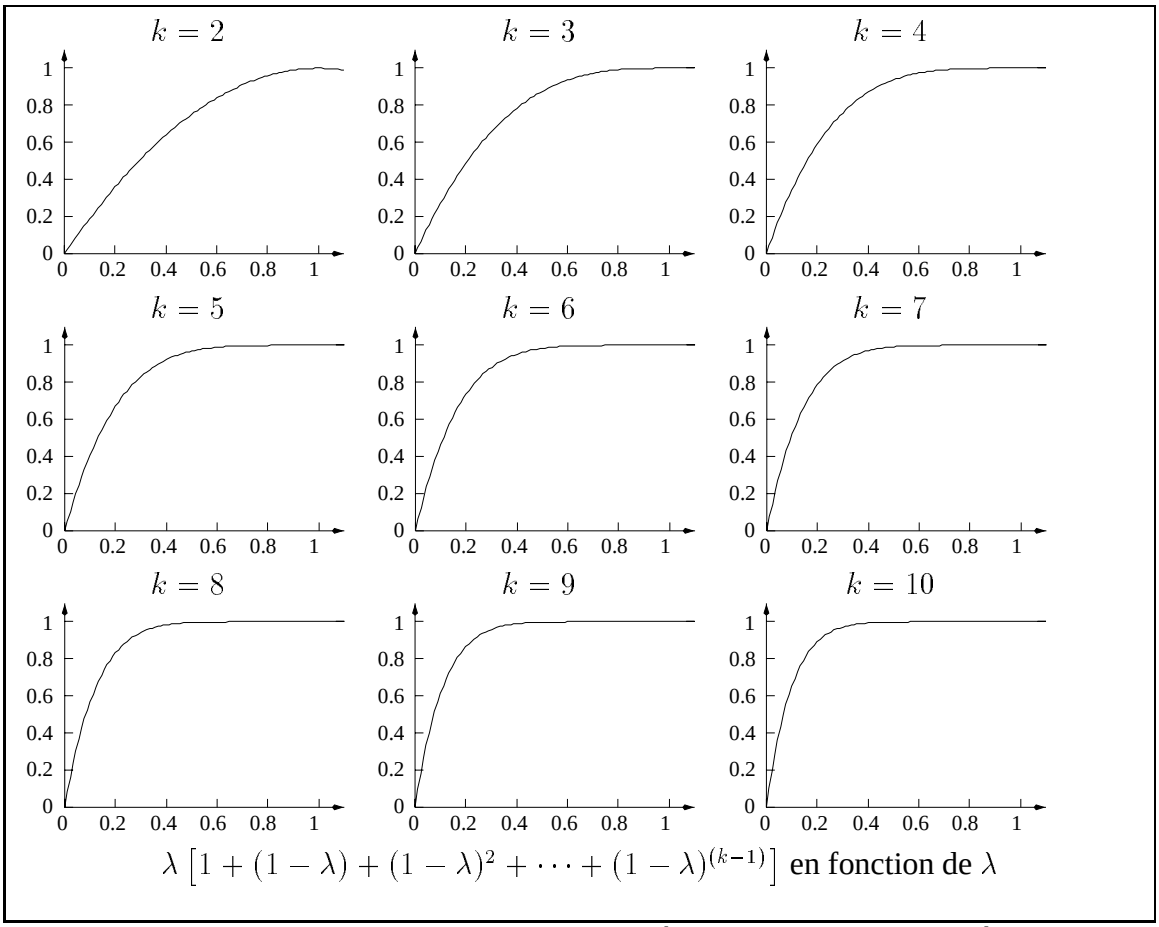


FIG. 6.1: Rapport entre les Valeurs propres du Pré-conditionnement Polynômial par Inverse Approchée

Les courbes de la figure 6.1 permettent d'observer l'effet de l'approximation obtenue en prenant un polynôme avec un nombre plus grand de termes. Une interprétation plus intéressante est que même pour des polynômes très petits ($k = 2$) on peut avoir des gains pour des valeurs propres très proches de 1. Ce type de pré-conditionnement sera appelé *pré-conditionnement par inverse approchée*.

Malheureusement, cette formulation théorique de recherche d'une matrice suffisamment proche de l'inverse du descripteur n'apporte pas forcément de bons résultats. En effet, pour la totalité des exemples testés (annexe B) ce pré-conditionnement n'a pas accéléré la convergence.

Une alternative au pré-conditionnement par inverse approchée, toujours polynômiale, consiste à utiliser un polynôme moins complexe que

$$I_{n_Q} + (I_{n_Q} - Q) + (I_{n_Q} - Q)^2 + \cdots + (I_{n_Q} - Q)^{(k-1)} + \cdots \tag{6.7}$$

[11]Cette formulation de l'inverse est vrai si et seulement si le rayon spectral de Q est inférieur à 1 [93], ce qui est le cas d'un générateur infinitésimal normalisé.

[12]Rappelons que la plus grand valeur propre du générateur infinitésimal Q est égale à 1.

Par exemple, une heuristique utilisée [93] est le pré-conditionnement par

$$I_{n_Q} + Q + Q^2 + \cdots + Q^{(k-1)} \tag{6.8}$$

Cette heuristique permet l'obtention d'un polynôme moins coûteux à multiplier. La multiplication d'un vecteur par le polynôme 6.7 implique en k multiplications vecteur-descripteur et $2k-1$ additions de vecteur, tandis que la multiplication par le polynôme 6.8 implique en k multiplications vecteur-descripteur et k additions de vecteur.

Bien que ce choix polynômial puisse sembler naïf, il est possible de justifier cette approximation par le théorème de Cayley-Hamilton [62], selon lequel l'inverse d'une matrice A peut être défini par la formule

$$A^{-1} = \frac{a_1}{-a_0} I_{n_A} + \frac{a_2}{-a_0} A + \frac{a_3}{-a_0} A^2 + \cdots + \frac{a_{n_A-1}}{-a_0} A^{n_A-2} + A^{n_A-1}$$

où a_i sont les indices du polynôme caractéristique minimal $\chi(\lambda)$ de A exprimé par

$$\chi(\lambda) = \sum_{i=1}^{n_A} a_i \lambda^i = det\left[\lambda I_{n_A} - A\right]$$

Le fait à remarquer, pour le pré-conditionnement, est que les facteurs $\frac{a_i}{-a_0}$ sont des scalaires, donc on peut avoir une approximation grossière de l'inverse de la matrice A par

$$A^{-1} \sim I_{n_A} + A + A^2 + \cdots + A^{k-1} + \cdots + A^{n_A-1}$$

où nous limitons ce polynôme à k termes. Ce type de pré-conditionnement sera appelé *pré-conditionnement translaté*, car il correspond aussi à une translation du rayon spectral de Q [59][13].

Bien que l'analyse des valeurs propres, à l'instar du pré-conditionnement par inverse approchée, n'indique pas une bonne approximation de l'inverse (figure 6.2), les tests pratiques ont montré quelques accélérations avec ce type de pré-conditionnement.

L'observation des gains apportés par le pré-conditionnement polynômial translaté peut être vérifiée dans les exemples de réseaux ouverts suivants[14]:

- modèle de partage de ressources $N = 12, P = 12, B = 2$, taux de requête et libération de ressources distincts pour chaque client (exemple 1.4.1 dans l'annexe B);
- modèle $C_1 = 7, C_2 = 9, C_3 = 11$ avec distribution presque uniforme des clients par les files (exemple 3.1.1 dans l'annexe B);
- modèle $C_1 = 7, C_2 = 9, C_3 = 11$ avec saturation de la première file (exemple 3.1.2 dans l'annexe B);
- modèle $C_1 = 8, C_2 = 8, C_3 = 12$ avec saturation de la première file (exemple 3.2.2 dans l'annexe B);

[13] En effet le polynôme 6.8 est une approximation de l'inverse de $(I_{n_Q} - Q)$.

[14] Les modèles sont définis selon la section 4.3.2 (page 79)et les jeux de paramètres sont introduites dans l'annexe B.

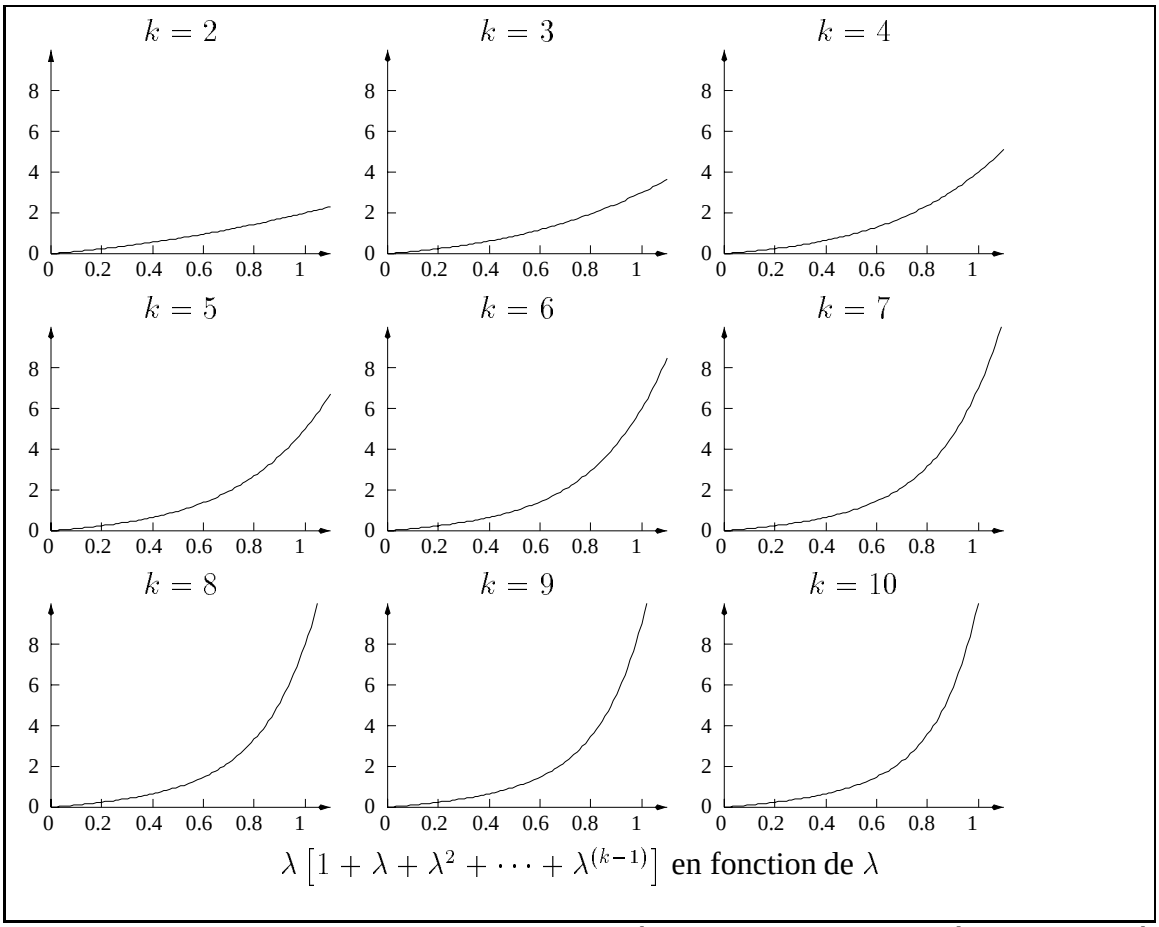


FIG. 6.2: Rapport entre les Valeurs propres du Pré-conditionnement Polynômial Translaté

Le tableau 6.3 montre le nombre d'itérations nécessaires à chacune des méthodes (colonne *it.*) et les temps d'exécution jusqu'à la convergence (colonne *sec.*) utilisant des tailles distinctes de polynôme (k). Notons que la première ligne ($k = 1$) correspond à l'utilisation non pré-conditionnée, car le polynôme utilisé correspond seulement à la matrice identité I_{n_Q}. Nous avons choisi les valeurs $k = 3, 5, 7$, car ni les valeurs paires, ni les valeurs supérieurs à 7 n'ont jamais présenté des accélérations. À l'instar des résultats précédents, nous soulignons en gras les meilleurs accélérations (réduction du nombre d'itérations et des temps d'exécution). Le symbole "—" indique les méthodes stagnées ou ralenties par le pré-conditionnement.

Le premier modèle (exemple 1.4.1) présente des réductions dans le nombre d'itérations dans toutes les méthodes. Malheureusement ces réductions n'arrivent pas à compenser l'augmentation du temps par itération. Le deuxième modèle (exemple 3.1.1) permet des réductions plus importantes dans le nombre d'itérations, qui ne sont pas encore suffisantes.

C'est dans les deux derniers modèles (exemple 3.1.2 et 3.2.2) que les réductions se manifestent aussi sur les temps d'exécution, notamment dans les méthodes utilisant un grand espace de Krylov ($m = 30$). Pour la méthode de GMRES nous observons aussi que le pré-conditionnement permet la convergence avec un espace petit ($m = 10$), ce

Exemple 1.4.1										
	Puissance		Arnoldi $m = 10$		Arnoldi $m = 30$		GMRES $m = 10$		GMRES $m = 30$	
k	it.	sec.	it.	sec.	it.	sec.	it.	sec.	it.	sec.
1	668	**35.2**	106	**6.5**	68	**5.0**	107	**6.9**	66	**5.0**
3	580	100.5	79	13.9	53	9.9	84	14.3	54	9.9
5	**511**	135.8	**78**	20.7	**51**	14.7	**78**	20.6	**51**	14.7
7	–	–	106	38.9	59	23.7	98	36.0	65	25.9
Exemple 3.1.1										
	Puissance		Arnoldi $m = 10$		Arnoldi $m = 30$		GMRES $m = 10$		GMRES $m = 30$	
k	it.	sec.	it.	sec.	it.	sec.	it.	sec.	it.	sec.
1	1258	**27.8**	283	**7.6**	216	**9.0**	280	**8.1**	233	**9.8**
3	**1062**	70.9	**160**	11.6	**133**	11.2	**180**	12.8	**133**	11.2
5	–	–	395	45.5	303	38.7	410	46.8	282	35.9
7	–	–	1009	158.0	699	118.8	1100	172.2	690	117.1
Exemple 3.1.2										
	Puissance		Arnoldi $m = 10$		Arnoldi $m = 30$		GMRES $m = 10$		GMRES $m = 30$	
k	it.	sec.	it.	sec.	it.	sec.	it.	sec.	it.	sec.
1	476	**10.5**	182	**5.2**	177	7.3	–	–	288	11.8
3	**314**	21.0	**109**	7.9	**80**	**6.5**	**148**	**10.6**	**80**	**6.7**
5	–	–	165	18.9	144	18.1	180	20.6	138	17.5
7	–	–	–	–	–	–	–	–	–	–
Exemple 3.2.2										
	Puissance		Arnoldi $m = 10$		Arnoldi $m = 30$		GMRES $m = 10$		GMRES $m = 30$	
k	it.	sec.	it.	sec.	it.	sec.	it.	sec.	it.	sec.
1	496	**12.6**	218	**7.5**	267	13.0	–	–	298	14.2
3	**295**	23.7	**100**	8.8	**85**	**8.4**	**134**	**11.5**	**82**	**8.2**
5	–	–	199	27.0	156	23.5	184	25.1	172	25.9
7	–	–	–	–	–	–	–	–	–	–

TAB. 6.3: Applications des Méthodes avec Pré-conditionnement Polynômial Translaté

qui n'était pas possible avant. Cependant, il faut observer que l'application de la méthode d'Arnoldi non pré-conditionnée avec un espace petit ($m = 10$) reste toujours l'option la plus rapide (plus petit temps d'exécution).

Le deuxième phénomène remarquable est la mauvaise qualité de ce type de pré-conditionnement pour des grandes valeurs de k. Nous pouvons aussi constater par l'analyse théorique des valeurs propres de la matrice pré-conditionnée (figure 6.2) que l'utilisation de polynômes longs n'est pas forcément intéressante. À ceci, nous ajoutons le fait que les grands polynômes augmentent de façon prohibitive le coût par itération. Ce pré-conditionnement ne semble performant que pour des petits valeurs de k[15].

[15]Cette conclusion qui exclu les pré-conditionnement polynômiaux avec beaucoup de termes été déjà observée dans [5].

Pré-conditionnement avec Décomposition LU des Termes Tensoriels avec des Fonctions

Comme nous l'avons vu (équation 6.6), chaque terme produit tensoriel du descripteur peut être décomposé en matrices L et U décrites sous un format tensoriel. Dans cette section nous allons reprendre le format du descripteur (équation 3.2) avant l'extraction des éléments diagonaux, car l'enlèvement des éléments diagonaux des matrices locales empêche la décomposition LU des matrices locales comme il sera vu dans la suite. Nous rappelons la définition du descripteur d'un RAS faite dans le chapitre 3, où N est le nombre d'automates et E est le nombre d'événements synchronisants:

$$Q = \sum_{j=1}^{(N+2E)} \bigotimes_{i=1}^{N}{}_{g} Q_j^{(i)}$$

Trois problèmes peuvent restreindre la décomposition LU des termes tensoriels et leur utilisation dans le pré-conditionnement.

- la décomposition des matrices fonctionnelles;
- les matrices non-décomposables (et matrices non-inversibles);
- les matrices singulières (systèmes à plusieurs solutions).

Matrices Fonctionnelles La première restriction à cette décomposition individuelle de chaque terme tensoriel est l'existence d'éléments fonctionnels dans les matrices locales. Étant donné que nous cherchons une matrice approchée, il est possible d'utiliser des heuristiques d'élimination d'éléments fonctionnels. Cinq possibilités sont testées:

- remplacer des éléments fonctionnels par zéro;
- remplacer les éléments fonctionnels par une de leurs évaluations choisie par l'utilisateur;
- remplacer les éléments fonctionnels par leur plus grande évaluation;
- remplacer les éléments fonctionnels par leur plus petite évaluation;
- remplacer les éléments fonctionnels par leur évaluation moyenne (moyenne algébrique de toutes les évaluations).

Les expériences numériques montrent que ces options sont semblables, avec un très léger avantage à l'utilisation de l'évaluation moyenne. Cependant, il est très clair que pour un meilleur pré-conditionnement il est nécessaire de promouvoir la réduction, voire l'élimination, des éléments fonctionnels à travers des techniques comme le groupement d'automates (voir section 5.3.3).

Matrices Non-Décomposables La deuxième restriction à la décomposition individuelle de chaque terme tensoriel est la structure des matrices locales. La décomposition LU de ces matrices n'est pas toujours possible. Notamment lors de l'application des algorithmes

traditionnels [85] il est possible de trouver un élément diagonal nul[16]. La solution la plus souvent employée pour éviter ce genre de problème est la *régulation* des matrices avant leur décomposition. Les techniques usuelles de régulation transforment les matrice locales, gardant certaines de leurs propriétés. Nous avons choisi deux heuristiques usuelles de régulation [93]:

- la translation, qui consiste à ajouter une petite valeur à tous les éléments diagonaux de la matrice (*shift*);
- la transformation de Winglet, qui consiste à remplacer la valeur de chacun des éléments diagonaux par le plus grand (en module) élément diagonal.

Les techniques de pré-conditionnement utilisant les décompositions LU, auront toujours trois variantes selon l'utilisation des régulations:

- sans régulation (appelée *nor*);
- régulation avec translation (appelée *shift*);
- régulation avec transformation de Winglet (appelée *winglet*).

En l'absence de régulation, une décomposition LU peut être invalidée et donc le terme tensoriel ne sera pas pris en compte pour le pré-conditionnement. Nos implémentations de pré-conditionnement avec décomposition LU auront une classification des termes tensoriels selon la qualité de sa décomposition LU. Les termes où toutes les matrices ont pu être décomposées correctement sont classés comme valides et sont considérés pour le pré-conditionnement. Les termes où au moins une des matrices n'a pas pu être correctement décomposée sont classés invalides et ne sont pas pris en compte lors du pré-conditionnement.

Matrices Singulières La troisième restriction est le cas des matrices singulières où nous pouvons obtenir une décomposition LU correcte mais nous ne pouvons pas résoudre le système linéaire représenté par les matrices triangulaires obtenues. C'est le cas des matrices de la partie locale du descripteur qui, dans leur forme originelle, sont toujours des générateurs infinitésimaux[17]. La décomposition LU d'un générateur infinitésimal produit une matrice U avec le dernier élément diagonal nul[18]. Ceci nous permet de toujours résoudre le système triangulaire représenté par la matrice L, mais la résolution du système représenté par la matrice U n'est soluble qu'à une constante près. Un problème similaire se retrouve dans la solution stationnaire des chaînes de Markov [100], où la décomposition LU du générateur infinitésimal doit être résolue. Dans ce cas spécifique n'importe

[16] L'impossibilité de faire la décomposition LU d'une matrice avec tous les éléments diagonaux nuls justifie le choix de faire la décomposition d'un descripteur où les éléments diagonaux n'ont pas encore été pré-calculés.

[17] La régulation des matrices locales peut éviter ce problème en enlevant la singularité des matrices.

[18] Deux types de décomposition LU sont possibles, la décomposition de Crout et la décomposition de Doolittle [100]. Très similaires, la différence entre ces deux options est l'obtention d'un matrice L ou une matrice U avec éléments diagonaux quelconques. La décomposition LU de Crout génère une matrice L avec éléments diagonaux quelconques et une matrice U avec éléments diagonaux égaux à 1. La décomposition LU de Doolittle, utilisée dans PEPS, génère une matrice U avec éléments diagonaux quelconques et une matrice L avec éléments diagonaux égaux à 1.

laquelle des solutions peut être obtenue et normalisée de façon à représenter un vecteur de probabilités. L'expérimentation pratique nous a montré que cette technique n'est pas utilisable pour le pré-conditionnement. Nous adoptons une heuristique [93] qui consiste à ne résoudre que le système triangulaire représenté par la matrice L (la matrice U n'est pas prise en compte). Ce choix drastique nous est imposé, car nous ne pouvons pas savoir laquelle des solutions du système représenté par la matrice U sera la plus intéressante pour le pré-conditionnement.

✍ **Soit**

$\mathfrak{Q}$ le descripteur obtenu avec l'élimination de tous les éléments fonctionnels du descripteur Q en les remplaçant par leur évaluation moyenne et avec l'application d'une des trois polices de régulation cités (*nor*, *shift* et *winglet*);

$\mathfrak{M}^{(i)}$ la matrice regroupant le i-ème terme de la somme du descripteur $\mathfrak{Q}$ ($\mathfrak{Q} \equiv \sum_{i=1}^{(N+2E)} \mathfrak{M}^{(i)}$);

$\mathfrak{L}^{(i)}$ la matrice triangulaire inférieure résultant de la décomposition LU de la matrice $\mathfrak{M}^{(i)}$;

$\mathfrak{U}^{(i)}$ la matrice triangulaire supérieure résultant de la décomposition LU de la matrice $\mathfrak{M}^{(i)}$ ($\mathfrak{M}^{(i)} \equiv \left[\mathfrak{L}^{(i)}\mathfrak{U}^{(i)}\right]$).

Puisque l'inverse d'une somme ne peut pas être exprimée par une combinaison des inverses des termes additionnés, nous ne pouvons pas utiliser directement $\mathfrak{Q}$ comme une approximation de Q. Par contre, l'inverse de chaque terme $\mathfrak{M}^{(i)}$ peut être exprimée aisément à l'aide d'une décomposition LU (équation 6.6).

L'idée de base du pré-conditionnement avec décomposition LU des termes tensoriels est l'utilisation de combinaisons des inverses des termes tensoriels. Trois possibilités peuvent être employées:

- utiliser la somme des inverses de tous les termes tensoriels $\mathfrak{M}^{(i)}$;
- utiliser le produit des inverses de tous les termes tensoriels $\mathfrak{M}^{(i)}$;
- utiliser l'inverse d'un seul terme tensoriel $\mathfrak{M}^{(i)}$.

Ces schémas, qui peuvent sembler naïfs, dérivent des méthodes de Schartz Additif et Multiplicatif [37, 74, 93]. Comme dans ces méthodes, le principe est d'utiliser l'inverse de chaque terme comme une composante du problème. De cette façon, l'utilisation des combinaisons de composantes permet un pré-conditionnement modulaire. Intuitivement, ce type de pré-conditionnement est intéressant lors qu'il s'agit d'un modèle avec peu d'interaction entre les composantes.

Dans l'option qui utilise la somme des inverses de chaque matrice $\mathfrak{M}^{(i)}$ comme pré-conditionneur, nous avons à multiplier le descripteur par:

$$\sum_{i=1}^{(N+2E)} \left[\mathfrak{M}^{(i)}\right]^{-1}$$

Prenons la méthode de la puissance comme exemple. Cette option équivaut à remplacer le contenu de la ligne 5 de l'algorithme 6.7 par l'opération de pré-conditionnement suivante:

$$x^{(k+1)} = y \sum_{i=1}^{2E+N} \left[\mathfrak{L}^{(i)}\mathfrak{U}^{(i)}\right]^{-1}$$

On va appeler ce type de pré-conditionnement de *pré-conditionnement additif*. Il faut remarquer que l'exécution de cette opération de pré-conditionnement demande un vecteur additionnel pour stocker les valeurs intermédiaires de la multiplication de y par chacun des $(N + 2E)$ termes.

Dans la deuxième option, nous utilisons comme matrice de pré-conditionnement le produit des inverses de chaque matrice $\mathfrak{M}^{(i)}$:

$$\prod_{i=1}^{(N+2E)} \left[\mathfrak{M}^{(i)}\right]^{-1}$$

Reprenons la méthode de la puissance comme exemple. Cette option équivaut à remplacer le contenu de la ligne 5 de l'algorithme 6.7 par l'opération de pré-conditionnement suivante:

$$x^{(k+1)} = y \prod_{i=1}^{(N+2E)} \left[\mathfrak{L}^{(i)}\mathfrak{U}^{(i)}\right]^{-1}$$

Nous allons appeler ce type de pré-conditionnement de *pré-conditionnement multiplicatif*. Cette option, au contraire du pré-conditionnement additif, ne nécessite pas de vecteur additionnel, puisque le résultat de chaque multiplication est toujours utilisé comme entrée de la prochaine multiplication.

La troisième et dernière de ces options de pré-conditionnement est la moins coûteuse, car un terme $\mathfrak{M}^{(i)}$ est choisi et son inverse utilisé comme matrice de pré-conditionnement:

$$\left[\mathfrak{M}^{(i)}\right]^{-1} \qquad \text{avec } i \in [1..(N+2E)]$$

De plus, il est possible d'utiliser de façon alternée chacun des termes $\mathfrak{M}^{(i)}$. Reprenons la méthode de la puissance, supposons que la valeur initiale de la variable i soit égale à 1. La ligne 5 de l'algorithme 6.7 doit être remplacée par:

$$x^{(k+1)} = y \left[\mathfrak{L}^{(i)}\mathfrak{U}^{(i)}\right]^{-1}$$
$$\textbf{next} \quad i \qquad \textit{(si } i < (N+2E)\textit{, } i = i+1\textit{, sinon } i = 1\textit{)}$$

Nous allons appeler ce type de pré-conditionnement de *pré-conditionnement alterné*.

Les pré-conditionnements avec décomposition LU des termes tensoriels n'ont pas abouti à des résultats encourageants. Bien que la convergence a pu être obtenue dans plusieurs cas, nous n'avons eu qu'une seule accélération par rapport aux résultats non pré-conditionnés (voir l'exemple 1.2.1 de l'annexe B). En effet, dans la méthode de la puissance presque aucune réduction a été possible. Il en reste que le parmi ces trois types de pré-conditionnements basés sur décomposition LU, le pré-conditionnement alterné est la meilleure des options. Son avantage par rapport aux deux autres types réside dans sa simplicité de traitement, car il n'augmente que très peu le coût en temps d'une itération. Cette conclusion nous suggère le prochain type de pré-conditionnement.

Pré-conditionnement avec la Diagonale

L'option la plus élémentaire pour une matrice de pré-conditionnement est d'utiliser la matrice diagonale contenant les éléments diagonaux de Q. Le système à résoudre devient:

$$xQD^{-1} = 0$$

où D est la matrice contenant seulement les éléments diagonaux du descripteur, *i.e.*:

$$D = \begin{pmatrix} q_{1,1} & & & \\ & q_{2,2} & & \\ & & \ddots & \\ & & & q_{n_Q,n_Q} \end{pmatrix}$$

Dans le cas d'une matrice stockée comme un descripteur (équation 3.2, page 63) ce type de pré-conditionnement est facile d'utilisation. En effet, la diagonale du descripteur est déjà calculée et même stockée dans un format vectoriel à part des éléments non-diagonaux stockés dans un format tensoriel (voir l'optimisation du pré-calcul de la diagonale présenté dans la section 5.3.1). De plus le calcul de l'inverse d'une matrice diagonale est élémentaire.

Ce type de pré-conditionnement est appelé *pré-conditionnement diagonal*. L'opération de pré-conditionnement (multiplication du vecteur $y = vQ$ par la matrice M^{-1}) est un produit scalaire du vecteur y par le vecteur contenant les inverses des éléments diagonaux du descripteur (D).

Bien que ce pré-conditionnement soit la plus naïve des approches, les meilleurs résultats ont été obtenus. Dans des nombreux exemples nous avons pu réduire le nombre d'itérations sans augmenter les temps d'exécution d'une itération. Le tableau 6.4 présente des extraits des résultats de l'annexe B. Dans ce tableau nous analysons les gains apportés par le pré-conditionnement diagonal dans les exemples suivants:

- modèles de partage de ressources (section 4.1) avec taux distincts par client:
  - exemple avec 12 clients et 4 ressources groupé en quatre automates (exemple 1.7.1 de l'annexe B);
  - exemple avec 12 clients et 4 ressources groupé en deux automates (exemple 1.8.1 de l'annexe B);
- modèles de réseau ouvert de files d'attente (section 4.3.2) groupés en deux automates et en deux situations distincts:
  - avec des taux de service amenant à une situation de distribution presque uniforme des clients par les files:
    $C_1 = 7$, $C_2 = 9$, $C_3 = 11$ (exemple 3.1.1 de l'annexe B);
    $C_1 = 8$, $C_2 = 8$, $C_3 = 12$ (exemple 3.2.1 de l'annexe B);
    $C_1 = 10$, $C_2 = 10$, $C_3 = 10$ (exemple 3.3.1 de l'annexe B);
  - avec des taux de service amenant à une situation de saturation de la première file:

$C_1 = 7$, $C_2 = 9$, $C_3 = 11$ (exemple 3.1.2 de l'annexe B);
$C_1 = 8$, $C_2 = 8$, $C_3 = 12$ (exemple 3.2.2 de l'annexe B);
$C_1 = 10$, $C_2 = 10$, $C_3 = 10$ (exemple 3.3.2 de l'annexe B);

Modèles de Partage de Ressources										
	Puissance		Arnoldi $m = 10$		Arnoldi $m = 20$		GMRES $m = 10$		GMRES $m = 20$	
Modèle	it.	sec.	it.	sec.	it.	sec.	it.	sec.	it.	sec.
1.7.1•	978	688.4	142	96.5	93	65.7	130	92.8	90	61.2
1.7.1⋆	–	–	75	54.1	67	46.3	80	58.0	**63**	**43.5**
1.8.1•	978	513.8	142	76.9	93	53.0	130	81.1	90	49.1
1.8.1⋆	–	–	75	46.7	67	38.6	80	50.2	**63**	**36.2**
Modèles de Réseau Ouvert avec Distribution Presque Uniforme des Clients										
	Puissance		Arnoldi $m = 10$		Arnoldi $m = 20$		GMRES $m = 10$		GMRES $m = 20$	
Modèle	it.	sec.	it.	sec.	it.	sec.	it.	sec.	it.	sec.
3.1.1•	1258	27.8	283	7.6	197	7.0	280	8.1	220	7.8
3.1.1⋆	–	–	216	**6.4**	**192**	7.1	239	7.4	215	7.9
3.2.1•	1459	36.2	349	11.9	234	9.6	330	11.3	273	11.3
3.2.1⋆	–	–	287	10.2	**205**	**8.8**	300	10.7	228	9.3
3.3.1•	1333	37.0	326	11.6	231	10.6	310	11.7	240	11.0
3.3.1⋆	–	–	245	9.7	**192**	**9.0**	280	11.1	220	10.3
Modèles de Réseau Ouvert avec Saturation de la Première File										
	Puissance		Arnoldi $m = 10$		Arnoldi $m = 20$		GMRES $m = 10$		GMRES $m = 20$	
Modèle	it.	sec.	it.	sec.	it.	sec.	it.	sec.	it.	sec.
3.1.2•	476	10.5	182	5.2	218	7.7	–	–	–	–
3.1.2⋆	–	–	**164**	**5.0**	179	6.0	–	–	–	–
3.2.2•	496	12.6	218	7.5	222	9.2	–	–	–	–
3.2.2⋆	–	–	**195**	**7.0**	220	9.5	–	–	–	–
3.3.2•	525	13.9	**172**	**6.5**	239	11.0	–	–	–	–
3.3.2⋆	–	–	320	12.7	217	10.1	–	–	–	–

• sans pré-conditionnement ⋆ avec pré-conditionnement diagonal

TAB. 6.4: Applications des Méthodes avec Pré-conditionnement Diagonal

Notons que le pré-conditionnement diagonal n'a pas accéléré la convergence de la méthode de la puissance. Par contre la méthode d'Arnoldi a pu être accélérée, notamment dans les cas où la convergence est plus lente (modèles avec saturation de la première file). Les réductions en nombre d'itérations sont plus intéressantes pour ce type de pré-conditionnement, car l'opération ajoutée (multiplication par l'inverse de la diagonale) est peu coûteuse.

6.3 Remarques Générales

À la fin de ce chapitre un certain nombre de conclusions peuvent être tirées. Tout d'abord, il est à remarquer que le format tensoriel employé par les RAS n'empêche pas l'application des méthodes classiques d'accélération du type Arnoldi et GMRES. De plus, ces méthodes présentent des gains importants pour tous les exemples expérimentés.

Méthodes de Base (non pré-conditionnées)

Le gain le plus significatif parmi les méthodes non pré-conditionnées a été trouvé dans le modèle de réseau fermé avec routage équilibré (exemple 2.1 dans l'annexe B). Dans ce cas la méthode de la puissance a pris 1010 itérations en 6.6 seconds, tandis que la méthode d'Arnoldi avec un espace de Krylov de taille 30 a pris seulement 76 itérations en 0.7 seconds. Ceci correspond à une réduction de presque un dixième du temps de la méthode itérative.

D'autres très bons résultats ont été atteint dans les modèles de partage de 4 ressources avec taux distincts pour chacun des 12 clients (exemples 1.7.1 et 1.8.1 dans l'annexe B), où nous avons réussi à obtenir des réductions de temps jusqu'à convergence du même ordre.

Pour une utilisation plus modeste de la mémoire (espace de Krylov de taille 10), nous avons trouvé de bonnes accélérations comme dans le cas des modèles de réseau ouvert avec distribution presque uniforme de clients. Notamment l'exemple 3.1.1 de l'annexe B montre une accélération de 1258 itérations en 27.8 seconds de la méthode de la puissance et 283 itérations en 7.6 seconds dans la méthode d'Arnoldi ($m = 10$). Ceci correspond à une réduction de presque un quart du temps jusqu'à la convergence.

Il faut rappeler que le choix de la taille de l'espace de Krylov reste une question ouverte (voir tableau 6.2, dans la page 143). Il est pourtant clair que pour des problèmes très grands on ne peut pas se permettre le stockage d'un nombre important de vecteurs. Cependant quelques accélérations remarquables ont pu être atteintes avec des nombres de vecteurs raisonnablement petits. Ceci est, par exemple, le cas du modèle 1.8.1 qui réduit le nombre d'itérations de 978 avec la méthode de la puissance à 284 itérations avec la méthode GMRES utilisant un espace de Krylov de taille 2.

Méthodes Pré-conditionnées

Bien que les techniques de pré-conditionnement n'ont pas toujours pu apporter des gains très significatifs dans les cas étudiés, quelques exemples ont eu de bons résultats. Parmi les plus grandes accélérations, nous avons l'application du pré-conditionnement diagonal aux modèles de partage de 4 ressources avec taux distincts pour chacun des 12 clients (exemples 1.7.1 et 1.8.1 dans l'annexe B). Dans ces cas, on a pu réduire le nombre d'itérations pour la méthode d'Arnoldi avec un espace de Krylov de taille 10 de 142 à 75 itérations, avec des gains respectifs de plus de 40% en temps jusqu'à la convergence. Il faut remarquer que ces gains sont a ajouter aux gains déjà apportés par l'utilisation de la méthode d'Arnoldi. Pour le cas spécifique de l'exemple 1.7.1, nous avons à comparer 978 itérations en 688.4 seconds avec la méthode de la puissance non pré-conditionnée avec 75 itérations en 54.1 seconds pour la méthode d'Arnoldi ($m = 10$) avec pré-conditionnement diagonal. Ceci représente un gain de l'ordre de 92% pour le temps jusqu'à la convergence.

Un deuxième succès relatif des techniques de pré-conditionnement est le cas des méthodes où le pré-conditionnement polynômial translaté a permis l'utilisation d'un espace de Krylov plus petit. Ceci est le cas des exemples des réseaux ouverts avec saturation de la première file (exemples 3.1.2, 3.2.2 et 3.3.2 dans l'annexe B). L'application de la méthode GMRES pour ces exemples avec le pré-conditionnement polynômial translaté

($k = 3$) a pu être faite avec un espace de Krylov de taille 10, tandis que le cas non pré-conditionné ne pouvait converger qu'avec un espace de Krylov de taille 30. Prenant exclusivement la comparaison entre la méthode de la puissance non pré-conditionnée et la méthode GMRES pour l'exemple 3.2.2, nous avons une réduction de 496 itérations en 12.6 seconds à 134 itérations en 11.5 seconds pour la méthode GMRES avec un espace de Krylov de taille 10 et pré-conditionnement polynômial translaté ($k = 3$).

Le principal échec des techniques de pré-conditionnement est le fait de n'avoir pas eu d'accélérations lors de l'application des pré-conditionnements à la méthode de la puissance. Ceci nous oblige à passer par une des méthodes de projection (Arnoldi et GMRES) pour avoir des accélérations. La seule exception est le cas du modèle de partage de 12 ressources avec 12 clients (exemple 1.2.1 dans l'annexe B), où la méthode de la puissance non pré-conditionnée a pris 662 itérations en 12 seconds et le pré-conditionnement alterné a pu réduire le nombre d'itérations à 258 et le temps d'exécution à 11.7 seconds.

Il n'en reste pas moins que les options les plus simples (et les moins coûteuses) de pré-conditionnement sont les options à envisager. Les pré-conditionneurs complexes impliquent une augmentation du coût des multiplications qui les rend inutiles. Ceci est le cas des exemples considérés à l'annexe B.

De ce fait nous pensons que parmi les types de pré-conditionnement présentés les seules options qu'il est intéressant d'essayer systématiquement sont:

– le pré-conditionnement diagonal;

– le pré-conditionnement polynômial translaté avec trois termes ($k = 3$)[19].

Nous pouvons encore ajouter à ces deux cas le pré-conditionnement alterné qui, même affichant un seule succès (exemple 1.2.1 cité), reste avec des temps assez proches des résultats non pré-conditionnés, notamment dans la méthode de la puissance.

Une dernière remarque à faire est que les temps obtenus ont été pris avec une implémentation séquentielle (logiciel PEPS chapitre 7). Nous pouvons imaginer des implémentations parallèles de la multiplication vecteur-descripteur qui puissent changer les rapports entre nombre d'itérations et temps d'exécution. Nous pouvons arriver à une situation où les pré-conditionnements qui réduisent le nombre d'itération soient plus intéressantes (même si le coût séquentiel d'une itération augmente beaucoup). Ceci peut réhabiliter les pré-conditionnements polynômiaux à 5 termes termes et surtout permettre l'utilisation du pré-conditionnement avec la méthode de la puissance.

[19]Nous rappelons que le l'utilisation d'un seul terme ($k = 1$) représente l'absence de pré-conditionnement, car le seule terme est une matrice identité. De plus les expériences avec deux termes ($k = 2$) n'ont pas eu de convergence. Donc, nous recommandons l'utilisation du pré-conditionnement polynômial translaté avec trois termes ($k = 3$) et non avec un nombre égal ou inférieur à trois termes.

Chapitre 7

PEPS 2.0

Dans ce chapitre est présentée la deuxième version du logiciel PEPS - *Performance Evaluation of Parallel Systems* [36]. PEPS est un outil informatique pour la définition et solution des modèles utilisant le formalisme RAS (chapitre 3) et sa première version fût proposée par Plateau, Fourneau et Lee [82]. En étant un outil académique, PEPS est construit de façon à donner la plus grande liberté à l'utilisateur. Notamment au niveau des solutions des modèles plusieurs options expérimentales sont disponibles.

Ce chapitre décrit dans la première section le format d'entrée des modèles RAS. La deuxième section décrit les opérations actuellement disponibles. La troisième section cite les fonctionnalités qui sont en développement pour la version 2.1 de PEPS et présente des données techniques sur le code du logiciel PEPS 2.0.

7.1 Description d'un RAS pour PEPS 2.0

PEPS reçoit la description d'un modèle RAS et calcule le vecteur de probabilités associé à la chaîne de Markov correspondant au modèle RAS décrit. Entre le format d'entrée du modèle RAS et la génération du vecteur de probabilités un certain nombre de structures de donnée intermédiaires peut être généré. Cette section décrit d'abord le format d'entrée d'un modèle RAS, ensuite les principales structures de donnée internes sont présentées.

Un modèle RAS est décrit par trois structures de donnée[1]:

- un tableau des fonctions contenant tous les éléments fonctionnels qui peuvent apparaître dans le modèle;
- un descripteur Markovien (équation 3.2, page 63), un ensemble de matrices contenant des éléments constants (nombres réels) et des éléments fonctionnels (identifiés par le tableau des fonctions);
- une fonction d'atteignabilité décrivant les états atteignables de l'espace produit du modèle.

[1]La prochaine version de PEPS à paraître dans le printemps 1998 (version 2.1) possédera des fonctionnalités permettant une description de plus haut niveau (voir section 7.3).

Chacune des structures est codée dans un fichier texte. Les extensions `.tft`, `.des`, `.fct` sont utilisés pour les fichiers décrivant respectivement les tableaux des fonctions, les descripteurs Markoviens et les fonctions d'atteignabilité. Ensuite, les informations contenues dans chacune des structures et le format du fichier correspondant est présenté.

7.1.1 Tableau des Fonctions

Les éléments fonctionnels ou fonctions d'un RAS sont des expressions arithmétiques où les variables sont l'état courant des automates. De cette façon, le tableau des fonctions est la structure qui définit les expressions arithmétiques correspondant à tous les éléments fonctionnels du modèle. Cette structure contient les information suivantes:

- le nombre d'automates et leur taille (nombre d'états locaux de chaque automate);
- le nombre d'éléments fonctionnels distincts utilisés dans les matrices du descripteur;
- le tableau avec la définition arithmétique de chacun des éléments fonctionnels distincts;
- la récompense associée à chaque état de chaque automate.

Le format générique du fichier (`.tft`) que décrit le tableau des fonctions est:

```
#domain      <le nombre d'automates> <le nombre d'états locaux de chaque automate>
#functions   <le nombre d'éléments fonctionnels distincts>

F0 :         <la définition arithmétique du premier élément fonctionnel distinct> ;
F1 :         <la définition arithmétique du deuxième élément fonctionnel distinct> ;
             ⋮
#rewards
0 0 :        < la récompense associée au premier état du premier automate>
0 1 :        < la récompense associée au deuxième état du premier automate>
             ⋮
1 0 :        < la récompense associée au premier état du deuxième automate>
             ⋮
```

Observations:

- les commentaires peuvent apparaître à tout moment limités par $\{\ldots\}$;
- les mots réservés `#domain` et `#functions`, ainsi que leurs suites sont obligatoires;
- le mot réservé `#rewards` et leur suite est optionnel, en l'absence de ces définitions la récompense du premier état est égal à 0, du deuxième égal à 1 et ainsi de suite;
- le nombre d'éléments fonctionnels distincts détermine le nombre total obligatoire de définitions arithmétiques;
- les identificateurs internes des définitions arithmétiques des éléments fonctionnels doivent apparaître successivement à partir de (`F0`);

– tous les symboles, identificateurs numériques, valeurs et mots réservés doivent être séparés par des blancs[2].

La syntaxe des définitions arithmétiques des éléments fonctionnels est faite en notation polonaise inversée avec les jetons de base suivants:

– les valeurs constantes:
  – valeurs numériques usuelles, par exemple:
    `1`, `2.5`, `.6`, `-.4`, `+3`, `3E+2`, `-5.768E-5`;
– les informations sur l'état des automates;
  – état courant d'un automate, par exemple: `@1` (état interne[3] de l'automate 1);
  – récompense de l'état courant d'un automate, par exemple: `%0` (récompense de l' état de l'automate 0);
  – somme des automates dans un état interne, par exemple: `$3` (le nombre total d'automates dans l'état interne 3);
– les opérateurs:
  – négation logique (unaire): `!`;
  – ou logique: `||`;
  – et logique: `&&`;
  – égalité: `==`;
  – inégalité: `!=`, `><` ou `<>`;
  – supérieur: `>`;
  – inférieur: `<`;
  – supérieur ou égal: `>=` ou `=>`;
  – inférieur ou égal: `<=` ou `=<`;
  – somme:`+`;
  – soustraction:`-`;
  – multiplication:`*`;
  – division:`/`;
  – maximum de deux valeurs: `max`;
  – minimum de deux valeurs: `min`;

À l'instar des langages non-typés, par exemple le langage C, le traitement des fonctions dans PEPS considère que toutes les opérations agissent sur des nombres réels. Même les valeurs entières (par exemple l'état interne d'un automate) sont traitées comme des nombres réels. Les opérateurs de comparaison génèrent le nombre réel 1.0 si la comparaison est vraie et 0.0 si la comparaison est fausse. De façon analogue, les opérateurs

[2]Les caractères de tabulation et de retour de chariot sont traités de la même façon que les blancs.

[3]La définition des états internes des automate est faite en attribuant 0 au premier état de l'automate, 1 au deuxième état de l'automate et ainsi de suite.

logiques considèrent 0.0 comme la valeur logique *faux* et tout nombre différent de 0.0 comme la valeur logique *vrai*.

La notation polonaise inversée a été choisie pour permettre la représentation d'une fonction par une liste de jetons qui peut être facilement interprétée par PEPS. Même sans l'utilisation de parenthèses, chaque liste de jetons est non ambiguë. Un format plus confortable pour l'utilisateur sera disponible avec la description de plus haut niveau prévue pour la prochaine version de PEPS (section 7.3). Le tableau 7.1 montre quelques fonctions et leur implémentation avec la syntaxe utilisée dans PEPS.

Fonction PEPS	Résultat	
`@2 0 == 2 *`	2	Si l'automate $\mathcal{A}^{(2)}$ est dans le premier état (état interne égal à zéro)
	0	Sinon
`@2 @3 + 5 < -3 +`	-2	Si la somme des états internes des automates $\mathcal{A}^{(2)}$ et $\mathcal{A}^{(3)}$ est inférieure à 5
	-3	Sinon
`%1 .5E2 < 3.2 * 2.7 max`	3.2	Si la récompense de l'état courant de l'automate $\mathcal{A}^{(1)}$ est inférieure à .005
	2.7	Sinon
`$0 8 >=`	1	Si le nombre des automates dans le premier état (état interne égal à zéro) est inférieure ou égal à 8
	0	Sinon
`%4 2 > @5 0 != &&`	1	la récompense de l'état courant de l'automate $\mathcal{A}^{(4)}$ est supérieure à 2 et l'automate $\mathcal{A}^{(5)}$ n'est pas dans son premier état
	0	Sinon
`-3.6 $9 /`		La division entre la valeur constante -3.6 est le nombre total d'automates dans l'état interne 9
`3.1415`		La valeur constante 3.1415

TAB. 7.1: Exemples de Fonctions dans PEPS

7.1.2 Descripteur Markovien

Le descripteur Markovien est construit à l'instar de son équation (introduite dans la page 63):

$$Q = \bigoplus_{i=1}^{N}{}_{g} Q_l^{(i)} + \sum_{e \in \varepsilon} \left(\bigotimes_{i=1}^{N}{}_{g} Q_{e^+}^{(i)} + \bigotimes_{i=1}^{N}{}_{g} Q_{e^-}^{(i)} \right)$$

La structure de donnée descripteur contient les informations suivantes:

- le nombre d'automates (N) et d'événements synchronisants ($cardinal(\varepsilon)$);
- le nombre d'états locaux de chaque automate (la taille des matrices correspondant à chaque automate);
- pour chaque automate $\mathcal{A}^{(i)}$:
  - une matrice contenant ses taux de transition locale $Q_l^{(i)}$;
- pour chaque événement synchronisant e:
  - une liste avec N matrices représentant l'occurrence de l'événement synchronisant e dans l'automate $\mathcal{A}^{(i)}$ ($Q_{e^+}^{(i)}$);
  - une liste avec N matrices représentant l'ajustement nécessaire à l'occurrence de l'événement synchronisant e dans l'automate $\mathcal{A}^{(i)}$ ($Q_{e^-}^{(i)}$);

Le format générique du fichier (`.des`) que décrit le descripteur Markovien est:

```
#type       <0 pour des modèles à temps discret ou 1 pour modèles à temps continu>
#automata   <le nombre d'automates>
#events     <le nombre d'événements synchronisants>
#sizes      <le nombre d'automates> <le nombre d'états locaux de chaque automate>

#locals
            <les matrices de transition locale>
#event 0
#positive
            <les matrices avec les taux d'occurrence du premier événement>
#negative
            <les matrices avec les taux d'ajustement diagonal du premier événement>
#event 1
            ⋮
```

Observations:

- PEPS 2.0 n'accepte que les modèles à échelle de temps continus;
- les commentaires peuvent apparaître à tout moment limités par $\{\ldots\}$;
- le nombre d'automates définit le nombre de matrices attendu à chaque bloc de matrices décrivant les automates;
- les blocs décrivant les matrices avec les taux des événement synchronisants doivent contenir même les matrices correspondant aux automates non concernés par l'événement (ces matrices sont toujours des identités);

– le nombre d'événements synchronisants détermine le nombre total obligatoire de définitions qui doivent apparaître successivement à partir de (`#event 0`);

– tous les symboles, identificateurs numériques, valeurs et mots réservés doivent être séparés par des blancs[4].

La définition de chacune des matrices est limitée par les symboles $[\ldots]$. Chaque ligne contient les éléments de la matrice séparés par des blancs et utilise le symbole `;` pour indiquer la fin de la ligne. Les lignes se succèdent jusqu'à la dernière ligne qui ne possède pas le symbole `;` à sa fin. Les éléments des matrices peuvent être des nombres réels constants ou bien des fonctions représentées par `F`i, i étant un indice interne de fonction déjà défini dans le tableau des fonctions. Le tableau 7.2 montre quelques exemples du codage des matrices dans PEPS.

Code PEPS	Matrice
`[ F0 2.4 F1 ; .5 -.7 2E-1 ; 0 6 -6 ]`	$\begin{pmatrix} F0 & 2.4 & F1 \\ 0.5 & -0.7 & 0.2 \\ 0 & 6 & -6 \end{pmatrix}$

`[ -4 1 3 ;`
`5 -10 5 ;`
`3 3 -6 ]`
	$\begin{pmatrix} -4 & 1 & 3 \\ 5 & -10 & 5 \\ 3 & 3 & -6 \end{pmatrix}$

`[ 1 0 ; 0 1 ]`	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

TAB. 7.2: Exemples de Matrices dans PEPS

7.1.3 Fonction d'Atteignabilité

La fonction d'atteignabilité est la structure la plus simple à définir, les informations qui la composent sont:

– le nombre d'automates et leur taille (nombre d'états locaux de chaque automate);

– la définition arithmétique d'une fonction qui retourne une valeur nulle pour les états globaux non atteignables.

Le format générique du fichier (`.fct`) que décrit la fonction d'atteignabilité est:

`#domain`	<le nombre d'automates> <le nombre d'états locaux de chaque automate>
`#function`	<la définition arithmétique de la fonction d'atteignabilité> ;

Observations:

– les commentaires peuvent apparaître à tout moment limités par $\{\ldots\}$;

– les mots réservés `#domain` et `#function` sont obligatoires;

– la définition arithmétique suit règles des éléments fonctionnels définies précédemment;

[4]Les caractères de tabulation et de retour de chariot sont traités de la même façon que les blancs.

– tous les symboles, identificateurs numériques, valeurs et mots réservés doivent être séparés par des blancs[5].

7.1.4 Exemple - Modèle de Partage de Ressources

Prenons un modèle de partage de ressources (voir section 4.1) avec 12 clients utilisant 8 ressources. Admettons que le taux de requête des ressources du i-ème client soit de $2i$ unités ($\lambda_i = 2i$), tandis que son taux de libération est de i unités ($\mu_i = i$).

La description de ce modèle pour PEPS a les fichiers suivants:

Descripteur Markovien - fichier `.des`

```
{ Model for ressource sharing N = 12 P = 8

  parameters:   lambda_i = 2 * i (i from 1 to 12)
                mu_i     =   i
}
#type 1
#automata 12
#events 0
#sizes 12       2 2 2 2 2 2 2 2 2 2 2 2
#locals
[  F0  F1 ;  1  -1 ]
[  F2  F3 ;  2  -2 ]
[  F4  F5 ;  3  -3 ]
[  F6  F7 ;  4  -4 ]
[  F8  F9 ;  5  -5 ]
[ F10 F11 ;  6  -6 ]
[ F12 F13 ;  7  -7 ]
[ F14 F15 ;  8  -8 ]
[ F16 F17 ;  9  -9 ]
[ F18 F19 ; 10 -10 ]
[ F20 F21 ; 11 -11 ]
[ F22 F23 ; 12 -12 ]
```

Tableau des Fonctions - fichier `.tft`

```
{ Model for ressource sharing N = 12 P = 8

  parameters:   lambda_i = 2 * i (i from 1 to 12)
                mu_i     =   i

  function f : $1 8 < ;
  ==>  the sum of automata using (state 1) is lesser than 8
}
#domain 12      2 2 2 2 2 2 2 2 2 2 2 2
```

[5]Les caractères de tabulation et de retour de chariot sont traités de la même façon que les blancs.

```
#functions 24
 F0  : $1 8 <  -2 * ;      { -lambda  1 f }
 F1  : $1 8 <   2 * ;      {  lambda  1 f }
 F2  : $1 8 <  -4 * ;      { -lambda  2 f }
 F3  : $1 8 <   4 * ;      {  lambda  2 f }
 F4  : $1 8 <  -6 * ;      { -lambda  3 f }
 F5  : $1 8 <   6 * ;      {  lambda  3 f }
 F6  : $1 8 <  -8 * ;      { -lambda  4 f }
 F7  : $1 8 <   8 * ;      {  lambda  4 f }
 F8  : $1 8 < -10 * ;      { -lambda  5 f }
 F9  : $1 8 <  10 * ;      {  lambda  5 f }
 F10 : $1 8 < -12 * ;      { -lambda  6 f }
 F11 : $1 8 <  12 * ;      {  lambda  6 f }
 F12 : $1 8 < -14 * ;      { -lambda  7 f }
 F13 : $1 8 <  14 * ;      {  lambda  7 f }
 F14 : $1 8 < -16 * ;      { -lambda  8 f }
 F15 : $1 8 <  16 * ;      {  lambda  8 f }
 F16 : $1 8 < -18 * ;      { -lambda  9 f }
 F17 : $1 8 <  18 * ;      {  lambda  9 f }
 F18 : $1 8 < -20 * ;      { -lambda 10 f }
 F19 : $1 8 <  20 * ;      {  lambda 10 f }
 F20 : $1 8 < -22 * ;      { -lambda 11 f }
 F21 : $1 8 <  22 * ;      {  lambda 11 f }
 F22 : $1 8 < -24 * ;      { -lambda 12 f }
 F23 : $1 8 <  24 * ;      {  lambda 12 f }
```

Fonction d'Atteignabilité -fichier `.fct`

```
{ Model for ressource sharing N = 12 P = 8

  parameters:   lambda_i = 2 * i (i from 1 to 12)
                mu_i     =   i
}
#domain 12      2 2 2 2 2 2 2 2 2 2 2 2
#function       $1 8 <= ;
{ the sum of automata using (state 1)
                  is lesser than or equal to 8 }
```

7.1.5 Exemple - Modèle de Réseaux de Files Fermé

Prenons un modèle de partage de ressources (voir section 4.2) avec un nombre de clients dans le réseau égal à 5. Admettons que les taux de service des files soient respectivement de 10, 5 et 4 unités ($\mu_1 = 10$, $\mu_2 = 5$ et $\mu_3 = 4$) et les probabilités de routage soient:

– 20% pour aller de la file s_1 à la file s_2 ($\pi_1 = 0.2$);

– 80% pour aller de la file s_1 à la file s_3 ($\pi_2 = 0.8$);
– 60% pour aller de la file s_2 à la file s_1 ($\pi_3 = 0.6$);
– 40% pour rester dans la file s_2 ($\pi_4 = 0.4$).

La description de ce modèle pour PEPS a les fichiers suivants:

Descripteur Markovien - fichier `.des`

```
{  Model for 3 closed queues example 5-5-5

   mu 1 = 10 units   pi 1 = 0.2
   mu 2 =  5 units   pi 2 = 0.8
   mu 3 =  4 units   pi 3 = 0.6
                     pi 4 = 0.4
   aut 0 => s1       ev 0 =>  s1 -> s2
   aut 1 => s2       ev 1 =>  s1 -> s3
   aut 2 => s3       ev 2 =>  s2 -> s1
                     ev 3 =>  s3 -> s1 }

#type 1      #automata 3     #events 4

#sizes 3     5 5 5       { C1 = 5, C2 = 5, C3 = 5 }
#locals
[ 0 0 0 0 0 ;    { aut 0 - local }
  0 0 0 0 0 ; 0 0 0 0 0 ; 0 0 0 0 0 ; 0 0 0 0 0 ]
[ 0 0 0 0 0 ;    { aut 1 - local }
  0 0 0 0 0 ; 0 0 0 0 0 ; 0 0 0 0 0 ; 0 0 0 0 0 ]
[ 0 0 0 0 0 ;    { aut 2 - local }
  0 0 0 0 0 ; 0 0 0 0 0 ; 0 0 0 0 0 ; 0 0 0 0 0 ]

#event 0         { ev 0 - customer from s1 to s2 }
#positive
[ 0 0 0 0 0 ;    { aut 0 - master }
  2 0 0 0 0 ;
  0 2 0 0 0 ;
  0 0 2 0 0 ;
  0 0 0 2 0 ]
[ 0 1 0 0 0 ;    { aut 1 - slave }
  0 0 1 0 0 ;
  0 0 0 1 0 ;
  0 0 0 0 1 ;
  0 0 0 0 0 ]
[ 1 0 0 0 0 ;    { aut 2 }
  0 1 0 0 0 ; 0 0 1 0 0 ; 0 0 0 1 0 ; 0 0 0 0 1 ]
#negative
[ 0  0 0 0 0 ;    { aut 0 - master }
```

```
  0 -2 0 0 0 ;
  0 0 -2 0 0 ;
  0 0 0 -2 0 ;
  0 0 0 0 -2 ]
[ 1 0 0 0 0 ;    { aut 1 - slave }
  0 1 0 0 0 ; 0 0 1 0 0 ; 0 0 0 1 0 ; 0 0 0 0 1 ]
[ 1 0 0 0 0 ;    { aut 2 }
  0 1 0 0 0 ; 0 0 1 0 0 ; 0 0 0 1 0 ; 0 0 0 0 1 ]

#event 1         { ev 1 - customer from s1 to s3 }
#positive
[ 0 0 0 0 0 ;    { aut 0 - master }
  8 0 0 0 0 ;
  0 8 0 0 0 ;
  0 0 8 0 0 ;
  0 0 0 8 0 ]
[ 1 0 0 0 0 ;    { aut 1 }
  0 1 0 0 0 ; 0 0 1 0 0 ; 0 0 0 1 0 ; 0 0 0 0 1 ]
[ 0 1 0 0 0 ;    { aut 2 - slave }
  0 0 1 0 0 ;
  0 0 0 1 0 ;
  0 0 0 0 1 ;
  0 0 0 0 0 ]
#negative
[ 0  0 0 0 0 ;    { aut 0 - master }
  0 -8 0 0 0 ;
  0 0 -8 0 0 ;
  0 0 0 -8 0 ;
  0 0 0 0 -8 ]
[ 1 0 0 0 0 ;     { aut 1 }
  0 1 0 0 0 ; 0 0 1 0 0 ; 0 0 0 1 0 ; 0 0 0 0 1 ]
[ 1 0 0 0 0 ;     { aut 2 - slave }
  0 1 0 0 0 ; 0 0 1 0 0 ; 0 0 0 1 0 ; 0 0 0 0 1 ]

#event 2          { ev 2 - customer from s2 to s1 }
#positive
[ 0 1 0 0 0 ;     { aut 0 - slave }
  0 0 1 0 0 ;
  0 0 0 1 0 ;
  0 0 0 0 1 ;
  0 0 0 0 0 ]
[ 0 0 0 0 0 ;     { aut 1 - master }
  3 0 0 0 0 ;
  0 3 0 0 0 ;
```

```
  0 0 3 0 0 ;
  0 0 0 3 0 ]
[ 1 0 0 0 0 ;     { aut 2 }
  0 1 0 0 0 ; 0 0 1 0 0 ; 0 0 0 1 0 ; 0 0 0 0 1 ]
#negative
[ 1 0 0 0 0 ;     { aut 0 - slave }
  0 1 0 0 0 ; 0 0 1 0 0 ; 0 0 0 1 0 ; 0 0 0 0 1 ]
[ 0  0 0 0 0 ;    { aut 1 - master }
  0 -3 0 0 0 ;
  0 0 -3 0 0 ;
  0 0 0 -3 0 ;
  0 0 0 0 -3 ]
[ 1 0 0 0 0 ;     { aut 2 }
  0 1 0 0 0 ; 0 0 1 0 0 ; 0 0 0 1 0 ; 0 0 0 0 1 ]

#event 3          { ev 3 - customer from s3 to s1 }
#positive
[ 0 1 0 0 0 ;     { aut 0 - slave }
  0 0 1 0 0 ;
  0 0 0 1 0 ;
  0 0 0 0 1 ;
  0 0 0 0 0 ]
[ 1 0 0 0 0 ;     { aut 1 }
  0 1 0 0 0 ; 0 0 1 0 0 ; 0 0 0 1 0 ; 0 0 0 0 1 ]
[ 0 0 0 0 0 ;     { aut 2 - master }
  4 0 0 0 0 ;
  0 4 0 0 0 ;
  0 0 4 0 0 ;
  0 0 0 4 0 ;
  0 0 0 0 4 ]
#negative
[ 1 0 0 0 0 ;     { aut 0 - slave }
  0 1 0 0 0 ; 0 0 1 0 0 ; 0 0 0 1 0 ; 0 0 0 0 1 ]
[ 1 0 0 0 0 ;     { aut 1 }
  0 1 0 0 0 ; 0 0 1 0 0 ; 0 0 0 1 0 ; 0 0 0 0 1 ]
[ 0  0 0 0 0 ;    { aut 2 - master }
  0 -4 0 0 0 ;
  0 0 -4 0 0 ;
  0 0 0 -4 0 ;
  0 0 0 0 -4 ]
```

Fonction d'Atteignabilité - fichier `.fct`

```
{ Model for 3 closed queues example 5-5-5 }
#domain 3     5 5 5
#function  { the sum of the states is equal to 5 }
```

```
  @0 @1 + @2 + 5 == ;
```

Tableau des Fonctions - fichier `.tft`

```
{  Model for 3 closed queues example 5-5-5 }
#domain 3       5 5 5
#functions 0    { no functions }
```

7.2 Interface Générale

Les structures de base utilisées pour saisir la définition d'un RAS dans PEPS sont traduites en des structures internes contenant essentiellement les mêmes informations.

- un fichier `.des` est traduit dans un fichier interne (extension `.dsc`) qui contiendra les matrices du descripteur;
- un fichier `.tft` est traduit dans un fichier interne (extension `.ftb`) qui contiendra le tableau avec la description des fonctions du modèle;
- un fichier `.fct` est utilisé dans la génération d'un fichier interne (extension `.rss`) qui contient le vecteur indiquant quels états globaux sont atteignables.

À ces trois structures internes, on ajoute quatre autres structures:

- fichier `.cnd`, le descripteur à échelle de temps continu normalisé, *i.e.*, prêt à l'emploi pour les méthodes de résolution;
- fichier `.lud`, le descripteur contenant la décomposition LU des matrices d'un modèle;
- fichier `.hbf`, une matrice en format creux *Hardwel-Boeing* [100] contenant l'évaluation complète d'un RAS;
- fichier `.vct`, un vecteur de probabilités (format de sortie d'une solution d'un RAS).

Deux autres types de fichiers sont aussi manipulés pour PEPS:

- fichier `.prf` contient les préférences de l'utilisateur lors de l'utilisation du logiciel;
- fichier `.tim` contient des rapports sur l'exécution des opérations sur PEPS, notamment l'application des méthodes de résolution (temps d'exécution, nombre d'itérations, etc).

Le tableau 7.3 montre tous les fichiers manipulés par PEPS.

7.2.1 Transformations de Descripteur

Les opérations de transformations des descripteurs font le passage d'un format interne à l'autre. Les quatre types de transformations disponibles sur PEPS 2.0 sont:

- Agrégation de Descripteur;
- Normalisation de Descripteur;
- Décomposition LU de Descripteur;
- Évaluation de Descripteur (génération d'un seule matrice creuse équivalente au descripteur).

Extension	Description
`.des`	Description textuelle des matrices du descripteur d'un modèle
`.tft`	Description textuelle des éléments fonctionnels utilisés dans le descripteur
`.fct`	Description textuelle de la fonction d'atteignabilité d'un modèle
`.dsc`	Structure interne de stockage des matrices du descripteur d'un modèle
`.ftb`	Structure interne de stockage du tableau de fonctions
`.rss`	Structure interne de stockage des états atteignables du modèle
`.cnd`	Structure interne de stockage d'un descripteur normalisé, *i.e.*, prêt à l'emploi pour les méthodes de solution
`.lud`	Structure interne de stockage des décompositions LU des matrices du descripteur d'un modèle
`.hbf`	Structure interne de stockage d'une matrice creuse
`.vct`	Structure interne de stockage d'un vecteur de probabilités
`.prf`	Préférences de l'utilisateur du logiciel
`.tim`	Rapport d'exécution des opérations du logiciel

TAB. 7.3: Fichiers Manipulés par PEPS 2.0

Agrégation de Descripteur

Cette opération demande:

– un descripteur non groupé (`.dsc`);

– un tableau de fonctions (`.ftb`);

– l'espace d'états atteignables du modèle non groupé (`.rss`).

Cette opération génère:

– un descripteur groupé (`.dsc`);

– un nouveau tableau de fonctions (`.ftb`);

– l'espace d'états atteignables du modèle groupé (`.rss`).

L'agrégation des descripteurs est l'opération qui permet de réduire le nombre d'automates du modèle par le biais de transformations purement algébriques (voir section 5.3.3). L'utilisateur doit aussi informer quels automates doivent être groupés. Ceci est fait à travers des questions qui indiquent dans quel groupe chacun des automates doit être agrégé.

Un nouveau tableau de fonctions est généré lors des l'évaluation des opérations tensorielles entre matrices locales d'automates d'un même groupe. L'espace d'états atteignables du modèle groupé est généré avec l'éventuelle réduction dans l'espace produit du modèle originel. À la fin du processus, chacun des groupes devient un automate du nouveau descripteur (descripteur groupé).

Notons que l'agrégation des descripteurs ne peut être appliquée qu'une fois, car un descripteur déjà groupé ne peut être groupé davantage. L'utilisateur peut, néanmoins, générer des descripteurs groupés avec différents niveaux d'agrégation à partir d'un même descripteur originel (non groupé).

Normalisation de Descripteur

Cette opération demande:

– un descripteur non normalisé (`.dsc`);

– un tableau de fonctions (`.ftb`).

Cette opération génère:

– un descripteur normalisé (`.cnd`);

– un nouveau tableau de fonctions (`.ftb`).

La normalisation d'un descripteur (`.dsc`), génère un descripteur prêt à l'emploi pour les méthodes de résolution. Descripteurs groupés ou non groupés sont acceptés comme entrée.

Plusieurs traitements doivent être fait avant l'application des méthodes itératives de solution. D'abord les éléments du descripteur doivent être normalisés avec leur plus grand élément. Ensuite, la recherche et l'élimination des dépendances fonctionnelles cycliques (voir section 5.2.2) doit être faite. Les optimisations de pré-calcul de la diagonale et le choix des ordres de traitement sont effectués. Le nouveau tableau de fonctions contient les nouvelles définitions des éléments fonctionnels, car le processus de normalisation peut impliquer la multiplication d'une fonction par le facteur de normalisation.

Décomposition LU de Descripteur

Cette opération demande:

– un descripteur non normalisé (`.dsc`);

– un tableau de fonctions (`.ftb`);

– l'espace d'états atteignables du modèle (`.rss`).

Cette opération génère:

– un descripteur décomposé (`.lud`).

Cette opération permet l'obtention d'une structure contenant la décomposition LU des matrices du descripteur (voir section 6.2.2). Descripteurs groupés ou non groupés sont acceptés comme entrée.

Lors de cette décomposition plusieurs options sont disponibles. Notamment, l'emploi de diverses politiques d'élimination des éléments fonctionnels et de régulation des matrices (voir section 6.2.2, page 154). L'élimination des éléments fonctionnels se fait selon leurs évaluations possibles dans l'espace d'états atteignables du modèle.

Comme nous avons déjà vu (section 6.2.2), cette opération doit s'appliquer sur un descripteur non normalisé (`.dsc`), car le pré-calcul de la diagonale empêche la décomposition LU des matrices locales. Pourtant, ce descripteur décomposé (`.lud`) doit représenter la même matrice que le descripteur normalisé (`.cnd`). Ceci oblige une normalisation avec le facteur utilisé pour générer le descripteur normalisé (`.cnd`). Ce facteur de normalisation peut être disponible sur un descripteur normalisé (`.cnd`) déjà évalué ou par la génération de la diagonale du descripteur non normalisé (`.dsc`). Pour optimiser ce processus, PEPS suggère toujours la génération d'un descripteur décomposé (`.lud`) lors de la normalisation d'un descripteur.

Évaluation de Descripteur

Cette opération demande:

– un descripteur non normalisé (`.dsc`);

– un tableau de fonctions (`.ftb`);

– l'espace d'états atteignables du modèle (`.rss`).

Cette opération génère:

– une matrice creuse au format *Hardwell-Boeing* (`.hbf`).

Cette opération permet l'obtention d'une seule structure contenant la matrice creuse équivalente à l'évaluation complète d'un descripteur. Descripteurs groupés ou non groupés sont acceptés comme entrée. À l'instar de l'opération d'agrégation de descripteur, cette opération fait l'agrégation totale de tous les automates du modèle en un seul automate. Cet automate est donc représenté par une seule matrice, car tous les changements d'états seront locaux à cet automate unique et seulement les états atteignables du modèle ont besoin d'être stockés. De la même façon, tous les éléments fonctionnels peuvent être évalués. Le résultat est donc, une matrice stockée en format creux où les éléments sont exclusivement constants (des nombres réels).

Ce format standard permet l'exportation des modèles développés en RAS à des logiciels de traitement de modèles Markoviens en format creux traditionnel [98]. Il faut remarquer que certains modèles de taille raisonnable en format descripteur peuvent demander énormément de place mémoire pour le calcul et pour le stockage du résultat.

7.2.2 Méthodes de Résolution

Les méthodes de solution impliquent en l'obtention d'un vecteur de probabilités que représente la solution stationnaire d'un modèle. Selon le format du modèle nous avons deux familles de méthodes de solution:

– les solutions des modèles RAS sur format de descripteur normalisé;

– les solutions des modèles RAS sur le format d'une seule matrice creuse sur format Hardwell-Boeing.

Solutions de Descripteurs

Cette opération demande:

– un descripteur normalisé (`.cnd`);

– un tableau de fonctions (`.ftb`);

– l'espace d'états atteignables du modèle (`.rss`);

– en option: un descripteur décomposé (`.lud`).

Cette opération génère:

– un vecteur de probabilités (`.vct`).

Cette opération applique l'une des trois méthodes de base (voir section 6.1) du logiciel PEPS, à savoir:

- méthode de la puissance;
- méthode d'Arnoldi;
- méthode GMRES.

Ces trois méthodes peuvent être appliqués en sept versions distincts selon les options de pré-conditionnement (voir section 6.2) suivantes:

- pas de pré-conditionnement;
- pré-conditionnement additif;
- pré-conditionnement multiplicatif;
- pré-conditionnement alterné;
- pré-conditionnement polynômial par inverse approché;
- pré-conditionnement polynômial translaté;
- pré-conditionnement diagonal;

Si une méthode est utilisée avec un pré-conditionnement additif, multiplicatif ou alterné, un descripteur décomposé doit être fourni. Les autres options nécessitent exclusivement un descripteur normalisé, le tableau de fonctions et l'espace d'états atteignables du modèle.

L'utilisateur peut préciser un grand nombre d'options pour l'obtention de la solution, notamment:

- le nombre maximal d'itérations (un nombre entier positif);
- le type de test entre les itérations:
  - aucun test (exécution du nombre maximal d'itérations);
  - un test de convergence avec les paramètres suivants:
    - la tolérance acceptée pour la convergence (un nombre réel positif plus petit que 1.0);
    - le type de vérification de la tolérance:

      la plus grand valeur absolue des résidus (précision individuelle absolue);

      la somme des valeurs absolues des résidus (précision accumulée absolue);

      la plus grand valeur absolue des résidus divisée par le plus grand élément du vecteur de solution (précision individuelle relative);
- le vecteur initial:
  - vecteur équiprobable;
  - vecteur calculé d'après la diagonale du descripteur;
  - vecteur quelconque fourni par l'utilisateur;

– le type d'optimisation de multiplication vecteur-descripteur (voir section 5.3.2):
  aucun re-ordonnancement des automates;
  re-ordonnancement des automates à chaque terme produit tensoriel;
  re-ordonnancement des automates à chaque facteur normal;
  re-ordonnancements faits automatiquement selon l'heuristique préconisée;

Solutions de Matrice Creuse

Cette opération demande:

– une matrice creuse format Hardwell-Boeing (`.hbf`).

Cette opération génère:

– un vecteur de probabilités (`.vct`).

Cette solution permet l'obtention du vecteur de probabilités de la solution stationnaire d'une matrice en format creux. La version actuelle de PEPS (version 2.0) permet seulement la résolution utilisant la méthode de la puissance. Cette option n'a pour objectif que de comparer, quand la taille du modèle en format matrice creuse le permet, les résultats obtenus en utilisant les méthodes de résolution des modèles en format descripteur.

7.2.3 Autres Opérations

Aux opérations de base du logiciel PEPS, les opérations génériques sont disponibles:

– Gestionnaire des Structures en Mémoire;

– Observation des Vecteurs de Probabilités;

– Préférences et Paramètres des Opérations;

Gestionnaire des Structures en Mémoire

Le gestionnaire des structures en mémoire permet de visualiser les structures de base (descripteurs, tableaux, vecteurs et matrices) dans la mémoire du logiciel. Des informations de base, ainsi que la place en kilo octets occupée par chacune des structures sont montrées à l'utilisateur qui peut enlever une ou plusieurs structures de la mémoire (libérer les structures alloués). Les structures affichées et leur informations associées sont:

– descripteur non normalisé (`.dsc`):
  nombre d'automates;
  nombre d'événements synchronisants;
  place occupée en kilo octets;

– tableau de fonctions (`.ftb`):
  nombre de fonctions;
  place occupée en kilo octets;

– vecteur d'états atteignables du modèle (`.rss`):
  nombre d'états de l'espace produit;
  place occupée en kilo octets;
– descripteur normalisé (`.cnd`):
  nombre d'automates;
  nombre d'événements synchronisants;
  place occupée en kilo octets;
– descripteur décomposé (`.lud`):
  nombre d'automates;
  nombre d'événements synchronisants;
  place occupée en kilo octets;
– matrice creuse (`.hbf`):
  ordre de la matrice;
  nombre d'éléments non nuls;
  place occupée en kilo octets;
– vecteur de probabilités (`.vct`):
  taille du vecteur;
  place occupée en kilo octets;

Observation des Vecteurs de Probabilités

Puisque le logiciel PEPS version 2.0, ne fait aucun traitement des vecteurs de probabilités calculés, un certain nombre d'opérations pour permettre la vérification des résultats est disponible. Ces opérations permettent:

– la visualisation des probabilités associés à un sous ensemble contigu d'états (visualisation des éléments du vecteur);
– la visualisation d'un aperçu d'un vecteur montrant:
  la somme des ses éléments;
  le plus grand élément (sa valeur et position);
  les quatre premiers et derniers éléments du vecteur;
– la comparaison entre deux vecteurs (montrant la plus grande différence absolu entre ses éléments).

Préférences et Paramètres des Opérations

La plus part des paramètres des opération exécutées par le logiciel PEPS sont définis dans l'option de préférences. Les paramètres définis sont:

– mode *verbose*:
  toutes les opérations exécutées génèrent des rapports (fichiers `.tim`) indiquant l'opération exécutée et le temps nécessaire;

les méthodes de résolution affichent à l'écran l'état actuel de la convergence

chaque fichier sauvé par le logiciel vérifie l'existence d'autres fichiers avec le même non et si tel est le cas demande à l'utilisateur si le nouveau fichier doit écraser le fichier déjà existant;

- nombre maximal d'itérations pour les méthodes de résolution;
- le type de test entre itérations pour les méthodes de solution:

  aucun test (exécution du nombre maximal d'itérations); ou

  test de convergence;
- la tolérance acceptée pour la convergence;
- le type de vérification de la tolérance:

  la plus grand valeur absolue des résidus (précision individuelle absolue);

  la somme des valeurs absolues des résidus (précision accumulée absolue);

  la plus grand valeur absolue des résidus divisée par le plus grand élément du vecteur de solution (précision individuelle relative);
- le vecteur initial:

  vecteur équiprobable;

  vecteur calculé d'après la diagonale du descripteur;

  vecteur quelconque fourni par l'utilisateur;
- le type de optimisation de multiplication à être exécutée (voir section 5.3.2):

  aucun re-ordonnancement des automates;

  re-ordonnancement des automates à chaque terme produit tensoriel;

  re-ordonnancement des automates à chaque facteur normal;

  re-ordonnancements faits automatiquement selon des heuristiques;
- le seuil (*threshold*) de la décomposition LU des matrices locales;
- la politique d'élimination de fonctions pour la décomposition LU des matrices locales:

  remplacer des éléments fonctionnels par zéro;

  remplacer les éléments fonctionnels par une de leurs évaluations;

  remplacer les éléments fonctionnels par leur plus grande évaluation;

  remplacer les éléments fonctionnels par leur plus petite évaluation;

  remplacer les éléments fonctionnels par leur évaluation moyenne (moyenne algébrique de toutes évaluations);
- la politique de régulation des matrices locales pour leur décomposition LU:

  ne faire aucune régulation;

  régulation avec translation (*shift*);

  régulation avec la transformation de Winglet;
- le nombre de vecteurs pour l'espace de Krylov des méthodes d'Arnoldi et GMRES;
- le nombre de termes pour le pré-conditionnement polynômial;

7.3 Fonctionnalités à Venir et Caractéristiques Techniques

La version de PEPS 2.0 est une version expérimentale qui contient toutes les fonctionnalités fondamentales pour le traitement numérique des modèles RAS à échelle de temps continu. Cependant une connaissance détaillée des RAS est nécessaire à l'utilisateur dans la version actuelle. Pour une utilisation plus large du logiciel, *i.e.*, sans demander à l'utilisateur des connaissances plus pointues des RAS un certain nombre de fonctionnalités doivent être incorporées. La version 2.1 de PEPS actuellement en développement a pour but d'inclure ces fonctionnalités. Notamment, la définition des RAS deviendra plus facile par le biais d'un format textuel pour permettre la définition du réseau d'automates (et non seulement du descripteur). De cette façon, le format descripteur, actuellement utilisé pour définir un RAS, n'a pas besoin d'être connu par l'utilisateur. De plus, identificateurs textuels pourrons être employés pour référencer les automates, les états locaux et les éléments fonctionnels du modèle. Il faut remarquer que toutes ces extensions rendrons beaucoup plus facile la définition des modèles pour PEPS, sans pour autant ajouter de nouvelles techniques de traitement numérique des modèles.

Le deuxième ensemble de fonctionnalités devant être incorporé dans le logiciel concerne la sortie de résultats. Il est prévu la mise au point d'un module d'intégration des résultats permettant la manipulation algébrique des vecteurs de probabilités. Ce module permet à l'utilisateur d'obtenir des mesures fines du modèle. Par exemple pour des modèles du type file d'attente (voir sections 4.2 et 4.3.2), il est possible d'établir le nombre moyen de clients dans une file, les probabilités de dépassement d'un seuil de clients dans une file, etc. En effet, ce module d'intégration de résultats correspond à l'évaluation de fonctions où les variables sont les éléments du vecteur de probabilités. Le module d'intégration de résultats permettra au logiciel PEPS d'assurer sa vocation d'outil d'évaluation quantitative de systèmes modélisés par les RAS.

Caractéristiques Techniques

Dans sa version actuelle, PEPS a été implémenté en langage C++ standard [102] selon les règles usuelles du langage. Des versions de PEPS ont été compilées et testés dans trois plateformes distincts:

- IBM RS6000, système AIX 4.2, compilateur `xlC` version 3.1.4 [115];
- SUN, système Solaris 2.5 (SunOS 5.5), compilateur `CC` version 4.2 [116];
- IBP PC, système Linux 2.0, compilateur `gcc` version 2.7.2.1 [117];

Le code de PEPS 2.0 et sa documentation [36] est disponible sur simple demande à partir du printemps 1998 à l'adresse électronique *peps@imag.fr*.

Chapitre 8

Conclusion

L'apport scientifique de cette thèse est double:

– l'efficacité de la multiplication vecteur-descripteur;

– l'efficacité des méthodes itératives et leur pre-conditionnement pour les RAS.

Dans cette conclusion, nous soulignons les bénéfices de chacun de ces axes. Notamment, nous rappelons que face à des modèles utilisant l'algèbre tensorielle généralisée, la complexité du produit par un vecteur reste du même ordre que celle des modèles utilisant l'algèbre tensorielle classique. De plus, nous montrons les réductions, parfois très significatives, pour le temps nécessaire à l'obtention de la solution stationnaire d'un modèle. Finalement, nous proposons des ouvertures théoriques et pratiques pour donner suite à cette thèse.

8.1 Multiplication Vecteur-Descripteur

Nouvelles Propriétés de l'Algèbre Tensorielle Généralisée

Les gains d'efficacité de la multiplication vecteur-descripteur présentés dans cette thèse sont basés sur la découverte de nouvelles propriétés de l'algèbre tensorielle généralisée. Avant la découverte de ces propriétés, le traitement des termes avec dépendances fonctionnelles devait se faire par la décomposition de chaque produit tensoriel généralisé en plusieurs produits tensoriels classiques [80]. Ces nouvelles propriétés (présentées au chapitre 2) ont permi le traitement efficace des termes sans cycle de dépendances fonctionnelles et a restreint la décomposition en produits tensoriels classiques aux cas avec cycles. De plus, cette décomposition peut se faire non plus sur la totalité des matrices fonctionnelles, mais seulement pour les matrices d'un transversal du circuit formé par les matrices faisant partie d'un cycle de dépendance fonctionnelle.

Les gains obtenus ainsi sont très importants pour tout RAS avec éléments fonctionnels. En reprenant des exemples présentés au chapitre 5 (voir page 119) nous vérifions des réductions de temps d'exécution énormes pour certains modèles. Seuls les cas très creux et avec très peu d'états atteignables sont mieux traités avec l'ancienne version des algorithmes. Le tableau 8.1 montre les temps d'une multiplication vecteur-descripteur dans

PEPS utilisant à la fois nos algorithmes (décrits au chapitre 5) et les anciens algorithmes de décomposition en produits tensoriels classiques [80].

Modèle	Notre Version	Ancienne Version	Réduction
$N = 16\ P = 16$	1.6 sec.	2 671.4 sec.	99.95%
$N = 16\ P = 15$	17.2 sec.	2 598.6 sec.	99.33%
$N = 16\ P = 8$	17.2 sec.	1 949.2 sec.	99.12%
$N = 16\ P = 4$	17.2 sec.	677.6 sec.	97.46%
$N = 16\ P = 2$	17.2 sec.	38.0 sec.	54.73%
$N = 16\ P = 1$	17.2 sec.	15.7 sec.	-9.55%

TAB. 8.1: Apport des Algorithmes basés sur les Nouvelles Propriétés de l'Algèbre Tensorielle Généralisée

Re-ordonnancement d'Automates

L'analyse des coûts de chaque partie de l'algorithme de base (algorithme 5.8 dans la page 106) nous a amené à deux choix de re-ordonnancement d'automates qui peuvent être employés pour optimiser davantage la multiplication. L'idée de base du re-ordonnancement d'automates est de réduire le nombre d'évaluations de fonctions. Cependant, le re-ordonnancement introduit des permutations de vecteurs qui représentent la multiplication des termes dans un autre ordre (voir section 5.3.2). De cette façon, nous avons trois possibilités de multiplication:

– méthode A: traite tous les termes du descripteur sans re-ordonnancement d'automates;

– méthode B: traite tous les termes du descripteur avec re-ordonnancement d'automates pour chaque terme produit tensoriel;

– méthode C: traite tous les termes du descripteur avec re-ordonnancement d'automates pour chaque facteur normal.

Une quatrième option naturelle est une méthode mixte qui consiste en appliquer à chaque terme du descripteur une des trois méthodes (A, B ou C). Pour chaque terme du descripteur une des méthodes sera la plus performante. Il s'agit d'établir un compromis entre la réduction du nombre d'évaluations de fonctions et la réduction du nombre de permutations.

Si les termes où la méthode A est plus performante sont facilement identifiables (les termes où aucune réduction du nombre d'évaluations de fonction est possible), les méthodes B et C vont représenter un choix délicat entre le coût des fonctions à évaluer par rapport au nombre de permutations nécessaires. En règle générale, les fonctions sont plus coûteuses que les permutations (par leur nombre et par leur complexité), donc le plus indiqué sera la méthode C, qui cherche à augmenter le nombre de permutations de vecteurs en réduisant le nombre d'évaluations de fonctions.

Pour certains modèles présentés le re-ordonnancement d'automates a pu réduire le temps d'exécution de plus de 90% (voir tableau 5.3 de la page 120). Ceci nous autorise à recommander fortement l'utilisation de ce type d'optimisation.

Groupement d'Automates

En comparant la multiplication d'un vecteur par le générateur d'un modèles RAS sous forme tensorielle (descripteur) avec la même opération faite avec une matrices creuse[1], nous pouvons remarquer une disavantage pour chacune des méthodes:

- la multiplication des matrices creuses demande beaucoup plus de place mémoire que les descripteurs (la place mémoire requise est proportionelle au nombre d'éléments non nuls de la matrice);
- la multiplication des descripteurs souvent ajoute beaucoup d'opérations (évaluation de fonctions, calcul de paramètres, permutation de vecteurs, etc) qui ne sont pas nécessaires à la multiplication des matrices creuses.

À partir de ces deux observations, nous sommes tentés de dire que l'utilisation des RAS est intéressante seulement si nous n'avons pas assez de place mémoire pour stocker (et multiplier par un vecteur) une matrice creuse.

L'optimisation basée sur le groupement d'automates représente des options intermédiaires entre l'approche RAS et l'approche matrice creuse. Pour les exemples présentés dans le chapitre 5 (page 124) nous avons pu constater que le temps de la multiplication de certains modèles a pu être réduit de plus de 95% avec à peine 5% d'augmentation de place mémoire pour le stockage du descripteur. Ce résultat a été obtenu avec le modèle de partage de ressources avec 20 clients et 19 ressources passant de 20 automates de taille 2 à 2 automate de taille 1024. La principale raison d'une réduction si importante est la petite taille des automates originaux (seulement 2 états). D'autre résultats intéressants dans ce même chapitre sont ceux des exemples de réseau ouvert de file d'attente (page 127). Pour ces cas, nous n'avons pas eu de réduction si importante, mais, à cause de l'élimination des états non atteignables, la place mémoire requise a diminué. Le cas avec des files de taille $C_1 = C_2 = 20$ et $C_3 = 50$ donne une réduction du temps de multiplication de 62% et une réduction de place mémoire de 46%.

Il n'en reste pas moins que l'opération de groupement, à l'instar de la génération d'une matrice creuse à partir d'un descripteur (voir l'opération d'aggrégation totale dans la description du logiciel PEPS) demande un certain temps et peut causer l'augmentation du nombre d'éléments non nuls. Si le temps de groupement a toujours été très raisonnable (ne dépassant jamais 30% du temps d'une multiplication), l'augmentation du nombre d'éléments non nuls et surtout l'augmentation du nombre des éléments fonctionnels peu

[1]Nous faisons une analyse sans rentrer dans des considérations à propos du coût de l'obtention d'un RAS par rapport au coût d'obtention d'une matrice creuse, car diverses techniques peuvent être employées pour passer d'un modèle à un générateur sous la forme d'une matrice creuse. Par exemple, nous pouvons utiliser des modèles en réseaux de Petri [3], en réseaux de files d'attente [18] où même des formalismes liés au chaînes de Markov [98]. Il est possible encore de décrire un RAS et évaluer toutes les opérations tensorielles (voir l'opération d'aggrégation totale dans la description du logiciel PEPS).

invalider complètement le groupement. Ceci a été le cas des modèles de réseau ouvert de file d'attente avec un mauvais groupement (voir page 127).

8.2 Méthodes de Solution

Bien que les méthodes de résolution implémentées dans PEPS 2.0 soient des méthodes itératives traditionnelles, les techniques de pré-conditionnement employées ont été développées en tenant compte du format tensoriel du descripteur.

L'implémentation de la méthode de la puissance ne fût qu'une application élémentaire de la multiplication vecteur-descripteur. L'implémentation des méthodes de projection (Arnoldi et GMRES) prouve l'intérêt que ces méthodes peuvent avoir dans le cadre des RAS. Le premier fait à remarquer est que les RAS, en réduisant les besoins de place mémoire pour stocker le générateur (par rapport au stockage en format creux), permettent l'utilisation d'un nombre plus important de vecteurs pour les méthodes de projection.

Le deuxième point important à remarquer est que des techniques de pré-conditionnement ont pu être trouvées pour certains exemples. Les résultats obtenus nous ont permis de mettre en avant quelques intuitions à propos des types de pré-conditionnement qui peuvent être envisagés pour des générateurs sur format tensoriel.

En règle générale, nous pouvons affirmer que l'opération de pré-conditionnement (multiplication par l'inverse approchée) ne doit pas être trop coûteuse. Ceci implique que les techniques de pré-conditionnement sont forcément très liées au format de stockage du générateur, *i.e.*, les techniques de pré-conditionnement habituellement appliquées aux générateurs en format creux ne sont pas *exportables* aux générateurs en format tensoriel.

En outre, certains exemples semblables du point de vue structurel du générateur (par exemple les variantes des réseaux ouverts à trois files) n'ont pas montré le même comportement face aux mêmes types de pré-conditionnement. Cette incertitude concernant la pertinence d'un pré-conditionnement nous amène à conclure que face à un modèle difficile (long à résoudre) les divers types pré-conditionneurs méritent d'être essayés. Pour les modèles moins difficiles, par contre, le pré-conditionnement n'est pas forcément justifié, car l'accélération de la convergence n'est pas assurée.

8.2.1 Comparaison avec d'autres Méthodes

Les résultats de l'action combinée des méthodes de projection et des techniques de pré-conditionnement peuvent être comparées à l'accélération de certains exemples avec une méthode semblable décrite par Uysal et Dayar [108]. Dans ces travaux, des implémentations traditionnelles et en bloc des méthodes de Gauss-Siedel et sur-relaxation successive sont développées pour des générateurs en format tensoriel. Prenons l'exemple du partage de ressources (section 4.1) avec N=12 et 16 clients, P=N-1 et N/2 ressources (modèles non-groupés) avec un taux de requête et de libération de ressources identiques pour tous les clients et égaux à $\lambda_i = 0.04$ et $\mu_i = 0.4$ respectivement.

Le tableau 8.2 montre les résultats obtenus pour la résolution de ce modèle avec les méthodes:

- Puissance;
- GS - Gauss-Siedel standard (extrait de [108]);
- SOR - sur-relaxation successive standard (extrait de [108]);
- GS B - Gauss-Siedel en bloc (extrait de [108]);
- SOR B - sur-relaxation successive en bloc (extrait de [108]);
- Arnoldi $m = 10$ - Arnoldi avec 10 vecteurs dans l'espace de Krylov et préconditionnement diagonale;
- GMRES $m = 10$ - GMRES avec 10 vecteurs dans l'espace de Krylov et préconditionnement diagonale;
- Arnoldi $m = 30$ - Arnoldi avec 30 vecteurs dans l'espace de Krylov et préconditionnement diagonale;
- GMRES $m = 30$ - GMRES avec 30 vecteurs dans l'espace de Krylov et préconditionnement diagonale;

Le nombre d'itérations (*it.*) indique le nombre de multiplications vecteur-descripteur effectuées jusqu'à la convergence avec tolérance de 10^{-10}. Le temps jusqu'à la convergence (*temps*) est indiqué en pourcentage du temps pris par la méthode de la puissance[2]. Par exemple, si l'indication du temps est de 3%, ceci indique que la méthode a mis seulement trois pour cent du temps utilisé par la méthode de la puissance de ce même exemple.

Méthodes proposées dans [108]										
Modèles		Puissance		GS		SOR		GS B		SOR B	
N	P	*it.*	*temps*	*it.*	*temps*	*it.*	*temps*	*it.*	*temps*	*it.*	*temps*
12	11	222	100%	28	20.3%	18	13.0%	26	17.9%	18	12.2%
12	6	222	100%	26	19.9%	18	12.2%	26	16.8%	18	11.5%
16	15	294	100%	34	18.2%	22	11.8%	32	16.5%	22	11.3%
16	8	294	100%	32	16.2%	22	11.1%	32	15.6%	22	10.6%
Méthodes proposées dans le chapitre 6											
Modèles		Puissance		Arnoldi $m = 10$		Arnoldi $m = 30$		GMRES $m = 10$		GMRES $m = 30$	
N	P	*it.*	*temps*	*it.*	*temps*	*it.*	*temps*	*it.*	*temps*	*it.*	*temps*
12	11	222	100%	30	13.1%	12	5.4%	32	14.3%	12	5.4%
12	6	222	100%	7	3.1%	7	3.2%	7	3.1%	7	3.2%
16	15	294	100%	52	17.4%	16	5.4%	89	29.5%	16	5.4%
16	8	294	100%	9	3.0%	9	3.0%	9	3.0%	9	3.0%

TAB. 8.2: Comparaison entre nos méthodes et les méthodes implémentés dans [108]

Nous pouvons remarquer que pour les exemples les moins creux ($P = N - 1$) les méthodes de sur-relaxation successive (SOR et SOR B) nécessitent légèrement moins de temps que nos méthodes de projection avec seulement 10 vecteurs. Pourtant, avec un plus grand nombre de vecteurs ($m = 30$) l'avantage de nos méthodes de projection sont indiscutables. Pour les exemples les plus creux ($P = N/2$), par contre, nos méthodes de projection sont nettement meilleures que les toutes les autres.

[2]Cette indication du temps est faite ainsi pour masquer les différences entre les machines utilisés pour nos expériences et celles prises dans [108]).

8.2.2 Résumé des Accélérations Obtenues

Avant cette thèse la résolution d'un modèle RAS devrait être faite selon les algorithmes présentés en [80]. Les accélérations qui ont été possibles dans la résolution des modèles RAS en utilisant les résultats de cette thèse sont:

- *df* traitement des dépendances fonctionnelles selon les nouvelles propriétés de l'algèbre tensorielle généralisée (section 5.2, page 98);
- *dd* pré-calcul de la diagonal du descripteur (section 5.3.1, page 110);
- *ra* re-ordonnancement d'automates (méthodes de multiplication distinctes - A, B et C) (section 5.3.2, page 112);
- *ga* groupement d'automates (section 5.3.3, page 121);
- *mp* méthodes de projection (Arnoldi et GMRES) (section 6.1, page 130);
- *pc* techniques de pré-conditionnement (section 6.2, page 143);

Pour illustrer les gains apportés avec chacun de ces concepts nous allons regarder l'évolution du temps jusqu'à la convergence pour trois modèles distincts[3]:

- *cas 1* modèle de partage de ressources (section 4.1), avec 12 clients, utilisant 4 ressources avec taux de requête et libération de ressources distincts pour chaque client (exemple 1.8.1 de l'annexe B);
- *cas 2* modèle de réseau fermé avec trois files (section 4.2), avec routage équilibré et 10 clients dans le réseaux (exemple 2.1 de l'annexe B);
- *cas 3* modèle de réseau ouvert avec trois files (section 4.3.2), avec capacités $C_1 = 7$, $C_2 = 9$ et $C_3 = 11$ avec distribution presque uniforme des clients (exemple 3.1.1 de l'annexe B).

Le tableau 8.3 montre l'évolution des temps jusqu'à la convergence, indiquant pour chaque accélération le nouveau temps nécessaire (*sec.*), les gains par rapport à l'accélération antérieure (*ant.*) et les gains des toutes les accélérations accumulées depuis le début (*acc.*). L'accélération entre deux valeurs est toujours indiquée par un facteur multiplicatif, par exemple si le temps est réduit à la moitié nous indiquons un facteur 2.00x. Si le facteur indiqué est égal à 1.00, aucune accélération a été obtenue. La première ligne du tableau indique le temps nécessaire jusqu'à la convergence avec les algorithmes présentés dans [80], *i.e.*, les algorithmes utilisés avant nos résultats.

Commentaires sur les résultats du *cas 1*

La première remarque est qu'aucun re-ordonnancement d'automates (*ra*) n'a été nécessaire, car la méthode A était déjà optimale. Le groupement des automates (*ga*) se fait en deux blocs, chacun représentant un sous-système avec 6 automates du modèle originel. La méthode de projection non pré-conditionnée (*mp*) qui a donné le meilleur résultat est la méthode GMRES avec un espace de Krylov de 30 vecteurs ($m = 30$). La méthode

[3]Ces modèles furent choisis parce qu'ils ont eu les plus grandes accélérations totales de chaque classe d'exemples.

	cas 1			*cas 2*			*cas 3*		
	sec.	*ant.*	*acc.*	*sec.*	*ant.*	*acc.*	*sec.*	*ant.*	*acc.*
[80]	27 919.1	–	–	8.1	–	–	15 347.6	–	–
df	954.3	29.26x	29.26x	8.1	–	–	548.7	27.97x	27.97x
dd	782.4	1.22x	35.68x	6.6	1.23x	1.23x	465.2	1.18x	32.99x
ra	–	–	–	–	–	–	151.1	3.08x	101.57x
ga	513.8	1.52x	54.34x	6.6	–	–	27.8	5.44x	552.07x
mp	45.7	11.24x	610.92x	0.7	9.43x	11.57x	7.0	3.97x	2 192.51x
pc	33.9	1.35x	823.57x	–	–	–	6.4	1.09x	2 398.06x

TAB. 8.3: Accélérations Obtenues

de projection pré-conditionnée (*pc*) qui a donné le meilleur résultat est la méthode d'Arnoldi avec un espace de Krylov de 30 vecteurs ($m = 30$) utilisant le pré-conditionnement diagonal.

Commentaires sur les résultats du *cas 2*

Ce modèle, n'ayant pas d'élément fonctionnel, ne permet pas d'obtenir des accélérations à cause du traitement des dépendances fonctionnelles (*df*). Pour la même raison, le re-ordonnancement d'automates (*ra*) est inutile. Nous avons choisi de ne pas faire de groupement (*ga*) pour ce modèle. La méthode de projection non pré-conditionnée (*mp*) qui a donné le meilleur résultat est la méthode d'Arnoldi avec un espace de Krylov de 30 vecteurs ($m = 30$). Aucun pré-conditionnement (*pc*) n'a pu être employé pour ce modèle.

Commentaires sur les résultats du *cas 3*

Pour ce modèle le re-ordonnancement des automates (*ra*) a été fait utilisant la méthode C de multiplication vecteur-descripteur. Le groupement d'automates (*ga*) a été fait de façon à éliminer les éléments fonctionnels, *i.e.,* en deux groupes, le premier avec les deux premiers automates ($\mathcal{A}^{(1)}$ et $\mathcal{A}^{(1)}$), le deuxième avec les deux derniers automates ($\mathcal{A}^{(3_1)}$ et $\mathcal{A}^{(3_2)}$). La méthode de projection non pré-conditionnée (*mp*) qui a donné le meilleur résultat est la méthode d'Arnoldi avec un espace de Krylov de 20 vecteurs ($m = 20$). La méthode de projection pré-conditionnée (*pc*) qui a donné le meilleur résultat est la méthode d'Arnoldi avec un espace de Krylov de 10 vecteurs ($m = 10$) utilisant le pré-conditionnement diagonal.

Commentaires Généraux

On peut facilement s'apercevoir que les modèles avec fonctions (*cas 1* et *3*) sont ceux qui présentent les plus grandes accélérations. Notons aussi que l'optimisation du pré-calcul de la diagonale (*dd*), même étant la plus modeste, est la seule optimisation qui donne des accélérations de façon sûre (ceci est vrai pour tous les modèles testés). Les méthodes de projection (*mp*) ont donné des accélérations dans presque tous les exemples testés (et dans la totalité des exemples montrés dans le tableau 8.3). Ces accélérations

sont souvent très grandes, car elles réduisent beaucoup le nombre d'itérations nécessaires à la convergence. Le pré-conditionnement (*pc*) n'a pas apporté de grandes accélérations, mais il faut noter que dans ces exemples les autres accélérations (notamment celle des méthodes de projection) ont déjà beaucoup accéléré la convergence. Comme remarque finale, il faut souligner que même pour des cas assez rapides à résoudre, comme le *cas 2*, une bonne accélération globale a pu être obtenue.

8.3 Perspectives et Travaux Futurs

La suite de cette thèse peut être vue en trois volets: les perspectives à court, moyen et long terme. Les perspectives à court terme concernent les travaux d'amélioration du logiciel PEPS. Les perspectives à moyen terme concernent les travaux dans le domaine des méthodes numériques. Les perspectives à long terme concernent les travaux où des connaissances dans d'autres domaines sont requises.

8.3.1 Perspectives à Court Terme

Comme nous l'avons déjà mentionné dans le chapitre 7, la version 2.1 de PEPS doit être finie au printemps de 1998. Cette nouvelle version prévoit l'ajout de fonctionnalités suivantes:

- entrée des modèles avec une syntaxe de réseau d'automates (et non celui de descripteur Markovien);
- utilisation d'identificateurs externes (textuels) pour les automates, états locaux et fonctions;
- améliorations dans la syntaxe des fonctions;
- module d'intégration des résultats.

En plus de ces fonctionnalités, une extension importante qui ne pose pas de problème théorique puisque le travail a déjà été fait dans [5], est la modélisation et résolution des modèles à échelle de temps discret. Les propriétés de l'algèbre tensorielle généralisée sont suffisantes pour implémenter la multiplication des descripteurs en temps discret, les changements dans le code PEPS doivent se limiter à des adaptations de structure de donnés.

8.3.2 Perspectives à Moyen Terme

Traitement de Modèles à Très Grand Espace d'États

Nous pouvons observer que de meilleurs résultats au niveau vitesse de convergence (plus petit temps d'exécution) sont obtenus avec des modèles résolus avec des méthodes de projection sur un espace de Krylov. Les modèles avec un grand espace d'états peuvent limiter le nombre de vecteurs de l'espace de Krylov et, de ce fait, empêcher l'utilisation des méthodes de projection à leur pleinne efficacité.

La première solution qui semble être envisageable est l'utilisation de vecteurs stockés en format creux [54]. Étant donné que les vecteurs manipulés par les RAS peuvent avoir

des tailles plus grandes que le nombre d'états atteignables cette option peut s'avérer très intéressante. De plus, dans le logiciel PEPS il existe déjà une structure vectorielle (`.rss` - états atteignables du modèle) qui stocke les informations concernant les états originaux (avant groupement), actuels (après groupement) et atteignables du modèle. Pour les modèles décrits par réseaux de Petri stochastiques généralisés et superposés (*superposed GSPN* [30]) cette type d'analyse numérique a déjà été proposée [55] nous laissant croire qu'une technique semblable peut être employée pour les RAS.

Perfectionner le Pré-conditionnement des RAS

Les résultats obtenus avec le pré-conditionnement des RAS (section 6.2) n'ont pas rempli nos attentes. Nous rappelons que le meilleur résultat obtenu avec le pré-conditionnement a été une réduction du temps jusqu'à la convergence de 44% (voir exemple 1.7.1 de l'annexe B). De plus, le pré-conditionnement n'a donné des résultats significatifs que lors de son application aux méthodes de projection.

Les techniques de pré-conditionnement utilisées dans les générateurs en format creux, par contre, donnent des résultats meilleurs et ce type de pré-conditionnement est applicable même à la méthode de la puissance. Ceci suggère l'étude d'autres alternatives de pré-conditionnement et le perfectionnement des alternatives déjà proposées.

Un exemple de perfectionnement des techniques de pré-conditionnement est la recherche d'une meilleure décomposition LU des termes produit tensoriel (voir section 6.2.2, page 154). La décomposition LU des termes tensoriels est faite en éliminant les éléments fonctionnels. Ce choix se justifie pour éviter la création de nouvelles fonctions lors de la décomposition des matrices locales et parce que l'algorithme de résolution de matrices triangulaires en format produit tensoriel (voir l'algorithme 5.4 dans la section 5.1.2) n'accepte que les matrices constantes.

La modification de l'algorithme 5.4 pour traiter des matrices triangulaires avec éléments fonctionnels semble être facile. Le principal problème à résoudre est la décomposition LU des matrices contenant des éléments fonctionnels sans générer des fonctions trop complexes. Une fois résolu ce problème, l'obtention d'une meilleure approximation de l'inverse d'une matrice décrite par un produit tensoriel devient possible, ce qui généralement donne un meilleur pré-conditionnement.

8.3.3 Perspectives à Long Terme

Paralléliser les Algorithmes

Nous avons augmenté l'efficacité de la multiplication et des méthodes itératives de résolution des RAS. De plus, ces résultats peuvent s'appliquer à tout formalisme pouvant décrire le générateur dans un format tensoriel. En ce qui concerne les accélérations du temps de résolution, une suite naturelle de cette thèse est l'étude de la parallélisation des algorithmes de multiplication vecteur-descripteur.

En principe, le format tensoriel, notamment celui de la somme tensorielle, semble être facilement parallélisable, car chacun des termes de la somme tensorielle peut être traités indépendamment. De façon analogue, chacun des termes des événements synchronisants

peut, en principe, être traité en parallèle. Comme règle générale nous pouvons affirmer que chaque terme de la somme du descripteur (équation 3.2, page 63) peut être traité en parallèle. Cette extension s'ajoute aux travaux de parallélisation des produits tensoriels [105]. De plus, la parallélisation des algorithmes de multiplication vecteur-descripteur pourra s'intégrer à une résolution totalement parallèle des RAS, notamment avec l'utilisation d'implémentations parallèles des méthodes de projection [31, 78].

Analyse Qualitative des RAS

Les systèmes modélisés avec les RAS peuvent être complexe. Ceci justifie l'existence d'outils capables de proceder à des analyses qualitatives des modèles RAS. Plusieurs algorithmes traditionnels d'analyse de graphes [11] peuvent être employés pour obtenir des informations locales à chaque automate. Nous pensons qu'il faut aussi envisager des outils pour faciliter la compréhension des interactions complexes entre les automates (éléments fonctionnels et événements synchronisants).

De plus, l'étude de certaines propriétés qualitatives des RAS peut amener à des avantages au niveau quantitatif aussi, comme c'est le cas des travaux qui partent des analyses structurelles des modèles RAS pour chercher des solutions approchées [68].

Simulation des RAS

Finalement, nous citons la possibilité de simulation des RAS qui peut permettre la résolution des modèles très grands (de l'ordre de 100.000 automates voire plus). L'utilisation des techniques de simulation rapide de modèles Markoviens [114] permet la simulation de réseaux de très grandes taille en un temps qui ne croît que linéairement avec le nombre d'automates (donc logarithmiquement avec le nombre d'états du modèle). Nous rappelons que pour les méthodes numériques itératives, le temps d'une seule itération est, dans le meilleur des cas, proportionnel au nombre d'états du modèle.

Cette possibilité peut élargir le domaine d'application du formalisme des RAS à des modèles où l'explosion combinatoire de l'espace d'états est prohibitif pour les methodes numériques. Ceci est le sujet de travaux sur la simulation du trafic routier d'une ville [84] où des modèles réels de ville sont prévus[4].

[4]Selon [84] le modèle de la ville de Grenoble (400 mille habitants) doit contenir environ 50.000 files d'attentes. Une grande capitale peut atteindre 1.000.000 de files d'attente.

Bibliographie

[1] O.Abu-Amsha, J.M.Vincent. **An algorithm to bound functionals of Markov chains with large state space**. Grenoble: Rapport de Recherche du Projet MAI, no. 25, 1996. Anonymous ftp *ftp://ftp.imag.fr/pub/MAI/Rapports/R_MAI025.ps.gz*

[2] M.Ajmone-Marsan, G.Balbo, G.Conte. *A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems*. **ACM Transactions on Computer Systems**, vol. 2, no. 2, 1984, pp. 93-122.

[3] M.Ajmone-Marsan, G.Balbo, G.Conte, S.Donatelli, G.Franceschinis. **Modelling with generalized stochastic Petri nets**. Chichester: John Wiley & Sons, 1995.

[4] W.E.Arnoldi. *The principle of minimized iteration in the solution of the matrix eigenvalue problem*. **Quart. Applied Mathematics**, vol. 9, 1951, pp. 17-29.

[5] K.Atif. **Modélisation du parallélisme et de la synchronisation**. INPG, Grenoble, 1993. (thèse de doctorat)

[6] F.Baccelli, G.Cohen, G.Olsder, J.P.Quadrat. **Synchronization and linearity: an algebra for discrete event systems**. John Willey & Sons, 1992.

[7] F.Baccelli, S.Hasenfuss, V.Schmidt. **Expansions for Steady-State Characteristics in $(\max, +)$-Linear Systems**. INRIA, Rapport de Recherche no. 2785, 1996. Anonymous ftp *ftp://ftp.inria.fr/INRIA/publication/RR/RR-2785.ps.gz*

[8] R.Barrett, M.Berry, T.F.Chan, J.Demmel, J.Donato, J.Dongarra, V.Eijkhout, R.Pozo, C.Romine, and H.Van der Vorst. **Templates for the solution of linear systems: building blocks for iterative methods**. Publication Electronique dans *http://www.netlib.org/templates.*

[9] F.Baskett, K.M.Chandy, R.R.Muntz, F.G.Palacios. *Open, closed and mixed networks of queues with different classes of customers*. **Journal of the ACM**, vol. 22, no. 2, 1975, pp. 248-260.

[10] R.Bellman. **Introduction to matrix analysis**. New York: McGraw-Hill, 1960.

[11] C.Berge. **Graphes et Hypergraphes**. Paris: Dunod, 1970.

[12] G.Birkhoff, R.Varga, D.Young. *Alternating direction implicit methods*. In: **Advances in Computers**, pp. 189-273. Academic Press, New York, 1962.

[13] V.S.Borkar. *Control of Markov chains with long-run average cost criterion: the dynamic programming equations*. **SIAM Journal of Control Optimization**, vol. 27, 1989, pp. 642-657.

[14] F.Boujdaine, J.M.Fourneau, N.Mikou. **Product form solution for stochastic automata networks with synchronizations**. (à paraître)

[15] G.W.Brahms. **Réseaux de Petri: théorie et pratique**. vol 1 & 2. Paris: Gauthier-Villars, 1983.

[16] R.Bronson. **Calcul matriciel**. New York: McGraw-Hill, Série Schaun, 1994.

[17] S.Bruel, G.Balbo. **Algorithms for closed queueing networks**. Cambridge: The MIT Press, 1980.

[18] P.Buchholtz. *A class of hierarchical queueing networks and their analysis*. **Queueing Systems**, vol. 15, 1994, pp. 59-80.

[19] J.Buzen *Computational algorithms for closed queueing networks with exponential servers*. **Communications of ACM**, vol. 16, no. 9, 1973, p.527-531.

[20] Z.Q.Cao, K.H.Kim, F.W.Roush. **Incline algebra and applications**. Ellis Horwood, 1984.

[21] K.M.Chandy, C.H.Sauer. *Approximate methods for analysing queueing network models of computer systems*. **ACM Computing Surveys**, vol. 10, 1978, pp. 281-317.

[22] P.J.Courtois. **Decomposability: queueing and computer systems applications**. London: Academic Press, 1977.

[23] P.J.Courtois, P.Semal. *Bounds for positive eigenvectors of non-negative matrices and for their approximations by decomposition*. **Journal of the ACM**, vol. 31, no. 4, 1994, pp. 804-825.

[24] Y.Dallery. **Approximate analysis of general open queueing networks with restricted capacity**. Grenoble: Laboratoire d'Automatique de Grenoble, 1989. (Rapport de Recherche)

[25] Y.Dallery, Z.Liu, D.Towsley. *Equivalence, reversibility, symmetry and concavity properties in fork-join queuing networks with blocking*. **Journal of the ACM**, vol. 41, no. 5, 1994, pp. 903-942.

[26] M.Davio. *Kronecker products and shuffle algebra*. **IEEE Transactions on Computers**, vol. C-30, no. 2, 1981, pp. 116-125.

[27] M.Davio, J.P.Deschamps, A.Thayse. **Discrete and switching functions**. New York: McGraw-Hill, 1978.

[28] T.Dayar, O.I.Pentakalos, A.B.Stephens. **Analytical modeling of robotic tape libraries using stochastic automata**. Technical Repport TR-97-198, CESDIS, NASA/GSFC, 1997.

[29] P.J.Denning, J.Buzen. *The operational analysis of queueing network models*. **ACM Computing Surveys**, vol. 10, 1978, pp. 235-280.

[30] S.Donnatelli. *Superposed stochastic automata: a class of stochastic Petri nets with parallel solution and distributed state space*. North-Holland: **Performance Evaluation**, vol. 18, 1993, pp. 21-36.

[31] J.Erhel. A parallel GMRES version for general sparse matrices. Electronic Transactions on Numerical Analysis, 3 pp. 160-176, 1995.

[32] P.Fernandes, B.Plateau. *Stochastic automata networks SAN: modelling and evaluation*. **Proceedings of the second process algebras and performance modelling workshop**, Ed. U.Herzog & M.Rettelbach. Universität Erlangen, 21-22, July 1994, pp. 127-136.

[33] P.Fernandes, B.Plateau, W.J.Stewart. **Efficient descriptor-vector multiplication in stochastic automata networks**. INRIA, Rapport de Recherche no. 2935, 1996. Anonymous ftp *ftp://ftp.inria.fr/INRIA/Publication/RR/RR-2935.ps.gz*

[34] P.Fernandes, B.Plateau, W.J.Stewart. *Numerical issues for stochastic automata networks*. **Proceedings of the fourth process algebras and performance modelling workshop**, Ed. M.Ribaudo. Torino, 4-5, July 1996. pp. 215-234.

[35] P.Fernandes, B.Plateau, W.J.Stewart. *Optimizing tensor product computations in stochastic automata networks*. **RAIRO**. (à paraître)

[36] P.Fernandes, B.Plateau. **PEPS 2.0: Using and Understanding**. Rapport INRIA. (à paraître - printemps 98)

[37] P.Fernandes, B.Plateau, Y. Saad, W.J.Stewart. **Efficient preconditioned iterative methods for stochastic automata networks**. Rapport INRIA. (à paraître)

[38] G.Florin, S.Natkin. *Les réseaux de Petri stochastiques*. **Tecniques et Sciences Informatiques**, vol 4, no. 1, 1985, pp. 143-160.

[39] J.M.Fourneau, F.Quessette. *Graphs and stochastic automata networks*. In: **Computations with Markov chains**. Ed.: W.J.Stewart. Boston: Kluwer Academic Publishers, 1995.

[40] S. Gaubert. *Performance evaluation of (max,+) automata*. **IEEE Transactions on on Automatic Control**, vol. 40, no. 12, 1995, pp.

[41] E.Gelenbe. *Product form queueing networks with negative and positive customer*. **Journal of Applied Probability**, vol. 28, 1991. pp. 656-663.

[42] W.Gordon, G.Newell. *Closed queueing systems with exponential servers*. **Operations Research**, vol. 15, 1967. pp. 254-265.

[43] G.Golub, C.Van Loan. **Matrix computations**. John Hopkins University Press, 1996.

[44] N.Götz, H.Hermanns, U.Herzog, V.Mertsiotakis, M.Rettelbach. *Constructive specification techniques - integrating functional performance and dependability aspects*. In: **Quantitative Methods in Parallel Systems**. Springer, 1995.

[45] S.Haddad. **Une catégorie régulière de réseaux de Petri de haut niveau: définition, propriétés et réduction**. Université Paris VI, 1987. (thèse de doctorat)

[46] S.Haddad. *A reduction theory for coloured nets*. In: **High-Level Petri Nets: Theory and Application**. Berlin: Springer-Verlag, 1991.

[47] J.Hillston. **A composition approach to performance modelling**. University of Edinburgh, 1994. (Phd. thesis)

[48] D.P.Heyman, M.J.Sobel. **Stochastic models in operations research**, (vol. 1) New York: McGraw-Hill, 1982.

[49] J.R.Jackson. *Networks of waiting lines*. **Operations Research**, vol. 5, 1957, pp. 518-521.

[50] J.R.Jackson. *Jobshop-like queueing systems*. **Management Science**, vol. 10, 1963, pp. 131-142.

[51] W.Jalby, B.Philippe. **Stability analysis and improvement of the block Gram-Schmidt algorithm**. INRIA, Rapport de Recherche no. 1162, 1990.

[52] K.Jensen. *Coloured Petri nets*. In: **High-Level Petri Nets: Theory and Application**. Berlin: Springer-Verlag, 1991.

[53] F.P.Kelly. **Reversibility and stochastic networks**. Chichester: John Wiley & Sons, 1979.

[54] P.Kemper. *Reachability analysis based on structures representations*. **Lecture Notes in Computer Science**, vol. 1091, 1996, 269-288.

[55] P.Kemper. *Numerical Analysys of superposed GSPNs*. **IEEE Transactions on computers**, vol. C-22, no. 9, 1996, 615-628.

[56] L.Kleinrock. **Queueing theory** (vol. 1 & 2). Chichester: John Wiley & Sons, 1975.

[57] H.Kobayashi. **Modeling and analysys**. Reading: Addison-Wesley, 1978.

[58] D.Krob. *The equality problem for rational series with multiplicities in the tropical semiring is undecidable*. **International Journal of Algebra and Computing**, vol. 3, 1993.

[59] P.Lascaux & R.Théodor. **Analyse numérique matricielle appliquée à l'art de l'ingénieur**. Paris: Masson, 1994. (vol. 1 & 2)

[60] E.Lazowska, J.Zahorjan, G.Graham, K.Sevic. **Quantitative system performance: computer system analysis using queueing networks models**. Englewood Cliffs: Prentice-Hall, 1984.

[61] R.J.Lechner. *Transformations among switching function canonical forms*. **IEEE Transactions on eletronic computers**, vol. EC-12, no. 2, 1963, pp. 129-130.

[62] S.Lipschultz. **Algèbre Linéaire**. New York: McGraw-Hill, Série Schaun, 1994.

[63] J.D.C.Little. *A proof of the queueing formula* $L = \lambda W$. **Operations Research**, vol. 9, 1961, pp. 383-387.

[64] S.Mahévas. **Modèles Markoviens de grande taille: calculs de bornes**. Université de Rennes I, 1997. (thèse de doctorat)

[65] D.Menascé, V.Almeida, L.Dowdy. **Capacity planning and performance modeling**. Englewood-Clifs: Prentice-Hall, 1994.

[66] C.D.Meyer. *Stochastic complementation, uncoupling Marlov chains and the theory of nearly reducible systems*. **SIAM Rewiew**, vol. 31, no. 2, 1989, pp. 240-272.

[67] R.Milner. **Communication and concurrency**. Englewood Cliffs: Prentice-Hall, 1985.

[68] N.L.Mokdad. **Méthodes et outils pour l'évaluation des performances des réseaux informatiques**. Université de Versailles-Saint-Quentin-en-Yvelines, 1997. (thèse de doctorat)

[69] P.Moller. *Notions de rang dans les dioides vectoriels*. In: **CNRS/CNET/INRIA Seminar: Algèbres Exotiques et Systèmes à Événements Discrets**, Issy-les-Moulineaux, France, 1987.

[70] P.Moreaux. **Struturation des chaînes de Markov des réseaux de Petri stochastiques - décomposition tensorielle et agrégation**. Université Paris Dauphine, 1996. (thèse de doctorat)

[71] M.F.Neuts. **Matrix geometric solutions in stochastic models, an algorithmic approach**. John Hopkins University Press, 1981.

[72] M.F.Neuts. **Stochastic geometric matrices of $G/M/1$ type and their applications**. Marcel Dekker inc., 1989.

[73] B.N.Parlet. *Canonical decomposition of Hessenberg matrices*. **Mathematical Computations**, vol. 21, 1967, pp. 223-227.

[74] D.Peaceman, H.Rachford. *The numerical solution of elliptic and parabolic differential equations*. **Journal of SIAM**, vol. 3, pp. 28-41, 1955.

[75] C.A.Petri. *General net theory*. In: **Proceedings of the joint IBM-University of Newcastle upon Tyne seminar**. University of Newcastle upon Tyne, 1977, pp.131-169.

[76] J.L.Peterson. **Petri net theory and the modelling of systems**. Englewood Cliffs: Prentice-Hall, 1981.

[77] B.Philippe, Y.Saad, W.J.Stewart. **Numerical methods in Markov chain modeling**. INRIA, Rapport de Recherche no. 1115, 1989.

[78] B.Philippe, B.Vital. *Parallel implementations for solving generalized eigenvalue problems with symmetric sparse matrices*. **Applied Numerical Mathematics**, vol. 12, 1993, pp. 391-402.

[79] B.Plateau. **De l'Evaluation du Parallélisme et de la Synchronisation**. Paris-Sud, Orsay, 1984. (thèse de doctorat)

[80] B.Plateau, K.Atif. *Stochastic automata networks for modelling parallel systems*. **IEEE transactions on software engineering**, vol. 17, no. 10, pp. 1093-1108, 1991.

[81] B.Plateau, J.M.Fourneau. *A methodology for solving markov models of parallel systems*. **Journal of parallel and distributed computing**, vol. 12, pp. 370-387, 1991.

[82] B.Plateau, J.M.Fourneau, K.H.Lee. *Peps: A package for solving complex Markov models of parallel systems*. In: R.Puigjaner, D.Potier, editors, **Modelling Techniques and Tools for Computer Performance Evaluation**, Spain, September, 1988.

[83] B.Plateau, W.J.Stewart. **Stochastic automata networks: product forms and iterative solutions**. INRIA, Rapport de Recherche no. 2939, 1996. Anonymous ftp *ftp://ftp.inria.fr/INRIA/Publication/RR/RR-2939.ps.gz*

[84] B.Plateau, B.Ycart. **Fast simulation kernel for urban trafic**. Rapport de Recherche MAI/IMAG, no. 41, 1997. Anonymous ftp *ftp://ftp.imag.fr/pub/MAI/Rapports/R_MAI041.ps.gz*

[85] W.H.Press, S.A.Teukolsky, W.T.Vetterling, B.P.Flannery. **Numerical recipes in C**. Cambridge: Cambridge University Press, 1992.

[86] F.Quessette. **De nouvelles méthodes de résolution pour l'analyse quantitative des systèmes paralleles et des protocoles**. Paris-Sud, Orsay, 1994. (thèse de doctorat)

[87] M.Rettelbach. **Stochastische prozeßalgebren mit zeitlosen aktivitäten und probabilistischen verzweigungen**. Universität Erlangen-Nürnberg, 1996. (dissertation)

[88] M.Reiser, S.S.Lavenberg. *Mean value analysis of closed multichain queueing networks*. **Journal of the ACM**, vol. 27, no. 2, 1980, pp. 313-322.

[89] W.Reisig. **Petri nets: an introduction**. Berlin: Springer-Verlag, 1985.

[90] G.Richter. *Clocks and their use for time modelling*. In: **Information Systems: Theoretical and Formal Aspects**. Amsterdam: North-Holland, 1985, pp.49-66.

[91] T.G.Robertazzi. **Computer networks and systems: queueing theory and performance evaluation**. New York: Springer-Verlag, 1990.

[92] Y.Saad, M.H. Schltz. *GMRES: a generalized minimal residual algorithm for solving nonsymetric linear systems*. **SIAM Journal of Scientific and Statistical Computing**, vol. 7, pp. 856-869, 1986.

[93] Y.Saad. **Iterative methods for sparse linear systems**. Boston: PWS Publishing Company, 1995.

[94] P.Schweitzer. *A survey of aggregation-disaggregation in large Markov chains* **Numerical Solutions of Markov Chains**. New York: Marcel Dekker Inc., 1991.

[95] H.A.Simon, A.Ando. *Aggregation of variables in dynamic systems*. **Econometrica**, vol. 29, 1961, pp.111-138.

[96] E.A.de Souza e Silva, R.R.Muntz. **Métodos Computacionais de soluçao de cadeias de Markov: aplicações a sistemas de computação e comunicação**. VIII Escola de Computação, Instituto de Informática, UFRGS, Porto Alegre, 1992.

[97] G.W.Stewart. **Introduction to matrix computations**. New York: Academic Press, 1973.

[98] W.J.Stewart. *MARCA: Markov chain analyser, a software package for Markov modelling*. **Numerical Solutions of Markov Chains**. New York: Marcel Dekker Inc., 1991.

[99] W.J.Stewart, K.Atif, B.Plateau. *The numerical solution of stochastic automata networks*. **European journal of operations research**, vol. 86, no. 3, pp. 503-525, 1995.

[100] W.J.Stewart. **Introduction to the numerical solution of Markov chains**. Princeton University Press, 1994.

[101] S.Stiham Jr., R.Weber. *A survey of Markov decision models for control of networks of queues*. **Queueing Systems**, vol. 13, 1993, pp. 291-314..

[102] B.Stroustrup. **The C++ programming language**. Addison-Wesley, Reading, Massachusetts, 1991.

[103] Y.Takahashi. **A lumping method for numerical calculation of stationary distributions of Markov chains**. Technical Report B-18, Department of Information Sciences, Tokyo Institute of Technology, 1975.

[104] M.Tazza. **Ein netztheoretisches modell zur quantitativen analyse von systemen (Q-modell)**. München: R. Oldenbourg Verlag, 1985. (Bericht, 149)

[105] A.Touzene. **Résolution des modèles markoviens sur machines à mémoires distribuees**. INPG, Grenoble, 1992. (thèse de doctorat)

[106] K.Trivedi. **Probability & statistics with reliability, queueing and computer science applications**. Englewwod Cliffs, New Jersey: Prentice-Hall, 1982.

[107] L.Truffet. *Near complete decomposability: bounding the error by a stochastic comparison method*. **Advanced Applied Probabilities**, vol. 29, 1997, pp. 803-855.

[108] E.Uysal, T.Dayar. *Iterative methods based on splittings for stochastic auatomata networks*. **European Journal of Operations Research**. (à paraître)

[109] R.S.Varga. **Matrix iterative analysys**. Englewwod Cliffs, New Jersey: Prentice-Hall, 1962.

[110] H.F.Walker. *Implementation of GMRES method using Householder transformations*. **SIAM Journal of Scientific Computing**, vol. 9, 1988, pp. 152-163.

[111] V.L.Wallace. **The solution of quasi birth and death processes arising from multiple access computer systems**. Technical Report, University of Michigan, 1969. (SEL technical report no. 35)

[112] J.Walrand. **An introduction to queueuing networks**. Englewwod Cliffs, New Jersey: Prentice-Hall, 1988.

[113] R.Weiss. *A theorectical overview of Krylov subspace methods*. In: W. Schönauer and R. Weiss, editors, Special issue on iterative methods for linear systems, pp. 33-56. **Applied Numerical Methods**, 1995.

[114] B.Ycart. **Simulation de modèles markoviens**. Grenoble: Université Joseph Fourier, 1997. (DESS d'Ingénierie Mathématique)

[115] IBM Corp. **C++ set for AIX/6000 - Version 2 Release 1**. North York, Ontario: IBM Canada Ltd. Laboratory, 1993.

[116] Sun Systems. **CC Compiler and the Workshop environment**. Sun Systems, 1996.

[117] GNU Documentation. **gcc compiler documentation** Format éléctronique seulement: `http://www.delorie.com/gnu/docs/`

Annexe A

Notation Employée

Dans cette annexe, toutes les notations employées dans cette thèse sont regroupées pour une consultation rapide.

✍ **Définitions d'Ensembles**

$\mathbb{N}$	l'ensemble de numéros naturels;
$\mathbb{R}$	l'ensemble de numéros réels;
$\mathbb{R}^*$	l'ensemble de numéros réels non nuls;
$\mathbb{R}^+$	l'ensemble de numéros réels positifs (zéro inclus);
$\mathbb{R}^-$	l'ensemble de numéros réels négatifs (zéro inclus);
$\mathbb{R}^{+*}$	l'ensemble de numéros réels strictement positifs (zéro exclu);
$\mathbb{R}^{-*}$	l'ensemble de numéros réels strictement négatifs (zéro exclu);
$[a..b]$	le sous-ensemble de $\mathbb{N}$ contenant tout les valeurs de a jusqu'à b (ces valeurs incluses);
$[a,b]$	le sous-ensemble de $\mathbb{R}$ contenant toutes les valeurs de a jusqu'à b (ces valeurs incluses);
$]a,b[$	le sous-ensemble de $\mathbb{R}$ contenant toutes les valeurs comprises entre a et b (ces valeurs exclues);
$]a,b]$	le sous-ensemble de $\mathbb{R}$ contenant toutes les valeurs de a jusqu'à b (a exclue, b incluse).

✍ **Permutations sur Ensembles d'Entiers**

σ	une permutation nommée σ sur l'intervalle $[1..N]$, un ordre d'indices entre 1 et N;
$\sigma(i)$	le rang de l'indice i dans l'ordre identifié par la permutation σ;
σ_k	l'indice placé au rang k de l'ordre identifié par la permutation σ (si $\sigma_k = i$, $\sigma(i) = k$);
$[i_1,\ldots,i_N]_\sigma$	une modification de l'ordre des n-uplets d'une coordonné selon une permutation σ, dans la pratique on change l'intervalle de variation de chacun des indices de façon à que le k-ème indice varie de $[1..n_k]$ vers $[1..n_{\sigma_k}]$.

A.1 Matrices

Les définitions matricielles ont été introduites dans le chapitre 2

✍ Matrices et Éléments

A	la matrice nommée A;
n_A	la dimension (nombre de lignes et de colonnes) de la matrice carrée A;
$a_{i,j}$	l'élément dans la ligne i et la colonne j de la matrice A;
$a_{[i,k],[j,l]}$	l'élément dans la ligne k du i-ème bloc horizontal et la colonne l du j-ème bloc vertical de la matrice A;
I_n	la matrice identité de dimension n;
$\delta_{i,j}$	l'élément dans la ligne i et la colonne j d'une matrice identité ($\delta_{ij} = 1$, si $i = j$, sinon $\delta_{ij} = 0$);
A^{-1}	l'inverse de la matrice A;
P_σ	la matrice de permutation des composantes d'un vecteur induite par la modification de l'ordre des automates définie par σ, dont les éléments sont définis par: $p_{[i_1,\dots,i_N][j_1,\dots,j_N]\sigma} = \begin{cases} 1 & \text{si } j_m = i_{\sigma_m} \text{ avec } m \in [1..N] \\ 0 & \text{sinon} \end{cases}$;
P_σ^T	la transposée et aussi l'inverse de la matrice P_σ.

✍ Suites Finies de Matrices

N	le nombre de matrices d'une suite finie de matrices;
$A^{(i)}$	la i-ème matrice d'une suite de matrices $A^{(1)}, A^{(2)}, \dots, A^{(N-1)}, A^{(N)}$;
$a_{i,j}^{(k)}$	l'élément dans la ligne i et la colonne j de la matrice $A^{(k)}$;
n_i	la dimension de la matrice $A^{(i)}$;
$nleft_i$	le produit des dimensions de toutes les matrices avant $A^{(i)}$, *i.e.*, $\prod_{k=1}^{i-1} n_k$ (cas particulier: $nleft_1 = 1$);
$nright_i$	le produit des dimensions de toutes les matrices après $A^{(i)}$, *i.e.*, $\prod_{k=i+1}^{N} n_k$ (cas particulier: $nright_N = 1$);
$\bar{n}_i$	le produit des dimensions de toutes les matrices sauf $A^{(i)}$, *i.e.*, $\prod_{k=1,k\neq i}^{N} n_k$ ($\bar{n}_i = nleft_i\ nright_i$);
$a_{[i_1,\dots,i_m,],[j_1,\dots,j_m]}$	l'élément de la matrice $A = \otimes_{l=1}^{N}$ de taille $\prod_{l=1}^{m} n_l$ localisé dans le bloc de taille $\prod_{l=2}^{m} n_l$ de coordonnées i_1, j_1, puis à l'intérieur de ce bloc, dans le bloc de taille $\prod_{l=3}^{m} n_l$ de coordonnées i_2, j_2 et ainsi de suite jusqu'à la position i_m, j_m du bloc (plus interne) de taille n_m.

✍ Matrices Fonctionnelles

a_k l'indice de la ligne k de la matrice A, utilisé dans le contexte des RAS aussi comme l'état de l'automate $\mathcal{A}$ auquel la matrice A correspond;

$A(\mathcal{B},\mathcal{C})$ la matrice fonctionnelle A qui possède comme paramètres les matrices B et C;

$a_{i,j}(\mathcal{B},\mathcal{C})$ l'élément fonctionnel i,j de la matrice $A(\mathcal{B},\mathcal{C})$;

$A(b_k,\mathcal{C})$ la matrice fonctionnelle $A(\mathcal{B},\mathcal{C})$ où l'indice de ligne de la matrice B est déjà connu et égal à k (cette matrice peut être considéré comme dépendante de la matrice C seulement);

$a_{i,j}(b_k,\mathcal{C})$ l'élément fonctionnel i,j de la matrice $A(b_k,\mathcal{C})$;

$A(b_k,c_l)$ la matrice fonctionnelle $A(\mathcal{B},\mathcal{C})$ avec les éléments évalués pour les états correspondants aux lignes k et l des matrices B et C respectivement (cette matrice, ayant tous ses paramètres connus, est considérée comme constante);

$a_{i,j}(b_k,c_l)$ l'élément constant (élément fonctionnel évalué) i,j de la matrice $A(b_k,c_l)$;

$\ell_k(A)$ la matrice avec tous les éléments mis à zéro, sauf ceux appartenant à la ligne k qui est égale à la ligne k de la matrice A ($A = \sum_{k=1}^{n_A} \ell_k(A)$).

✍ Opérateurs Matriciels

$A \times B$ le produit ordinaire des matrices A et B (dans cet ordre);

$A + B$ la somme (traditionnelle) des matrices A et B;

$A \otimes B$ le produit tensoriel classique des matrices A et B (dans cet ordre);

$A \oplus B$ la somme tensorielle des matrices carrées A et B;

$A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A})$ le produit tensoriel généralisé entre les matrices $A(\mathcal{B})$ et $B(\mathcal{A})$;

$A(\mathcal{B}) \underset{g}{\oplus} B(\mathcal{A})$ la somme tensorielle généralisée entre les matrices $A(\mathcal{B})$ et $B(\mathcal{A})$.

A.2 Réseaux d'Automates Stochastiques

Cette notation a été introduite au chapitre 3.

✍ **Automates**

$\mathcal{A}^{(i)}$	le i-ème automate d'un RAS;
n_i	le nombre des états de $S^{(i)}$ ($n_i = cardinal(S^{(i)})$);
$S^{(i)}$	l'ensemble des états (locaux) de l'automate $\mathcal{A}^{(i)}$;
$x^{(i)}$	un état (local) de l'automate $\mathcal{A}^{(i)}$ ($x^{(i)} \in S^{(i)}$);
$Q^{(i)}(x^{(i)}, y^{(i)})$	l'étiquette de transition de l'état local $x^{(i)}$ vers l'état local $y^{(i)}$;
$\tau_l[x^{(i)}, y^{(i)}]$	le taux local de l'étiquette $Q^{(i)}(x^{(i)}, y^{(i)})$;
$succ_l(x^{(i)})$	l'ensemble des états $y^{(i)}$ tels que le taux $\tau_l[x^{(i)}, y^{(i)}]$ est non identiquement nul.

✍ **Événements Synchronisants**

$\iota^{(e)}$	l'indice de l'automate maître de l'événement e;
$\tau_e[x^{(i)}, y^{(i)}]$	le taux synchronisant du triplet de synchronisation (e, τ_e, π_e) associée à la transition $Q^{(i)}(x^{(i)}, y^{(i)})$;
$\pi_e[x^{(i)}, y^{(i)}]$	la probabilité de routage du triplet de synchronisation (e, τ_e, π_e) associée à la transition $Q^{(i)}(x^{(i)}, y^{(i)})$;
$succ_e(x^{(i)})$	l'ensemble des états $y^{(i)}$ tels que l'étiquette $Q^{(i)}(x^{(i)}, y^{(i)})$ possède un triplet de synchronisation avec l'identificateur e et $\tau_e[x^{(i)}, y^{(i)}] \neq 0$, $\pi_e[x^{(i)}, y^{(i)}] \neq 0$ (éléments fonctionnels non identiquement nuls);
$\eta^{(e)}$	l'ensemble des indices i ($i \in [1..N]$), tel que l'automate $\mathcal{A}^{(i)}$ possède dans au moins une des étiquettes de $Q^{(i)}$ un triplet de synchronisation avec l'identificateur de l'événement e.

✍ **Réseaux d'Automates**

N	le nombre d'automates du réseau;
ε	l'ensemble des identificateurs d'événements synchronisants;
S	l'espace d'état du RAS défini comme $\prod_{i=1}^{N} S^{(i)}$;
R	le sous-ensemble de S comprenant tous les états $\tilde{x}$ tels que $F(\tilde{x}) = 1$.

✍ États Globaux

$\tilde{x}$ l'état global (combinaison d'états locaux) d'un RAS à N automates, $\tilde{x} = (x^{(1)}, \ldots, x^{(N)})$ où $x^{(i)}$ est l'état local de l'automate $\mathcal{A}^{(i)}$ ($\tilde{x} \in S$);

$\tilde{x}(x^{(i)} \xrightarrow{l} y^{(i)})$ l'état global obtenu en remplaçant l'état local $x^{(i)}$ par l'état local $y^{(i)}$ dans l'automate $\mathcal{A}^{(i)}$;

$x^{(\omega)}$ la composition des états locaux $x^{(i)}$ où $i \in \omega$, avec $\omega \subset [1..N]$;

$\tilde{x}(x^{(i)} \xrightarrow{l} y^{(i)})$ l'état global obtenu en remplaçant l'état local $x^{(i)}$ par l'état local $y^{(i)}$ dans l'automate $\mathcal{A}^{(i)}$ ($y^{(i)} \in succ_l(x^{(i)})$);

$\tilde{x}(x^{(i)} \xrightarrow{i \in \eta^{(e)}} y^{(i)})$ l'état global obtenu en remplaçant tous les états locaux $x^{(i)}$ par $y^{(i)}$ dans tous les automates $\mathcal{A}^{(i)}$ ($i \in \eta^{(e)}$ et $y^{(i)} \in succ_e(x^{(i)})$);

✍ Éléments Fonctionnels

$f(x^{(\omega)})$ l'élément fonctionnel $f(\mathcal{A}^{(\omega)})$ évalué pour les états locaux $x^{(\omega)}$;

$\mathcal{A}^{(i)}(\mathcal{A}^{(\omega)})$ l'automate $\mathcal{A}^{(i)}$ qui possède comme paramètres les automates $\mathcal{A}^{(j)}$ avec $j \in \omega$;

$\mathcal{A}^{(i)}(x^{(\omega)})$ l'automate $\mathcal{A}^{(i)}(\mathcal{A}^{(\omega)})$ avec tous ses éléments fonctionnels évalués pour la composition des états locaux $x^{(\omega)}$).

✍ Descripteur Markovien

$Q_l^{(i)}$ la matrice regroupant tous les taux de transitions locaux de l'automate $\mathcal{A}^{(i)}$;

$Q_{e^+}^{(i)}$ la matrice regroupant tous les probabilités et taux exprimés dans les triplets de synchronisation de l'automate $\mathcal{A}^{(i)}$ faisant référence à l'événement $e \in \varepsilon$;

$Q_j^{(i)}(x^{(i)}, y^{(i)})$ l'élément de la matrice $Q_j^{(i)}$ dans la ligne $x^{(i)}$ et la colonne $y^{(i)}$, avec $i \in [1..N]$ et $j \in \{l, e^+, e^-\}$;

I_{n_i} la matrice identité de taille n_i, avec $i \in [1..N]$;

Q le générateur Markovien correspondant à la chaîne de Markov associée à un RAS *bien défini*;

$Q(\tilde{x}, \tilde{y})$ l'élément du générateur Markovien d'un RAS correspondant à la transition de l'état global $\tilde{x}$ vers l'état global $\tilde{y}$.

A.3 Paramètres des Modèles RAS

Cette notation a été introduite au chapitre 4.

✍ **Modèles de Partage de Ressources**

N	nombre de clients du modèle;
P	nombre de ressources du modèle;
B	nombre de automates groupés du descripteur correspondant au modèle[1];
λ_i	taux des requêtes du client i;
μ_i	taux des libérations de ressources prises par le client i.

✍ **Modèles de Réseau Fermé de Files d'Attente**

N	nombre de files d'attente;
P	nombre de clients du modèle;
s_i	la i-ème file du modèle;
$i \mapsto j$	nom de l'événement représentant le départ d'un client de la file s_i vers la file s_j;
μ_i	taux de service de la file s_i;
p_{ij}	probabilité d'un client parti de la file s_i aller la file s_j.

✍ **Modèles de Réseau Ouvert de Files d'Attente**

K	nombre de classes distincts de clients du modèle;
C	nombre de files d'attente du modèle;
s_i	la i-ème file d'attente du modèle;
C_i	capacité de file d'attente s_i;
λ_i^k	le taux exponentiel d'arrivée des clients de la classe k venus de l'extérieur vers la file s_i;
μ_i^k	le taux exponentiel de service des clients de la classe k dans la file s_i;
$j \overset{k}{\mapsto} i$	nom de l'événement représentant le départ d'un client de la classe k de la file s_i vers la file s_j;
p_{ij}	probabilité d'un client parti de la file s_i aller la file s_j;
$p_{i\chi}$	probabilité d'un client parti de la file s_i sortir vers l'extérieur.

[1] Définition introduite lors de mesures numériques du modèle de partage de ressources.

A.4 Méthodes de Solution

Cette notation a été introduite au chapitre 6.

✍ **Méthodes de Projection**

$\mathcal{K}_m$ un espace de Krylov [113] de taille m;

$\mathcal{K}_m(A, v_1)$ un espace de Krylov défini par une matrice carré A et un vecteur v_1 de norme 1 ($\mathcal{K}_m(A, v_1) = span\left\{v_1, v_1A, v_1A^2, v_1A^3, \ldots, v_1A^{m-1}\right\}$);

v_j le j-ème vecteur de l'espace $\mathcal{K}_m(A, v_1)$;

V_m la matrice de dimension $n_A \times m$ où chaque colonne j est le vecteur v_j ($j \in [1..m]$) de l'espace $\mathcal{K}_m(A, v_1)$;

H_m une matrice carrée de dimension m avec tous les éléments non nuls situés au dessus de la deuxième diagonale inférieure, ***i.e.***, une matrice d'Hessenberg supérieure;

$\bar{H}_m$ la matrice de dimension $(m+1) \times m$ obtenue en ajoutant à H_m la ligne $[0, 0, \ldots, 0, h_{m+1,m}]$;

G_i la matrice de dimension m et de rang 2 pour faire la rotation de Givens [43] qui élimine l'élément $h_{i+1,i}$.

✍ **Techniques de Pré-conditionnement**

L une matrice triangulaire inférieure résultante de la décomposition LU d'un matrice quelconque;

U une matrice triangulaire supérieure résultante de la décomposition LU d'un matrice quelconque;

$L^{(i)}$ la matrice L d'une décomposition LU de la matrice $A^{(i)}$ d'une suite finie des matrices;

$U^{(i)}$ la matrice U d'une décomposition LU de la matrice $A^{(i)}$ d'une suite finie des matrices.

$\mathfrak{Q}$ le descripteur obtenu avec l'élimination de tous les éléments fonctionnels du descripteur Q en les remplaçant par leur évaluation moyenne et avec l'application d'une des trois polices de régulation:

- sans régulation;
- régulation avec translation (***shift***);
- régulation avec transformation de Winglet;

$\mathfrak{M}^{(i)}$ la matrice regroupant le i-ème terme de la somme du descripteur $\mathfrak{Q}$ ($\mathfrak{Q} \equiv \sum_{i=1}^{(N+2E)} \mathfrak{M}^{(i)}$);

$\mathfrak{L}^{(i)}$ la matrice triangulaire inférieure résultant de la décomposition LU de la matrice $\mathfrak{M}^{(i)}$;

$\mathfrak{U}^{(i)}$ la matrice triangulaire supérieure résultant de la décomposition LU de la matrice $\mathfrak{M}^{(i)}$ ($\mathfrak{M}^{(i)} \equiv \left[\mathfrak{L}^{(i)}\mathfrak{U}^{(i)}\right]$).

Annexe B

Tableaux de Mesures Numériques

Conditions de Mesure et Lecture des Tableaux

La totalité des expériences ont été faites avec le logiciel PEPS (chapitre 7). Tous les temps d'exécution (colonne *sec.*) ont été mesurés avec une précision du dixième de seconde sur une station IBM RS6000 avec système AIX 4.2 et 96 Mega octets de mémoire vive. La convergence a été vérifiée avec une précision absolue à la dixième décimale, *i.e.*, les résultats ont une tolérance de l'ordre de 1^{-10}. Les vecteurs initiaux de toutes expériences sont des vecteurs équiprobables.

Les méthodes d'Arnoldi et GMRES utilisées sont les versions avec redémarrage à chaque 10, 20 et 30 pas, à cause des limitations de mémoire et temps d'exécution. Les nombres d'itérations fournies (colonne *it.*) font état du nombre de pas internes de chaque méthode et non du nombre de redémarrages exécutées. Les données des convergences présentant de bons rapports *vitesse de convergence / utilisation mémoire* sont marqués en gras. Nous considérons une méthode ayant un bon rapport, les méthodes qui ont les convergences les plus rapides (moins de temps d'exécution) si comparées à des méthodes n'utilisant pas plus de mémoire qu'elles. Par exemple, l'application de la méthode GMRES avec redémarrage à chaque 20 pas aura un bon rapport si sa convergence est plus rapide que l'application de tous les cas des méthodes:

- de la puissance (utilisant un vecteur);
- d'Arnoldi utilisant 10 vecteurs;
- GMRES utilisant 10 vecteurs;
- d'Arnoldi utilisant 20 vecteurs;
- GMRES utilisant 20 vecteurs.

De plus, les donnés de la convergence la plus rapide toutes méthodes confondues sont marqués en gras et encadrés.

Tous les modèles utilisent les optimisations citées dans le chapitre 5. Notamment le groupement d'automates est utilisé et le nombre de groupes (automates groupés) est indiqué pour chaque modèle. Les résultats d'expériences avec la mention – indiquent des expériences où la méthode a stagné ou bien où le pré-conditionnement à beaucoup ralenti la convergence.

Pour chaque exemple, les méthodes de la puissance, d'Arnoldi et GMRES sont exécutées avec les types de pré-conditionnements suivants:

nop sans pré-conditionnement;

pia pré-conditionnement polynômial par l'inverse approchée;

pyt pré-conditionnement polynômial translaté;

alt pré-conditionnement alterné;

adt pré-conditionnement additif;

mlt pré-conditionnement multiplicatif;

dia pré-conditionnement diagonal.

La taille des polynômes pour les pré-conditionnements polynômiaux (cas *pia* et *pyt*) sont:

$k = 3$ 3 termes;

$k = 5$ 5 termes;

Les options de régulation pour les pré-conditionnements utilisant des décomposition LU (cas *alt*,*mlt* et *dia*) sont:

nor sans régulation;

rts régulation par translation standard (*shift*);

rtw régulation par translation de Winglet.

B.1 Modèles de Partage de Ressources

Ces modèles sont définis selon les paramètres presentés à la section 4.1. Deux groupes de modèles distincts ont été choisis:

– modèles avec taux distincts pour chacun des clients;

– modèles avec le même taux pour tous les clients;

Le modèles du premier groupe présentent une convergence plus lente, tandis que les exemples du deuxième groupe sont plus rapides à converger. Pour les modèles du premier groupe les paramètres suivants ont été choisis:

– taux de requête du i-ème client:

$$\lambda_i = 2i;$$

– taux de libération du i-ème client:

$$\mu_i = i;$$

Pour les modèles du deuxième groupe tous les taux de requête sont égaux à une unité ($\lambda_i = 1$). Tandis que les taux de libération sont égaux à 0.4 ($\mu_i = 0.4$).

Pour la totalité des modèles le nombre de clients est indiqué par le paramètre N, le nombre de ressources est indiqué par le paramètre P. Le nombre d'automates groupés est indiqué par le paramètre B. Par exemple un modèle référencé par $N = 12, P = 8, B =$

2 représente un modèle avec 12 clients et 8 ressources groupé en 2 automates (chacun représentant 6 clients).

Les exemples des modèles de partage de ressources présentés sont indiqués par un numéro composé de trois parties. La première partie indique le type de modèle (toujours égal à 1 pour les modèles de partage de ressources). La deuxième partie indique l'ensemble de paramètres N, P et B selon la liste suivante:

1. $N = 12, P = 12, B = 4$;
2. $N = 12, P = 12, B = 2$;
3. $N = 12, P = 11, B = 4$;
4. $N = 12, P = 11, B = 2$;
5. $N = 12, P = 8, B = 4$;
6. $N = 12, P = 8, B = 2$;
7. $N = 12, P = 4, B = 4$;
8. $N = 12, P = 4, B = 2$;

La troisième partie indique les taux utilisées, avec 1 pour les modèles avec taux distincts et 2 pour les modèles avec le même taux pour tous les clients.

. . .

Exemple 1.1.1 - Modèles de Partage de Ressources - Taux distincts - $N = 12, P = 12, B = 4$

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**662**	**16.2**	100	3.2	82	3.1	62	2.7	107	3.4	78	2.9	66	2.9
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	546	39.2	77	6.0	58	4.9	50	4.4	78	6.1	58	4.8	51	4.5
pyt - $k = 5$	461	54.3	70	8.6	60	7.8	51	6.8	77	9.5	58	7.5	50	6.7
alt - nor	582	18.7	198	7.2	144	6.3	138	6.7	190	7.2	145	6.3	126	6.1
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	109	6.3	83	5.3	74	5.1	108	6.2	80	5.1	71	4.8
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	437	39.9	–	–	–	–	–	–
mlt - nor	–	–	380	45.9	255	32.3	195	25.8	360	43.4	240	30.5	180	23.9
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	**88**	**2.9**	75	2.9	57	2.5	89	2.9	**74**	**2.8**	**55**	**2.4**

. . .

Exemple 1.1.2 - Modèles de Partage de Ressources - Taux identiques - $N = 12, P = 12, B = 4$

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**148**	**3.6**	14	0.4	13	0.4	13	0.4	30	1.0	13	0.4	13	0.4
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	–	–	25	1.9	18	1.5	18	1.5	27	2.1	18	1.4	18	1.4
pyt - $k = 5$	–	–	30	3.7	18	2.4	18	2.3	30	3.7	18	2.3	18	2.3
alt - nor	152	5.0	89	3.4	71	3.1	57	2.8	97	3.7	69	3.0	58	2.9
alt - rts	–	–	–	–	112	5.4	83	4.5	–	–	–	–	60	3.3
alt - rtw	–	–	–	–	–	–	84	4.6	–	–	–	–	60	3.3
adt - nor	–	–	38	2.2	33	2.0	29	2.1	38	2.2	33	2.0	29	2.1
adt - rts	–	–	293	23.1	113	9.7	79	7.1	–	–	60	5.1	–	–
adt - rtw	–	–	130	10.3	110	9.3	58	5.2	–	–	60	5.1	30	2.8
mlt - nor	–	–	400	48.7	180	23.0	119	16.1	–	–	156	20.0	117	15.5
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	**10**	**0.3**	10	0.3	10	0.3	10	0.3	10	0.3	10	0.3

. . .

Exemple 1.2.1 - Modèles de Partage de Ressources - Taux distincts - $N = 12, P = 12, B = 2$

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	662	13.2	100	2.6	82	2.7	62	2.4	107	2.8	78	2.5	66	2.5
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	572	33.7	78	5.0	62	4.3	53	4.0	82	5.2	60	4.2	53	4.0
pyt - $k = 5$	503	47.8	75	7.6	62	6.6	52	5.7	79	8.0	58	6.2	51	5.7
alt - nor	**258**	**11.7**	159	7.8	138	7.6	141	8.8	170	8.3	147	8.2	119	7.5
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	40	4.9	60	7.6
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	**87**	**2.4**	75	2.5	57	2.3	89	2.5	**74**	**2.4**	**55**	**2.2**

. . .

Exemple 1.2.2 - Modèles de Partage de Ressources - Taux identiques - $N = 12, P = 12, B = 2$

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**148**	**2.9**	14	0.3	13	0.4	13	0.4	30	0.8	13	0.4	13	0.4
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	–	–	25	1.6	18	1.2	18	1.3	27	1.7	18	1.3	18	1.3
pyt - $k = 5$	–	–	30	3.0	18	1.9	18	1.9	29	2.9	18	1.9	18	1.9
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	49	3.6	37	2.9	50	4.3	50	4.0	28	2.2	45	7.8
alt - rtw	–	–	59	4.2	32	2.5	47	4.0	60	4.5	28	2.2	38	3.2
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	80	10.2	47	6.1	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	30	4.1
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	40	5.9	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	40	6.0	60	9.2
dia	–	–	**10**	**0.2**	10	0.3	10	0.3	10	0.3	10	0.3	10	0.3

. . .

Exemple 1.3.1 - Modèles de Partage de Ressources - Taux distincts - $N = 12, P = 11, B = 4$

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**668**	**134.2**	106	21.6	82	18.3	68	15.5	107	21.8	78	17.4	66	15.0
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	554	326.9	77	45.6	59	37.6	52	33.6	79	46.3	60	38.4	52	33.7
pyt - $k = 5$	469	489.7	74	72.0	57	59.9	46	49.1	76	74.0	58	61.5	46	49.3
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	500	143.6	450	118.8
adt - rts	–	–	–	–	–	–	–	–	–	–	480	143.3	420	119.5
adt - rtw	–	–	–	–	–	–	–	–	–	–	440	125.9	390	102.9
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	96	19.9	80	18.3	64	15.0	**95**	**19.6**	**76**	**17.1**	**64**	**14.8**

...

Exemple 1.3.2 - Modèles de Partage de Ressources - Taux identiques - $N = 12, P = 11, B = 4$

...

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**130**	**26.5**	30	6.9	**12**	**2.6**	12	2.6	25	5.8	12	2.6	12	2.5
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 2$	130	82.5	27	18.2	20	12.4	20	11.7	25	16.6	19	11.8	19	11.2
pyt - $k = 5$	126	131.9	27	29.5	20	20.4	20	19.1	26	28.2	20	20.4	20	19.1
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	110	29.8	90	23.4	63	16.4	–	–	60	15.9	30	7.8
adt - rts	–	–	118	32.8	99	28.4	71	18.7	80	22.2	60	16.9	60	15.7
adt - rtw	–	–	127	36.6	119	32.8	62	16.2	–	–	60	17.3	60	15.7
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	**19**	**4.4**	12	2.6	**12**	**2.5**	19	4.5	12	2.6	12	2.6

. . .

Exemple 1.4.1 - Modèles de Partage de Ressources - Taux distincts - $N = 12, P = 11, B = 2$

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**668**	**35.2**	106	6.5	82	5.6	68	5.0	107	6.9	78	5.5	66	5.0
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	580	100.5	79	13.9	66	11.5	53	9.9	84	14.3	63	11.2	54	9.9
pyt - $k = 5$	511	135.8	78	20.7	62	17.2	51	14.7	78	20.6	59	16.4	51	14.7
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	510	61.9
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	420	52.4
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	**96**	**6.2**	80	5.8	64	5.0	95	6.3	**76**	**5.5**	**64**	**5.0**

...

Exemple 1.4.2 - Modèles de Partage de Ressources - Taux identiques - $N = 12, P = 11, B = 2$

...

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**130**	**7.2**	30	1.9	**12**	**0.7**	12	0.7	**25**	**1.6**	12	0.7	12	0.7
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	–	–	27	4.6	20	3.4	20	3.5	26	4.3	20	3.3	20	3.5
pyt - $k = 5$	–	–	27	7.2	19	5.3	19	5.4	27	7.4	19	5.3	19	5.4
alt - nor	–	–	110	12.2	36	4.0	25	2.9	109	12.2	36	4.2	25	2.9
alt - rts	–	–	99	11.2	36	4.0	25	2.9	99	10.9	36	4.0	25	3.0
alt - rtw	–	–	90	10.1	36	4.1	25	2.9	100	11.2	36	4.1	25	2.8
adt - nor	–	–	110	17.7	57	9.5	39	6.6	60	9.9	40	6.7	39	6.5
adt - rts	–	–	–	–	68	11.0	50	8.2	80	12.8	40	6.5	47	7.8
adt - rtw	–	–	–	–	55	8.9	42	7.0	100	15.8	40	6.6	46	10.4
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	96	6.3	80	5.6	64	4.9	95	6.4	76	5.6	64	5.0

...

Exemple 1.5.1 - Modèles de Partage de Ressources - Taux distincts - $N = 12, P = 8, B = 4$

...

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**679**	**475.3**	117	79.3	95	66.8	78	56.0	116	83.0	**76**	**52.0**	73	50.2
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 5$	550	1903.3	96	337.2	75	251.5	63	203.0	95	331.7	72	240.2	62	199.9
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	105	72.6	79	56.7	67	48.9	**96**	**69.6**	75	52.1	**66**	**46.2**

...

Exemple 1.5.2 - Modèles de Partage de Ressources - Taux identiques - $N = 12, P = 8, B = 4$

...

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**59**	**43.9**	**9**	**6.8**	**9**	**6.6**	**9**	**5.8**	9	6.8	9	6.6	9	5.9
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	–	–	25	53.8	19	38.4	19	36.8	24	51.8	19	38.6	19	36.7
pyt - $k = 5$	–	–	27	95.7	18	59.4	18	57.8	26	92.3	18	59.4	18	57.7
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	9	6.8	9	6.7	9	5.9	9	6.8	9	6.6	9	5.8

...

Exemple 1.6.1 - Modèles de Partage de Ressources - Taux distincts - $N = 12, P = 8, B = 2$

...

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**679**	**434.2**	117	78.0	95	64.3	78	52.8	116	85.8	76	51.7	73	47.8
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	639	1322.8	89	184.8	77	186.1	66	157.3	93	199.5	72	179.6	64	161.5
pyt - $k = 5$	584	1867.5	88	271.6	77	238.1	64	227.8	94	290.4	73	225.5	63	223.7
alt - nor	–	–	–	–	–	–	–	–	–	–	620	439.2	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	660	539.0
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	660	538.4
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	**105**	**70.8**	79	54.2	67	45.7	96	71.5	**75**	**51.2**	**66**	**43.4**

...

Exemple 1.6.2 - Modèles de Partage de Ressources - Taux identiques - $N = 12, P = 8, B = 2$

...

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**59**	**38.5**	9	5.6	9	5.4	9	5.5	9	5.6	9	5.6	9	5.5
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	–	–	28	47.6	19	34.3	19	30.9	25	42.0	19	34.4	19	30.9
pyt - $k = 5$	–	–	27	75.8	19	58.3	19	51.2	27	75.8	19	58.1	19	51.3
alt - nor	–	–	–	–	59	37.7	45	27.8	–	–	–	–	45	27.9
alt - rts	–	–	–	–	58	44.2	46	35.1	–	–	58	42.9	46	32.4
alt - rtw	–	–	–	–	58	40.4	46	35.6	–	–	58	40.6	46	35.7
adt - nor	–	–	–	–	–	–	59	48.1	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	9	5.6	9	5.6	**9**	**5.3**	**9**	**5.4**	9	5.6	9	5.3

. . .

Exemple 1.7.1 - Modèles de Partage de Ressources - Taux distincts - $N = 12, P = 4, B = 4$

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**978**	**688.4**	142	96.5	93	65.7	87	62.3	130	92.8	90	61.2	77	52.8
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	920	3194.6	88	187.3	74	168.1	67	132.4	102	217.9	66	134.6	64	126.6
pyt - $k = 5$	–	–	90	318.4	68	230.2	59	192.1	89	337.5	67	228.1	58	186.6
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	337	254.4	215	155.7	207	162.3	200	144.9	180	135.2	150	102.2
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	160	135.1	152	123.3	137	106.3	140	125.7	120	102.3	120	93.2
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	**75**	**54.1**	67	46.3	**54**	**37.5**	80	58.0	**63**	**43.5**	56	38.9

...

Exemple 1.7.2 - Modèles de Partage de Ressources - Taux identiques - $N = 12, P = 4, B = 4$

...

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**21**	**14.9**	**5**	**3.7**	**5**	**3.6**	**5**	**3.2**	5	3.7	5	3.6	5	3.2
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	–	–	17	36.2	14	27.9	14	26.7	18	38.3	14	27.9	14	26.7
pyt - $k = 5$	–	–	20	70.3	14	45.7	14	44.5	19	66.8	14	45.7	14	44.5
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	5	3.7	5	3.6	5	3.2	5	3.7	5	3.6	5	3.2

...

Exemple 1.8.1 - Modèles de Partage de Ressources - Taux distincts - $N = 12, P = 4, B = 2$

...

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**978**	**513.8**	142	76.9	93	53.0	87	51.9	130	81.1	90	49.1	77	45.7
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	943	2828.1	89	162.1	74	134.1	67	130.8	102	179.5	67	116.9	64	110.9
pyt - $k = 5$	–	–	110	321.8	73	212.5	67	193.3	105	306.4	73	211.9	62	179.0
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	690	530.7
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	**75**	**46.7**	67	38.6	**54**	**33.9**	80	50.2	**63**	**36.2**	56	36.8

...

Exemple 1.8.2 - Modèles de Partage de Ressources - Taux identiques - $N = 12, P = 4, B = 2$

...

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**21**	**11.9**	5	3.0	**5**	**2.9**	5	2.5	5	3.0	5	2.9	**5**	**2.5**
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	–	–	19	31.0	14	22.8	14	19.8	18	29.3	14	22.8	14	19.8
pyt - $k = 5$	–	–	–	–	14	37.44	14	32.8	19	51.2	14	37.5	14	32.9
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	5	3.0	5	2.9	5	2.6	**5**	**2.9**	5	2.9	5	2.6

B.2 Modèles de Réseau Fermé avec Trois Files d'Attente

Ces modèles sont définis selon les paramètres définis à la section 4.2. Nous avons choisis cinq modèles avec 10 clients en trois configurations distinctes selon les probabilités de routage et taux de service choisis:

– système équilibré, *i.e.*, approximativement le même rapport entre la probabilité de visite et le taux de service de chacune des files:

$mu_1 = 10$, $mu_2 = 5$, $mu_3 = 8$, $\pi_1 = 0.2$, $\pi_1 = 0.8$, $\pi_1 = 0.5$ et $\pi_1 = 0.5$;

– surcharge (concentration des clients) sur la deuxième file;

$mu_1 = 10$, $mu_2 = 5$, $mu_3 = 4$, $\pi_1 = 0.7$, $\pi_1 = 0.3$, $\pi_1 = 0.6$ et $\pi_1 = 0.4$;

– surcharge sur la troisième file;

$mu_1 = 10$, $mu_2 = 5$, $mu_3 = 4$, $\pi_1 = 0.2$, $\pi_1 = 0.8$, $\pi_1 = 0.6$ et $\pi_1 = 0.4$;

– faible charge sur la deuxième file;

$mu_1 = 10$, $mu_2 = 10$, $mu_3 = 10$, $\pi_1 = 0.2$, $\pi_1 = 0.8$, $\pi_1 = 0.6$ et $\pi_1 = 0.4$;

– faible charge sur la troisième file.

$mu_1 = 10$, $mu_2 = 10$, $mu_3 = 10$, $\pi_1 = 0.5$, $\pi_1 = 0.5$, $\pi_1 = 0.5$ et $\pi_1 = 0.5$;

Pour la totalité des modèles le nombre de clients dans le réseau est égal à 10. Le nombre d'automates est toujours égal à 3 (aucun groupement est exécuté).

Les exemples des modèles de réseau fermé présentés sont indiqués par un numéro composé de deux parties. La première partie indique le type de modèle (toujours égal à 2 pour modèles de réseau fermé). La deuxième partie indique les probabilités et taux utilisés selon la liste suivante:

1. système équilibré;
2. surcharge sur la deuxième file;
3. surcharge sur la troisième file;
4. faible charge sur la deuxième file;
5. faible charge sur la troisième file.

. . .

Exemple 2.1 - Réseau Fermé avec Trois Files d'Attente - Système Equilibré

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**1010**	**6.6**	**126**	**1.0**	**90**	**0.8**	**76**	**0.7**	147	1.1	95	0.8	77	0.7
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	640	12.9	80	1.6	57	1.3	46	1.0	85	1.8	54	1.2	45	1.0
pyt - $k = 5$	–	–	120	4.0	80	2.8	60	2.2	113	3.8	77	2.7	59	2.1
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	–	–	–	–	–	–	–	–	–	–	–	–

. . .

Exemple 2.2 - Réseau Fermé avec Trois Files d'Attente - Surcharge sur la Deuxième File

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**457**	**3.0**	100	0.8	77	0.7	**58**	**0.5**	**93**	**0.8**	**74**	**0.7**	57	0.6
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	309	6.2	87	1.8	54	1.2	45	1.0	69	1.4	54	1.2	45	1.0
pyt - $k = 5$	–	–	76	2.6	78	2.8	59	2.2	9.0	3.1	76	2.7	56	2.0
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	–	–	–	–	–	–	–	–	–	–	–	–

. . .

Exemple 2.3 - Réseau Fermé avec Trois Files d'Attente - Surcharge sur la Troisième File

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**528**	**3.5**	104	0.8	85	0.8	62	0.6	**95**	**0.8**	**82**	**0.7**	**60**	**0.6**
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	362	7.3	84	1.8	59	1.3	49	1.1	79	1.7	58	1.3	49	1.1
pyt - $k = 5$	–	–	–	–	82	2.9	66	2.4	95	3.2	83	2.9	63	2.3
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	–	–	–	–	–	–	–	–	–	–	–	–

. . .

Exemple 2.4 - Réseau Fermé avec Trois Files d'Attente - Faible Charge pour la Deuxième File

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**410**	**2.7**	99	0.8	80	0.7	**60**	**0.6**	**80**	**0.6**	79	0.7	62	0.6
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	276	5.6	65	1.4	53	1.2	43	1.0	59	1.3	55	1.2	43	1.0
pyt - $k = 5$	–	–	82	2.8	71	2.5	56	2.0	84	2.9	72	2.5	56	2.0
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	–	–	–	–	–	–	–	–	–	–	–	–

. . .

Exemple 2.5 - Réseau Fermé avec Trois Files d'Attente - Faible Charge pour la Troisième File

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**723**	**4.8**	117	0.9	80	0.7	**62**	**0.6**	**99**	**0.8**	**77**	**0.7**	62	0.6
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	540	10.3	74	1.6	53	1.2	37	0.8	75	1.6	54	1.2	35	0.8
pyt - $k = 5$	–	–	122	4.2	80	2.8	57	2.1	125	4.3	76	2.7	55	2.0
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	–	–	–	–	–	–	–	–	–	–	–	–

B.3 Modèles de Réseau Ouvert avec Trois Files d'Attente

Ces modèles sont définis selon les paramètres présentés à la section 4.3.2. Nous avons choisis deux types de modèles:

- avec taux d'arrivée et de service amenant à une distribuition presque uniforme des clients sur les files;
- avec taux d'arrivée et de service amenant à une saturation de la première file.

Les modèles du premier type (distribution presque uniforme des clients) ont les paramètres suivants:

- taux d'arrivée des clients de la classe 1 dans la file s_1:
  $\lambda_1^1 = 1$;
- taux d'arrivée des clients de la classe 2 dans la file s_2:
  $\lambda_2^2 = 2$;
- taux de service dans la file s_1:
  $\mu_1^1 = 3$;
- taux de service dans la file s_2:
  $\mu_2^2 = 2$;
- taux de service dans la file s_3 pour des clients de la classe 1:
  $\mu_3^1 = 2$;
- taux de service dans la file s_3 pour des clients de la classe 2:
  $\mu_3^2 = 2$;

Les modèles du deuxième type (saturation de la première file) ont les paramètres suivants:

- taux d'arrivée des clients de la classe 1 dans la file s_1:
  $\lambda_1^1 = 1$;
- taux d'arrivée des clients de la classe 2 dans la file s_2:
  $\lambda_2^2 = 2$;
- taux de service dans la file s_1:
  $\mu_1^1 = 3$;
- taux de service dans la file s_2:
  $\mu_2^2 = 4$;
- taux de service dans la file s_3 pour des clients de la classe 1:
  $\mu_3^1 = 5$;
- taux de service dans la file s_3 pour des clients de la classe 2:
  $\mu_3^2 = 6$;

La capacité de chacune des files s_i est indiquée avec les paramètres C_i et tous les modèles sont groupés en deux automates, le premier regroupant l'automate $\mathcal{A}^{(1)}$ et $\mathcal{A}^{(2)}$ et le deuxième regroupant les automates $\mathcal{A}^{(3_1)}$ et $\mathcal{A}^{(3_2)}$. Par exemple un modèle référencé

$C_1 = 7, C_2 = 9, C_3 = 11$ représente un modèle où la file s_1 peut avoir entre 0 et 6 clients, la file s_2 peut avoir entre 0 et 8 clients, la file s_3 peut avoir entre 0 et 10 clients.

Les exemples des modèles de réseau ouvert présentés sont indiqués par un numéro composé de trois parties. La première partie indique le type de modèle (toujours égal à 3 pour modèles de réseau ouvert). La deuxième partie indique l'ensemble de paramètres C_1, C_2 et C_3 selon la liste suivante:

1. $C_1 = 7, C_2 = 9, C_3 = 11$;
2. $C_1 = 8, C_2 = 8, C_3 = 12$;
3. $C_1 = 10, C_2 = 10, C_3 = 10$;

La troisième partie indique l'ensemble des taux utilisés, avec 1 pour les modèles avec le même taux de service pour toutes les files et 2 pour les modèles avec taux de service distincts.

. . .

Exemple 3.1.1 - Réseau Ouvert avec Trois Files d'Attente - distribution presque uniforme des clients - $C_1 = 7, C_2 = 9, C_3 = 11$

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**1258**	**27.8**	283	7.6	197	7.0	216	9.0	280	8.1	220	7.8	233	9.8
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	1062	70.9	160	11.6	164	12.9	133	11.2	180	12.8	155	12.3	133	11.2
pyt - $k = 5$	–	–	395	45.5	273	32.8	303	38.7	410	46.8	279	34.0	282	35.9
alt - nor	–	–	–	–	–	–	850	38.4	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	**216**	**6.4**	192	7.1	192	8.2	239	7.4	215	7.9	194	8.3

...

Exemple 3.1.2 - Réseau Ouvert avec Trois Files d'Attente - saturation de la première file - $C_1 = 7, C_2 = 9, C_3 = 11$

...

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**476**	**10.5**	182	5.2	218	7.7	177	7.3	–	–	–	–	288	11.8
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	314	21.0	109	7.9	100	7.9	80	6.5	148	10.6	97	7.6	80	6.7
pyt - $k = 5$	–	–	165	18.9	176	21.3	144	18.1	180	20.6	164	19.7	138	17.5
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	**164**	**5.0**	179	6.0	179	7.7	–	–	–	–	227	9.7

. . .

Exemple 3.2.1 - Réseau Ouvert avec Trois Files d'Attente - distribution presque uniforme des clients - $C_1 = 8, C_2 = 8, C_3 = 12$

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**1459**	**36.2**	349	11.9	234	9.6	246	12.1	330	11.3	273	11.3	279	13.6
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	1218	97.1	195	16.7	171	16.0	153	15.4	202	17.4	175	16.3	159	15.6
pyt - $k = 5$	–	–	505	68.7	365	52.6	347	52.3	480	65.4	340	50.2	319	48.0
alt - nor	–	–	–	–	–	–	1170	63.0	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	**287**	**10.2**	205	8.8	212	10.7	300	10.7	228	9.3	240	12.1

. . .

Exemple 3.2.2 - Réseau Ouvert avec Trois Files d'Attente - saturation de la première file - $C_1 = 8, C_2 = 8, C_3 = 12$

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**496**	**12.6**	218	7.5	222	9.2	267	13.0	–	–	–	–	298	14.2
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	295	23.7	100	8.8	117	11.0	85	8.4	134	11.5	119	11.2	82	8.2
pyt - $k = 5$	–	–	199	27.0	172	24.6	156	23.5	184	25.1	192	28.1	172	25.9
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	**195**	**7.0**	220	9.5	177	8.8	–	–	–	–	297	15.0

. . .

Exemple 3.3.1 - Réseau Ouvert avec Trois Files d'Attente - distribution presque uniforme des clients - $C_1 = 10, C_2 = 10, C_3 = 10$

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**1333**	**37.0**	326	11.6	231	10.6	206	11.0	310	11.7	240	11.0	251	13.5
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	1077	94.4	163	15.5	158	16.2	148	16.2	146	13.8	143	14.9	141	15.6
pyt - $k = 5$	–	–	424	63.9	310	49.1	302	50.3	400	60.3	280	44.5	299	50.0
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	**245**	**9.7**	**192**	**9.0**	193	10.6	280	11.1	220	10.3	202	10.9

. . .

Exemple 3.3.2 - Réseau Ouvert avec Trois Files d'Attente - saturation de la première file - $C_1 = 10, C_2 = 10, C_3 = 10$

. . .

pré-condit.	Puissance		Arnoldi						GMRES					
			$m = 10$		$m = 20$		$m = 30$		$m = 10$		$m = 20$		$m = 30$	
	it.	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*	*it.*	*sec.*
nop	**525**	**13.9**	**172**	**6.5**	239	11.0	180	9.7	–	–	–	–	234	12.1
pia - $k = 3$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pia - $k = 5$	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pyt - $k = 3$	357	31.4	124	11.8	100	10.3	85	9.4	256	24.1	102	10.6	83	9.2
pyt - $k = 5$	–	–	215	32.3	158	25.2	168	28.0	185	27.9	176	28.3	160	26.6
alt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
alt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
adt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - nor	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rts	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mlt - rtw	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dia	–	–	320	12.7	217	10.1	209	11.6	–	–	–	–	225	12.3

.

Si c’était à refaire je le ferais autrement,
je crois que ça veut dire que j’ai appris quelque chose ...

Résumé:

Cette thèse propose des techniques numériques visant à optimiser les méthodes itératives d'évaluation de performances de modèles Markoviens. Ces techniques s'appliquent à des modèles où la matrice de transition de la chaîne de Markov associée est stockée sous un format tensoriel. Dans cette thèse le formalisme des réseaux d'automates stochastiques est employé pour la description des modèles.
L'évaluation de performances cherchée est la détermination de l'état stationnaire de la chaîne de Markov. Nous définissons une algèbre tensorielle généralisée et nous démontrons des propriétés qui servent de base aux algorithmes de résolution de la chaîne de Markov. Le principal apport de cette thèse réside dans l'efficacité des ces algorithmes qui sont de type itératif. Ceci est fait à deux niveaux: la réduction du coût de chaque itération et la réduction du nombre d'itérations nécessaire à la convergence.
La multiplication d'un vecteur par une matrice en format tensoriel (appelée *descripteur*) est l'opération de base des itérations. L'efficacité de cette opération est le premier objectif à atteindre. A cet effet, nous proposons une approche (*produit vecteur-descripteur*) qui réduit l'espace mémoire sans augmenter la complexité par rapport à la solution, en général plus gourmande en mémoire, basée sur en stockage de la matrice en format creux.
Le deuxième objectif est l'adaptation des méthodes de la puissance, d'Arnoldi et GMRES dans ses versions standards et pré-conditionnées de façon à minimiser le nombre d'itérations sans trop augmenter le coût de chaque itération. De plus, nos techniques calculent un pré-conditionneur sans calcul explicite de la matrice correspondante au descripteur.
La totalité des concepts introduits est utilisée dans le logiciel *PEPS 2.0*. Plusieurs exemples pratiques de modèles de réseaux d'automates stochastiques ont été évalués avec *PEPS 2.0* pour illustrer les résultats de cette thèse.

Mots clés: Méthodes numériques, Algèbre tensoriel, Méthodes itératives, Évaluation des performances, Réseaux d'automates stochastiques, Produit vecteur-descripteur.

Abstract:

This thesis develops techniques for optimizing the numerical evaluation of Markovian models. These techniques are applied to models in which the transition matrix of the associated Markov chain is stored in a tensor format. A stochastic automata network formalism is used to describe the models.
The performance index of interest is the stationary distribution of the Markov chain. We define a generalized tensor algebra and we prove a number of theorems that allow us to lay the foundations for the algorithms needed to solve the Markov chains. The main objective in this thesis is the efficiency of iterative algorithms for solving Markov chains. This objective has two aspects: the reduction of the computational cost of each iteration step; and the reduction of the number of iterations needed for convergence.
The basic operation in each iteration step is the product of a vector by a matrix stored in a tensor format, the so-called *descriptor*. The efficiency of such operations is our first goal. We propose an approach (*vector-descriptor product*) which reduces memory requirements without increasing the computational complexity of the usual standard sparse matrix approach.
The second goal is the adaptation of the power, Arnoldi and GMRES methods in their standard and preconditioned versions. These implementations attempt to reduce the number of iterations needed for convergence, while avoiding an increase in the iteration step costs. In addition, our technique does not need to explicitly generate the matrix corresponding to the descriptor to provide a preconditioner.
All the concepts developed in this thesis are implemented in the *PEPS 2.0* software package. Several practical examples of stochastic automata network models have been tested with *PEPS 2.0* in order to illustrate the results of this thesis.

Title: Numerical methods to solve Markovian systems with large state space.

Keywords: Numerical methods, Tensor algebra, Iterative methods, Performance evaluation, Stochastic automata networks, Vector-descriptor product.