INFORMATIQUE THÉORIQUE ET APPLICATIONS

C. DELPORTE-GALLET
H. FAUCONNIER
M. NIVAT

Parallélisation d'algorithmes avec un nombre fixe de processeurs

Informatique théorique et applications, tome 24, n° 4 (1990), p. 353-386

<http://www.numdam.org/item?id=ITA_1990__24_4_353_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 24, n° 4, 1990, p. 353 à 386)

PARALLÉLISATION D'ALGORITHMES AVEC UN NOMBRE FIXE DE PROCESSEURS (*)

par C. DELPORTE-GALLET (1), H. FAUCONNIER (1) et M. NIVAT (1)

Communiqué par A. ARNOLD

Résumé. – *On s'intéresse ici au gain de temps que la « parallélisation » d'un problème peut permettre de réaliser. On choisit de considérer que le nombre de processeurs est fixé et indépendant de la taille des entrées.*

Plus que de caractériser les problèmes pour lesquels le gain de temps obtenu par la parallélisation est optimal, le but est surtout de montrer que certains problèmes simples ne peuvent être parallélisés de façon optimale.

Abstract. – *In this paper, we study the speed-up that can be given by the parallelization of a problem. We are interested in the case when the number of processors is fixed and independent of the dimension of the problem.*

The considered problem is to find out if there exists an optimal parallel algorithm, and chiefly to show that some problems are not optimally parallelizable.

1. INTRODUCTION

Actuellement, la technologie permet de produire de plus en plus d'architectures parallèles (par exemple [4], [5], [6], [10]...), aussi le problème se pose de pouvoir les utiliser le plus efficacement possible.

Un certain nombre d'études ont été faites pour paralléliser les problèmes classiques afin d'obtenir des algorithmes optimaux. Or il semble que si pour certains problèmes on peut espérer un gain appréciable lors de la mise en parallèle, d'autres sont intrensèquement séquentiels et on gagnera peu dans leur parallélisation. C'est à la mise en évidence de la classe de ces problèmes et à ses caractéristiques que nous nous sommes intéressés.

(*) Reçu juin 1988, révisé en janvier 1989.
(1) L.I.T.P., Université Paris 7, 2, place Jussieu, 75251 Paris Cedex 05, France.

0988-3754/90/04 353 34/$5.40/

Pour pouvoir comparer les algorithmes parallèles, il faut préciser la mesure de performances : si on obtient un algorithme parallèle dont le temps d'exécution est de T avec p processeurs, alors il est clair que la séquentialisation de cet algorithme permettra d'obtenir un temps de pT pour un seul processeur, aussi pour comparer l'efficacité de plusieurs algorithmes parallèles, une mesure assez naturelle sera le nombre de processeurs utilisés multiplié par le temps de l'algorithme parallèle. Comme pour [8], [3]..., un algorithme parallèle sera optimal si cette mesure est de l'ordre du temps du meilleur algorithme séquentiel connu.

Dans [1], [2], [9]... on obtient une mesure où le nombre de processeurs dépend de la taille des entrées (par exemple l'addition de n nombres est parallélisable de manière optimale avec $n/\log n$ processeurs). Nous avons préféré supposer que le nombre de processeurs était indépendant de la taille des entrées. On peut justifier ce choix de plusieurs manières, d'abord dans l'état actuel de la technologie, le nombre de processeurs est toujours limité (même si cette limite peut être relativement grande), de plus, il est clair que si le nombre de processeurs n'est pas constant, le temps de communication (ou de synchronisation dans le cas de mémoire partagée) n'est plus constant et deviendra non négligeable par rapport au temps de calcul. Aussi des modèles comme les PRAM CRCW ne nous paraissent pas très réalistes.

Nous avons volontairement fait d'autres restrictions en considérant qu'une mesure de temps d'un problème est la taille de ses entrées, et en réduisant fortement la communication (en la restreignant à un partage éventuel des données au début et à la fin de la vie des processeurs).

Dans ce cadre, on considère qu'un problème se parallélise « bien » si, disposant de k processeurs, on arrive à le décomposer en k parties de taille à peu près équivalente dont le traitement par les processeurs résoud le problème.

Étant donné un problème Π dont l'entrée est un vecteur de mots, la « k-décomposition » de Π sera :

(1) la décomposition des entrées en k vecteurs de mots;

(2) le traitement indépendant de chacun des éléments de cette décomposition en parallèle avec un résultat sur un alphabet fini;

(3) le traitement des résultats de l'étape précédente.

Une « bonne k-décomposition » sera telle que la taille de chaque élément de la décomposition soit de l'ordre de grandeur de la taille de l'entrée divisée par k (on dira alors que le problème est k-parallélisable). Comme on considère que le « temps » de calcul d'un processeur est la longueur des entrées de ce processeur, si θ est la longueur de la plus longue entrée de (2), θ représente

le « temps » de la k-décomposition. Aussi, la mesure de l'efficacité de la parallélisation sera $\lambda = \theta k$, correspondant au produit du « temps » de calcul parallèle fois le nombre de processeurs. L'optimalité de la parallélisation correspondra à une valeur de λ égale à la taille des entrées à une constante près.

Une autre mesure de la qualité de la parallélisation est le recouvrement *i. e.* le nombre d'éléments communs aux entrées des processeurs.

On montre que si la k-parallélisation est optimale alors le recouvrement est borné. On peut aussi interpréter ce résultat autrement : si la k-parallélisation n'est pas optimale, alors le traitement du problème nécessitera un échange d'information non borné entre les processeurs (et ce quel que soit le modèle choisi, car on peut toujours coder les messages échangés entre les processus par un recouvrement initial des entrées). En un certain sens, cela indique que si un problème se parallélise mal dans notre modèle, il se parallélisera avec une communication non bornée dans n'importe quel modèle, et donc si on considère que le temps de communication et d'échange entre processeurs est crucial, il se parallélise mal.

Nous avons aussi étudié un certain nombre d'exemples et nous avons obtenu les résultats suivants :

Si A_k est la classe des problèmes sur les mots k-parallélisables alors on montre que A_k est fermée pour les opérations booléennes et que $A_{kk'} \subset A_k \subset A_1$ pour $k' \neq 1$ et $k \neq 1$ (les inclusions étant strictes).

On montre que l'appartenance à un rationnel est k-parallélisable pour tout k.

On montre que le test de l'addition en base quelconque (*i. e.* le test de $\bar{u} + \bar{v} = \bar{w}$ ou $\bar{u}$ est l'entier représenté par u) est parallélisable pour tout k.

Dans une deuxième partie, on généralise ces définitions à des problèmes sur des matrices ce qui permet de traiter des problèmes de traitement d'images, par exemple, on vérifie que l'agrandissement d'images est un problème k-parallélisable ainsi que la recherche d'une figure donnée.

2. PARALLÉLISATION DES PROBLÈMES SUR DES VECTEURS DE MOTS

2.1. Notations

On définit les notations suivantes :

– On notera $\lceil n \rceil$ (resp. $\lfloor n \rfloor$) la partie entière haute (resp. basse) de n.

– $\vec{\mathbf{N}}$ sera un p-vecteur d'entiers; on notera $(\vec{\mathbf{N}})_i$ la i-ième composante du vecteur $\vec{\mathbf{N}}$.

– $\vec{\mathbf{1}}$ sera le p-vecteur dont toutes les composantes sont égales à 1.

– Si u est un mot, $u(m)$ est la m-ième lettre du mot u si elle existe et 1 sinon; $u[X, Y]$ sera le mot $u(X)\ldots u(Y)$ si $X < Y$ et 1 sinon.

– Si A est un alphabet, $\vec{\mathbf{u}}$ sera un p-vecteur de mots sur A avec $\vec{\mathbf{u}} = \langle u_1, \ldots, u_p \rangle$; on notera $|v|$ la longueur du mot v. $|u|_a$ sera le nombre d'occurrences de a dans u et on notera u_i par $(\vec{\mathbf{u}}')_i$.

– Pour $\vec{\mathbf{u}}$ on notera $\vec{\mathbf{u}}[\vec{\mathbf{X}}, \vec{\mathbf{Y}}]$ le p-vecteur de mots $\langle v_1, \ldots, v_p \rangle$ tel que $v_i = (\vec{\mathbf{u}})_i[(\vec{\mathbf{X}})_i, (\vec{\mathbf{Y}})_i]$.

– $A^{\vec{\mathbf{N}}}$ est l'ensemble des p-vecteurs $\langle u_1, \ldots, u_p \rangle$ de mots sur A tels pour tout i $|u_i| = (\vec{\mathbf{N}})_i$.

– On notera . le produit scalaire :

$$\vec{\mathbf{u}}.\vec{\mathbf{v}} = \sum_{1 \leqq i \leqq p} (\vec{\mathbf{u}})_i (\vec{\mathbf{v}})_i.$$

2.2. Définitions et premières propriétés

Un problème Π sera une fonction à valeur sur $\{0, 1\}$ dont les entrées sont des vecteurs de mots sur l'alphabet A.

Étant donné une entrée $u_1, u_2, \ldots, u_p$, on décompose chaque u_i en k facteurs, le i-ième facteur étant l'entrée du i-ième processeur. Chaque facteur, est repéré par la position de son début $(\vec{\mathbf{X}}_j)_i$ et par sa longueur $(\vec{\mathbf{L}}_j)_i$.

DÉFINITION 2.2.1 : Une k-décomposition d est une application de l'ensemble des p-vecteurs d'entier qui à tout $\vec{\mathbf{N}}$ associe k p-vecteurs d'entiers $((\vec{\mathbf{X}}_i), (\vec{\mathbf{L}}_i))$ pour $1 \leqq i \leqq k$ avec $1 \leqq (\vec{\mathbf{X}}_i)_j \leqq (\vec{\mathbf{X}}_i)_j + (\vec{\mathbf{L}}_i)_j \leqq (\vec{\mathbf{N}})_j$ où $1 \leqq j \leqq p$ et $\vec{\mathbf{X}}_i \leqq \vec{\mathbf{X}}_{i+1}$.

Un k uple de p-vecteurs $((\vec{\mathbf{X}}_i), (\vec{\mathbf{L}}_i))_{1 \leqq i \leqq k}$ tels que

$$1 \leqq (\vec{\mathbf{X}}_i)_j \leqq (\vec{\mathbf{X}}_i)_j + (\vec{\mathbf{L}}_i)_j \leqq (\vec{\mathbf{N}})_j \quad \text{pour } 1 \leqq j \leqq p \qquad \text{et} \qquad \vec{\mathbf{X}}_i \leqq \vec{\mathbf{X}}_{i+1},$$

sera une *k-décomposition de* $\mathbf{N}$.

DÉFINITION 2.2.2 : Soit Π un problème, une $k-a$ décomposition de Π sera une application qui à tout $\vec{\mathbf{N}}$ associera un triplet :

$$\langle d(\vec{\mathbf{N}}) = (\vec{\mathbf{X}}_i, \vec{\mathbf{L}}_i)_{1 \leqq i \leqq k}, (B_i)_{1 \leqq i \leqq k}, \Phi \rangle$$

où :

(1) $d(\vec{\mathbf{N}}) = ((\vec{\mathbf{X}}_i), (\vec{\mathbf{L}}_i))_{1 \leqq i \leqq k}$ est une k-décomposition de $\vec{\mathbf{N}}$;

(2) les B_i sont des fonctions : $A^{\vec{\mathbf{L}}_i} \to \{0, \ldots, a-1\}$;

(3) Φ est une fonction $\Phi: \{0, \ldots, a-1\}^k \to \{0, 1\}$ telle que pour tout $\vec{\mathbf{u}} \in A^{\vec{N}}$:

$$\Pi(\vec{\mathbf{u}}) = 1 \quad \Leftrightarrow \quad \Phi((B_1(\vec{\mathbf{u}}[\vec{\mathbf{X}}_1, \vec{\mathbf{X}}_1 + \vec{\mathbf{L}}_1 - 1]), \ldots, B_k(\vec{\mathbf{u}}[\vec{\mathbf{X}}_k, \vec{\mathbf{X}}_k + \vec{\mathbf{L}}_k - 1])) = 1.$$

Il est clair que d qui à $\vec{\mathbf{N}}$ associe $d(\vec{\mathbf{N}})$ est une k-décomposition.

Si on suppose que l'on dispose de k processeurs indépendants (*i.e.* qui ne communiquent pas entre eux) les données du problème Π sont un vecteur de mots $\vec{\mathbf{u}}$, chaque processeur i prendra en entrée le vecteur de mots $\vec{\mathbf{u}}[\vec{\mathbf{X}}_i, \vec{\mathbf{X}}_i + \vec{\mathbf{L}}_i - 1]$, calculera la fonction B_i, le résultat final étant collecté par la fonction Φ.

Exemple 1: Soit Π_w le problème de tester si étant donné $w \in X^* w$ est un facteur d'un mot u.

Soit la $k-2$ décomposition $\langle d(N) = (X_i, L_i)_{1 \leqq i \leqq k}, (B_i)_{1 \leqq i \leqq k}, \Phi \rangle$:

(*a*) pour $1 \leqq i \leqq k-1$:

$$X_i = (i-1)\left[\frac{N}{k}\right] + 1 \qquad \text{et} \qquad L_i = \left[\frac{N}{k}\right] + |w| - 1$$

$$X_k = (k-1)\left[\frac{N}{k}\right] + 1 \qquad \text{et} \qquad L_k = N - (k-1)\left[\frac{N}{k}\right].$$

(*b*) $B_i(u) = 1 \Leftrightarrow w$ est un facteur de u.

(*c*) $\Phi(x_1, \ldots, x_k) = 1 \Leftrightarrow$ il existe $i\, x_i = 1$.

On peut tout de suite remarquer que :

Propriété 2.2.1 : Une $k-a$ décomposition de Π est aussi une $k-a'$ décomposition de Π pour $a' > a$.

Propriété 2.2.2 : Tout problème Π admet une $k-a$ décomposition pour tout k et pour tout a.

En effet :

- soit $d(\vec{\mathbf{N}})$ définie par :

$$\vec{\mathbf{X}}_1 = \vec{1}, \qquad \vec{\mathbf{L}}_1 = \vec{\mathbf{N}}$$

les $(\vec{\mathbf{X}}_i, \vec{\mathbf{L}}_i)$ pour $i \neq 1$ étant quelconques.

- soit B_1 définie par $B_1(\vec{\mathbf{u}}) = \Pi(\vec{\mathbf{u}})$ les autres B_i étant quelconques;
- soit Φ définie par $\Phi(x_1, \ldots, x_k) = 1 \Leftrightarrow x_1 = 1$.

On a bien ainsi une $k-a$ décomposition pour Π.

La décomposition donnée ci-dessus consiste à ce qu'un seul processeur (le premier) fasse tout le travail, les autres ne faisant rien (ou n'importe quoi). Il est clair que du point de vue des performances, une telle décomposition n'apporte aucun gain par rapport au traitement purement séquentiel du problème et occupe inutilement $k-1$ processeurs. Pour pouvoir comparer les $k-a$ décompositions, on va introduire une notion de temps : on évalue le temps de calcul d'un processeur par la taille de ses entrées, en supposant de plus que le temps du regroupement des calculs par la fonction Φ soit négligeable (ce qui se justifie car la taille de l'entrée de Φ est indépendante de la taille des entrées du problème), le temps total de la $k-a$ décomposition serait égal à la longueur de l'entrée la plus grande. C'est pourquoi, on pose la définition :

DÉFINITION 2.2.3 : Soit $\langle d(\vec{\mathbf{N}}) = (\vec{\mathbf{X}}_i, \vec{\mathbf{L}}_i)_{1 \leqq i \leqq k} \rangle$ une k-décomposition, le *temps* de d pour $\vec{\mathbf{N}}$ sera $\theta(d(\vec{\mathbf{N}})) = \max_{1 \leqq i \leqq k} (\vec{\mathbf{L}}_i . \vec{\mathbf{1}})$.

Le « gain » obtenu par la parallélisation correspondant à $d(\vec{\mathbf{N}})$ sera égal au rapport entre le temps de $d(\vec{\mathbf{N}})$ et le temps correspondant à l'utilisation d'un seul processeur, c'est-à-dire la longueur des entrées du problème. La parallélisation est clairement optimale si ce rapport est égal au nombre de processeurs utilisés. C'est pourquoi on pose la définition :

DÉFINITION 2.2.4 : Soit $\langle d(\vec{\mathbf{N}}) = (\vec{\mathbf{X}}_i, \vec{\mathbf{L}}_i)_{1 \leqq i \leqq k} \rangle$, la *performance* de la décomposition d pour $\vec{\mathbf{N}}$ sera $\lambda(d(\vec{\mathbf{N}})) = k\,\theta(d(\vec{\mathbf{N}}))$.

Par extension si $\langle d(\vec{\mathbf{N}}) = (\vec{\mathbf{X}}_i, \vec{\mathbf{L}}_i)_{1 \leqq i \leqq k}, (B_i)_{1 \leqq i \leqq k}, \Phi \rangle$ est une $k-a$ décomposition, le *temps* (resp. la *performance*) de la $k-a$ décomposition pour $\vec{\mathbf{N}}$ sera le temps (resp. la performance) de $d(\vec{\mathbf{N}})$.

On peut remarquer tout de suite que :

$$\theta(d(\vec{\mathbf{N}})) \leqq \vec{\mathbf{N}} . \vec{\mathbf{1}}$$

et donc aussi :

$$\lambda(d(\vec{\mathbf{N}})) \leqq k(\vec{\mathbf{N}} . \vec{\mathbf{1}})$$

La $k-a$ décomposition donnée dans la propriété 2.2.2 vérifiait :

$$\theta(d(\vec{\mathbf{N}})) = \vec{\mathbf{N}} . \vec{\mathbf{1}} \qquad \text{et} \qquad \lambda(d(\vec{\mathbf{N}})) = k\vec{\mathbf{N}} . \vec{\mathbf{1}}.$$

Dans l'exemple 1, on a :

$$\theta(d(N)) = \left\lceil \frac{N}{k} \right\rceil + |w| - 1$$

$$\lambda(d(N)) = k\,\theta(d(N)) \leqq N + k|w|.$$

Si le problème Π n'est pas purement local, c'est-à-dire si la réponse dépend de toutes les entrées de Π, on peut avoir au mieux avec une $k-a$ décomposition un temps $\theta(d(\vec{\mathbf{N}}))$ de $\vec{\mathbf{N}}.\vec{\mathbf{1}}/k$ et donc une performance $\lambda(d(\vec{\mathbf{N}}))$ de $\vec{\mathbf{N}}.\vec{\mathbf{1}}$.

DÉFINITION 2.2.5 : Π dépend de toutes ses entrées si et seulement si pour toutes les k-décompositions $d(\vec{\mathbf{N}})$ de Π, tous les $\vec{\mathbf{u}}$, tout i, j tels que $1 \leqq j \leqq |(\vec{\mathbf{u}})_i|$ il existe m tel que $(\vec{\mathbf{X}}_m)_i \leqq j < (\vec{\mathbf{X}}_m)_i + (\vec{\mathbf{L}}_m)_i$.

On supposera dans la suite que Π dépend de toutes ses entrées; dans ce cas on aura :

PROPOSITION 2.2.3 : *Pour toute k décomposition* $d(\vec{\mathbf{N}})$, *on a :*

$$\left\lceil \frac{\vec{\mathbf{N}}.\vec{\mathbf{1}}}{k} \right\rceil \leqq \theta(d(\vec{\mathbf{N}})) \leqq \vec{\mathbf{N}}.\vec{\mathbf{1}}$$

et

$$k\left\lceil \frac{\vec{\mathbf{N}}.\vec{\mathbf{1}}}{k} \right\rceil \leqq \lambda(d(\vec{\mathbf{N}})) \leqq k\,\vec{\mathbf{N}}.\vec{\mathbf{1}}.$$

DÉFINITION 2.2.6

– une k-décomposition d est *optimale* si et seulement si il existe une constante C telle que pour tout $\vec{\mathbf{N}}$ on ait : $\lambda(d(\vec{\mathbf{N}})) \leqq \vec{\mathbf{N}}.\vec{\mathbf{1}} + C$;

– un problème Π sera dit $k-a$ *parallélisable* si et seulement si il existe une $k-a$ décomposition $\langle d(\vec{\mathbf{N}}), (B_i)_{1 \leqq i \leqq k}, \Phi \rangle$ de Π où d est optimale;

– un problème Π sera dit k *parallélisable* si et seulement si il existe un a tel que Π soit $k-a$ parallélisable.

Ainsi, dans l'exemple 1, $d(N)$ est optimale, le problème de la reconnaissance d'un mot u donné dans un texte est donc $k-2$ parallélisable.

Remarque 1 : Un problème Π est toujours $1-a$ parallélisable.

Remarque 2 : Si Π est $k-a$ parallélisable alors Π est $k-a'$ parallélisable pour $a' \geqq a$.

On a aussi la proposition suivante :

PROPOSITION 2.2.4 : *Si* Π *est* $k-a$ *parallélisable et si* k' *divise* k *alors* Π *est* $k'-a^{k/k'}$ *parallélisable.*

Preuve : Soit $\langle d(\mathbf{N})=(\mathbf{X}_i, \mathbf{L}_i)_{1 \leqq i \leqq k}, (B_i)_{1 \leqq i \leqq k}, \Phi \rangle$ une $k-a$ décomposition pour Π.

Φ est une fonction de $\{0, \ldots, a-1\}^k \to \{0, 1\}$, l'entrée de Φ étant un mot de longueur k sur $\{0, \ldots, a-1\}$ il peut être codé par une fonction G en un mot de longueur $\lceil k \operatorname{Log}(a)/\operatorname{Log}(r) \rceil$ sur l'alphabet $\{0, \ldots, r-1\}$.

On suppose que $r \geqq a$, et soit $m=\lfloor \operatorname{Log}(r)/\operatorname{Log}(a) \rfloor$.

On peut définir une autre décomposition pour Π.

$d'(\vec{\mathbf{N}})$ est une $\lceil k/m \rceil$ décomposition obtenue en regroupant m éléments de $d(\vec{\mathbf{N}})$:

(1) $(\vec{\mathbf{X}}'_i, \vec{\mathbf{L}}'_i)$ pour $1 \leqq i \leqq \lceil k/m \rceil$ avec :

- $\vec{\mathbf{X}}'_i = \vec{\mathbf{X}}_{(i-1)m+1}$;
- $\vec{\mathbf{L}}'_i = \sum_{(i-1)m+1 \leqq j < i.m} \vec{\mathbf{L}}_j$.

(2) $B'_i(\vec{\mathbf{u}}) = G(B_{(i-1)m+1}(\vec{\mathbf{u}}_1), \ldots, B_{im}(\vec{\mathbf{u}}_m))$ avec $\vec{\mathbf{u}} \in A^{\vec{\mathbf{L}}_i}$ et $\vec{\mathbf{u}}_j \in A^{\vec{\mathbf{L}}_{(i-1)m+j}}$.

(3) $\Phi'(x_1, \ldots, x_{\lfloor k/m \rfloor}) = \Phi(G^{-1}(x_1, \ldots, x_{\lfloor k/m \rfloor}))$.

On a ainsi obtenu une $\lceil k/\lfloor \operatorname{Log}(r)/\operatorname{Log}(a) \rfloor \rceil$-$r$ décomposition pour Π.

On a

$$\theta(d'(\vec{\mathbf{N}})) = \left\lfloor \frac{\operatorname{Log}(r)}{\operatorname{Log}(a)} \right\rfloor \theta(d(\vec{\mathbf{N}}))$$

et

$$\lambda(d'(\vec{\mathbf{N}})) = \left\lceil \frac{k}{\lfloor \operatorname{Log}(r)/\operatorname{Log}(a) \rfloor} \right\rceil \left\lfloor \frac{\operatorname{Log}(r)}{\operatorname{Log}(a)} \right\rfloor \theta(d(\vec{\mathbf{N}})).$$

Si $d(\vec{\mathbf{N}})$ est optimale et si $\operatorname{Log}(r)/\operatorname{Log}(a)$ est un diviseur de k alors $d'(\vec{\mathbf{N}})$ est aussi optimale.

Une autre notion importante pour évaluer les $k-a$ décompositions est la notion de *recouvrement*. Le recouvrement d'une $k-a$ décomposition sera le nombre d'occurences des entrées de Π qui servent d'entrée à *plusieurs* processeurs. En effet une k-décomposition peut permettre à plusieurs processeurs de se partager des entrées, *a priori* une « bonne » k-décomposition devra avoir un recouvrement petit.

DÉFINITION 2.2.7 : Soit $d(\vec{N}) = (\vec{X}_i, \vec{L}_i)_{1 \leqq i \leqq k}$ une k-décomposition :

– Le *recouvrement* du i-ième élément de la k-décomposition pour l'entrée n sera par définition :

$$R_n(d(\vec{N}), i) = \text{Card}\{j/(\vec{X}_i)_n \leqq j < (\vec{X}_i)_n + (\vec{L}_i)_n \text{ et il existe } m \neq i (X_m)_n \leqq j < (X_m)_n + (L_m)_n\}.$$

– Le recouvrement du i-ième élément de la k-décomposition sera

$$R(d(\vec{N}), i) = \sum_{n=1}^{n=p} R_n(d(\vec{N}), i)$$

$R(d(N), i)$ est le nombre de lettres des entrées du i-ième élément de la k-décomposition qui sont aussi des entrées d'autres éléments de la k-décomposition.

– Le recouvrement de $d(\vec{N})$ sera : $R(d(\vec{N})) = \underset{1 \leqq i \leqq k}{\text{Max}} (R(d(\vec{N}), i))$.

Remarques et exemples : (1) La k-décomposition de la propriété 2.2.1 montre que tout problème Π admet une $k-a$ décomposition avec un recouvrement nul.

(2) On peut considérer la *k décomposition standard :*

c'est la k-décomposition $d(\vec{N}) = (\vec{X}_i, \vec{L}_i)_{1 \leqq i \leqq k}$ donnée par :

$$(\vec{X}_i)_j = (i-1)\left\lceil \frac{(\vec{N})_j}{k} \right\rceil \quad \text{et} \quad (\vec{L}_i)_j = \left\lceil \frac{(\vec{N})_j}{k} \right\rceil.$$

Cette k-décomposition est optimale puisque :

$$\theta(d(\vec{N})) = \sum_{n=1}^{n=p} \left\lceil \frac{(\vec{N})_n}{k} \right\rceil$$

et

$$\lambda(d(\vec{N})) = k\,\theta(d(\vec{N})) \leqq \vec{N}.\vec{1} + k.$$

On a aussi :

$$R(d(\vec{N})) = 0.$$

(3) Dans l'exemple 1 on a :

$$R(d(N)) = 2|w|.$$

Ce qui montre que $d(\vec{\mathbf{N}})$ est une k-décomposition optimale de Π_w, qui a un recouvrement non nul.

Si la k-décomposition est optimale, alors nécessairement le recouvrement est borné :

PROPOSITION 2.2.5 : *Si d est une k-décomposition optimale alors il existe C tel que pour tout $\vec{\mathbf{N}}$ on ait $R(d(\vec{\mathbf{N}})) \leqq C$.*

Preuve : On écrira dans cette preuve d pour $d(\vec{\mathbf{N}})$.

$$R(d) = \max_{1 \leqq i \leqq k} \left(\sum_{l=1}^{p} R_l(d, i) \right)$$

d'où

$$R(d) \leqq \sum_{i=1}^{k} \sum_{l=1}^{p} R_l(d, i)$$

$$R(d) \leqq \sum_{l=1}^{p} \sum_{i=1}^{k} R_l(d, i)$$

or

$$\sum_{i=1}^{k} R_l(d, i) = \sum_{i=1}^{k} (\vec{\mathbf{L}}_i)_l - (\vec{\mathbf{N}})_l$$

d'où

$$R(d) \leqq \sum_{l=1}^{p} \sum_{i=1}^{k} (\vec{\mathbf{L}}_i)_l \sum_{l=1}^{p} (\vec{\mathbf{N}})_l$$

$$R(d) \leqq \sum_{i=1}^{k} \vec{\mathbf{L}}_i . \vec{\mathbf{1}} - \vec{\mathbf{N}} . \vec{\mathbf{1}}$$

$$R(d) \leqq k \max_{1 \leqq i \leqq k} (\vec{\mathbf{L}}_i . \vec{\mathbf{1}}) - \vec{\mathbf{N}} . \vec{\mathbf{1}} = \lambda(d) - \vec{\mathbf{N}} . \vec{\mathbf{1}}$$

si d est optimal alors : $R(d) \leqq C$.

Remarque : Ce qui signifie que si le recouvrement est non borné, alors d'une part dans notre modèle la décomposition n'est pas optimale, et d'autre part, dans tous les autres modèles il faudra échanger une information non bornée entre les processeurs, quelque soit la forme de cette échange.

De plus si on a une $k-a$ décomposition optimale pour un problème Π on peut toujours construire une $k-a'$ décomposition optimale sans recouvrement pour Π :

PROPOSITION 2.2.6 : *Si Π est k-parallélisable, alors Π est k-parallélisable pour une k-décomposition de recouvrement nul.*

Preuve : L'idée principale de la construction est la suivante : après élimination du recouvrement, le processus i codera dans sa sortie :

(1) d'une part la partie de ses entrées que les autres processus utilisaient avant élimination du recouvrement;

(2) d'autre part, le processus i fera le calcul pour toutes les valeurs possibles des entrées dont il ne dispose plus.

Ce codage peut se réaliser avec un alphabet fini parce que le recouvrement est borné.

On va juste exposer l'idée de la preuve en se restreignant au cas où les entrées sont des mots. Partant d'une $k-a$ décomposition $\langle d(\mathrm{N}) = (\mathrm{X}_i, \mathrm{L}_i)_{1 \leqq i \leqq k}, B_{i_1 \leqq i \leqq k}, \Phi \rangle$ on va construire une $k-a'$ décomposition $d'(N) = \langle (X'_i, L'_i)_{1 \leqq i \leqq k}, (B'_i)_{1 \leqq i \leqq k}, \Phi' \rangle$.

On va éliminer de proche en proche les recouvrements et coder les lettres se trouvant dans les recouvrements par les fonctions B'_i.

(1) (X'_i, L'_i) correspondra à l'entrée (X_i, L_i) de laquelle on aura supprimé les occurrences figurant dans des (X_j, Y_j) pour $j < i$. Ainsi, si u_i est l'entrée correspondant à (X'_i, L'_i) on peut décomposer u_i en $u_i^1 u_i^2 u_i^3$ où :

– u_i^1 est formé d'occurrences figurant dans des éléments (X_j, L_j) de la k-décomposition $d(N)$ pour $j < i$;

– u_i^2 est formé d'occurrences ne figurant que dans (X_i, L_i);

– u_i^3 est formé d'occurrences figurant dans des éléments (X_j, L_j) de la k-décomposition $d(\mathrm{N})$ pour $j > i$.

(2) $B'_i(u_i^2 u_i^3)$ retournera un couple (f, w) où :

– f sera une fonction : $A^{R(d(N))} \to \{0, \ldots, a-1\}$ telle qu'on ait

$$f(u_i^1) = B_i(u_i^1 u_i^2 u_i^3).$$

– w est un mot permettant de « coder » les occurrences qui figurent dans des (X_j, L_j) pour $j > i$ [ce qui est possible si la k-décomposition $d(N)$ est optimale grâce à la proposition 2.2.5] et donc on aura $w = u_i^3$.

(3) Φ' sera défini par :

$$\Phi'((f_1, w_1), \ldots, (f_n, w_n)) = \Phi((f_1(\ \ ), \ldots, f_i(h(w_1, \ldots, w_{i-1}), \ldots).$$

Où $h(w_1, \ldots, w_{i-1})$ est une fonction qui « restitue » u_i^1 à partir des $w_1, \ldots, w_{i-1}$.

On a donc :

$$\Phi'((f_1, w_1), \ldots, (f_n, w_n)) = \Phi((f_1(\ ), \ldots, f_i(u_i^1), \ldots)$$
$$= \Phi((B_1(u[X_1, X_1 + L_1 - 1]), \ldots, B_k(u[X_k, X_k + L_k - 1])).$$

Ce qui montre que $d'(N)$ est bien une k-décomposition pour Π. D'autre part on à :

$$\theta(d'(N)) \leqq \theta(d(N))$$
$$\lambda(d'(N)) \leqq \lambda(d(N))$$
$$R(d'(N)) = 0.$$

Ainsi si d est optimale il en est de même pour d'.

On va maintenant étudier d'autres propriétés des k-décompositions optimales :

PROPOSITION 2.2.7 : *Une décomposition* d: $\langle d(\vec{N}) = (\vec{X}_i, \vec{L}_i)_{1 \leqq i \leqq k} \rangle$ *est optimale si et seulement si il existe une constante* C *telle que :*

pour tout $\vec{N}$, *pour tout* i $1 \leqq i \leqq k$ *on a :*

$$\left\lfloor \frac{\vec{N}.\vec{1}}{k} \right\rfloor - C \leqq \vec{L}_i.\vec{1} \leqq \left\lfloor \frac{\vec{N}.\vec{1}}{k} \right\rfloor + C.$$

Preuve : Soit une k décomposition d: $\langle d(\vec{N}) = (\vec{X}_i, \vec{L}_i)_{1 \leqq i \leqq k} \rangle$

(1)

$$\left\lfloor \frac{\vec{N}.\vec{1}}{k} \right\rfloor - C \leqq \vec{L}_i.\vec{1} \leqq \left\lfloor \frac{\vec{N}.\vec{1}}{k} \right\rfloor + C$$

entraîne évidemment que d est optimale.

(2) Réciproquement, si d est optimale, il existe C_0 tel que :

$$k \operatorname*{Max}_{1 \leqq i \leqq k} (\vec{L}_i.\vec{1}) \leqq \vec{N}.\vec{1} + C_0.$$

Et donc il existe C' tel que pour tout i :

$$\vec{L}_i.\vec{1} \leqq \left\lfloor \frac{\vec{N}.\vec{1}}{k} \right\rfloor + C'.$$

Si Π utilise toutes ses entrées on a :

$$\sum_{i=1}^{i=k} \vec{L}_i . \vec{I} \geqq \vec{N} . \vec{I}.$$

Et soit un i_0 on a aussi :

$$\vec{L}_{i_0} . \vec{I} \geqq \vec{N} . \vec{I} - \sum_{i=1,\, i \neq i_0}^{i=k} \vec{L}_i . \vec{I}$$

et donc

$$L_{i_0} . \vec{I} \geqq \vec{N} . \vec{I} - (k-1) \left(\left\lfloor \frac{\vec{N} . \vec{I}}{k} \right\rfloor + C' \right) \geqq \left\lfloor \frac{\vec{N} . \vec{I}}{k} \right\rfloor - (k-1) C'.$$

En posant $C = (k-1) C'$ on termine la preuve de la proposition.

2.3. Cas où les entrées sont des mots

Dans le cas où l'entrée de Π est un mot (et non un vecteur de mots), la proposition précédente indique que toutes les k-décompositions optimales sont essentiellement identiques. On a :

PROPOSITION 2.3.1 : *Si d définie par : $d(N) = (X_i, L_i)_{1 \leqq i \leqq k}$ est optimale alors il existe K tel que pour tout N :*

$$(i-1) \left\lfloor \frac{N}{k} \right\rfloor - K \leqq X_i \leqq (i-1) \left\lfloor \frac{N}{k} \right\rfloor + K.$$

Preuve : (1) Par induction sur i on montre que :

$$X_i \leqq (i-1) \left\lfloor \frac{N}{k} \right\rfloor + (i-1) C$$

où C est la constante provenant de la proposition 2.2.7.

Pour $i = 1$ c'est évident.

On a (par induction et en utilisant l'optimalité de d) :

$$X_{i+1} \leqq X_i + L_i \leqq (i-1) \left\lfloor \frac{N}{k} \right\rfloor + (i-1) C + \left\lfloor \frac{N}{k} \right\rfloor + C$$

et donc :

$$X_{i+1} \leqq i\left\lfloor \frac{N}{k} \right\rfloor + i\,C.$$

(2) De même on montre par induction sur i que :

$$X_{k-i} \geqq (k-i-1)\left\lfloor \frac{N}{k} \right\rfloor - (i+1)\,C.$$

Pour $i=0$ on a :

$$X_k + L_k = N$$

et donc :

$$X_k \geqq N - \left\lfloor \frac{N}{k} \right\rfloor - C \geqq k\left\lfloor \frac{N}{k} \right\rfloor - \left\lfloor \frac{N}{k} \right\rfloor - C \geqq (k-1)\left\lfloor \frac{N}{k} \right\rfloor - C.$$

D'autre part :

$$X_{k-(i+1)} \geqq X_{k-i} - L_{k-(i+1)} \geqq (k-i-1)\left\lfloor \frac{N}{k} \right\rfloor - (i+1)\,C - \left\lfloor \frac{N}{k} \right\rfloor - C$$

$$\geqq (k-i-2)\left\lfloor \frac{N}{k} \right\rfloor - ((i+1)+1)\,C.$$

En posant $K=(k+1)\,C$ on termine la preuve de la proposition.

On va maintenant étudier diverses applications des propriétés précédentes. Supposons que les entrées de Π et Π' sont des mots, si op est un opérateur booléen, on notera $\Pi \,\mathrm{op}\, \Pi'$ le problème tel que $\Pi \,\mathrm{op}\, \Pi' = 1 \Leftrightarrow (\Pi = 1)\,\mathrm{op}\,(\Pi' = 1)$. On a alors :

PROPOSITION 2.3.2 : *Si* Π *et* Π' *sont k-parallélisables alors* $\Pi \,\mathrm{op}\, \Pi'$ *est aussi k-parallélisable.*

Preuve : soit $\langle d(N)=(X_i, L_i)_{1 \leqq i \leqq k}, (B_i)_{1 \leqq i \leqq k}, \Phi \rangle$ une $k-a$ décomposition optimale pour Π et soit $\langle d'(N)=(X'_i, L'_i)_{1 \leqq i \leqq k}, (B'_i)_{1 \leqq i \leqq k}, \Phi' \rangle$ une $k-a'$ décomposition optimale pour Π'. On construit la $k-a''$ décomposition $\langle d''(N)=(X''_i, L''_i)_{1 \leqq i \leqq k}, (B''_i)_{1 \leqq i \leqq k}, \Phi'' \rangle$ définie par :

- $X''_i = \mathrm{Min}(X_i, X'_i)$.
- $L''_i = \mathrm{Max}(X_i + L_i, X'_i + L'_i) - X''_i$.

– $a'' = aa'$ l'alphabet de sortie des B_i'' sera considéré comme un couple $(\alpha, \beta) - B_i''$ sera défini par :

$$B_i''(u) = ((B_i(u[X_i'' - X_i,\ X_i'' - X_i + L_i - 1]), B_i'(u[X_i'' - X_i',\ X_i'' - X_i' + L_i' - 1])$$

– Φ'' sera défini par :

$$\Phi''((x_1, x_1'), \ldots, (x_k, x_k')) = 1 \quad \Leftrightarrow \quad \Phi(x_1, \ldots, x_k) \operatorname{op} \Phi'(x_1', \ldots, x_k') = 1.$$

Grâce aux propositions 2.3.1 et 2.2.7 on vérifie que $\langle d''(n), (B_i'')_{1 \leqq i \leqq k}, \Phi'' \rangle$ est bien une k-décomposition optimale pour $\Pi \operatorname{op} \Pi'$.

Considérons maintenant $L_m = \{u$: il existe n tel que $u = u_1 \ldots u_m$ et pour $1 \leqq i \leqq m$, $|u_i| = n$ et $|u_i|_c = |u_i|_b\}$. Soit Π_m le test de $u \in L_m$. On aura :

PROPOSITION 2.3.3 :

(1) *Π_m est m-2-parallélisable;*

(2) *si k' divise k alors Π_k est k'-parallélisable;*

(3) *si k' ne divise pas k alors Π_k n'est pas k'-parallélisable.*

Preuve :

(1) Il suffit de considérer la k-décomposition standard $d(N)$ avec :

– si N est un multiple de k $B_i(u) = 1 \Leftrightarrow |u|_c = |u|_b$ et $\Phi(x_1, \ldots, x_k) = 1 \Leftrightarrow$ pour tout i $1 \leqq i \leqq k$ $x_i = 1$;

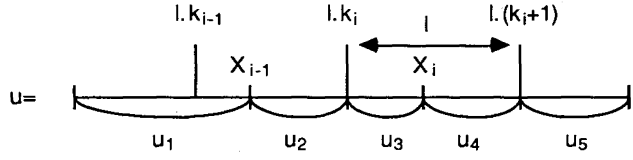
– si N n'est pas un multiple de k alors les B_i seront quelconques et $\Phi(x_1, \ldots, x_k) = 0$ pour tous les x_i.

(2) C'est la proposition 2.2.4.

(3) On va d'abord prouver le lemme :

LEMME : *Soit $\langle d(N) = (X_i, L_i)_{1 \leqq i \leqq k}, B_{i_{1 \leqq i \leqq k}}, \Phi \rangle$ une $k-a$ décomposition optimale sans recouvrement pour Π_m. Pour $d(\mathrm{lm})$ on aura : pour tout i si r_i est le reste de la division euclidienne de X_{i-1} par l, alors* $a \geqq \underset{1 \leqq i \leqq k}{\operatorname{Min}} (\operatorname{Min}(r_i, l - r_1))$.

Preuve du lemme : Soit $\langle d(N)=(X_i, L_i)_{1 \leqq i \leqq k}, (B_i)_{1 \leqq i \leqq k}, \Phi \rangle$ est une $k-a$ décomposition optimale sans recouvrement pour Π_m :



Soit u tel que $|u|=\text{lm}$:

Soit $i \leqq k$ supposons que $r_i=\text{Min}(r_i, l-r_i)$. On a $X_i=lk_i+r_i$ avec $0 \leqq r_i \leqq l/2$.

Soit $u \in L_m$ on peut décomposer u en $u_1 u_2 u_3 u_4 u_5$:

$$u_2=u[X_{i-1}, lk_i]$$
$$u_3=u[lk_i+1, X_i-1]$$
$$u_4=u[X_i, l(k_i+1)].$$

Ainsi le i-1-ième élément de la décomposition est $u_2 u_3$.

Soit $v_j=c^j b^{r_i-j}$ et $t_j=c^{r_i-j} b^j t'$ pour $0 \leqq j \leqq r_i$ où $|t'|_c=|t'|_b$. Si $w_{j_1 j_2}=u_1 u_2 v_{j_1} t_{j_2} u_5$, on a :

$$w_{j_1 j_2} \in L_m \Leftrightarrow j_1=j_2.$$

Nécessairement on aura $B_{i-1}(u_2 v_{j_1}) \neq B_{i-1}(u_2 v_{j_2})$ si $j_1 \neq j_2$, car sinon les entrées de Φ seraient identiques pour $w_{j_1 j_1}$ et $w_{j_2 j_1}$ ce qui est absurde car $w_{j_1 j_1}$ appartient à L_m mais $w_{j_2 j_1}$ n'appartient pas à L_m. Ce qui montre que B_i prendra au moins r_i+1 valeurs.

Revenons à la preuve de la proposition :

Supposons que Π_k est k'-parallélisable, soit $k=q_0 k'+r_0$ avec $0 \leqq r_0 < k'$ d'après la proposition 2.3.1 on aura pour une entrée de longueur kl

$$(i-1)\left\lfloor \frac{l(qk'+r_0)}{k'} \right\rfloor - C \leqq X_i \leqq (i-1)\left\lfloor \frac{l(qk'+r_0)}{k'} \right\rfloor + C$$

et pour l suffisamment grand on aura :

$$(i-1)q_0 l+(i-1)\left\lfloor \frac{r_0 l}{k'} \right\rfloor - C \leqq X_i \leqq (i-1)q_0 l+(i-1)\left\lfloor \frac{r_0 l}{k'} \right\rfloor + C.$$

Ainsi si $r_0 \neq 0$ le reste de la division euclidienne de X_i par l ne pourra être borné, et il n'existera pas de a pour lequel la $k-a$ décomposition soit optimale.

Soit $L \subseteq A^*$ un langage, on dira que L est k-parallélisable si le test de $w \in L$ est k-parallélisable. L'exemple précédent montre qu'il existe des langages algébriques qui ne sont pas k-parallélisables (puisque L_1 est algébrique et n'est pas k-parallélisable pour $k \neq 1$). Cependant, on peut aisément vérifier que $L'_1 = c^n b^n$, $L'_2 = c^n b^n c^n b^n$, ... sont k-parallélisables pour tout k. Pour les langages rationnels, la situation est différente :

PROPOSITION 2.3.4 : *Si $L \subseteq A^*$ est rationnel, alors pour tout k le test de $w \in L$ est k-parallélisable.*

Preuve : Soit $T = \langle A, Q, q_0, F, \delta \rangle$ un automate fini où X est l'alphabet d'entrée, Q est un ensemble fini d'états, q_0 est l'état initial, F l'ensemble des états terminaux, δ la fonction de transition. On suppose que le langage reconnu par T est L. On considère la $k-a$ décomposition suivante :

(1) $d(N) = (X_i, L_i)_{1 \leq i \leq k}$ est la k-décomposition standard.

(2) l'alphabet de sortie des B_i sera considéré comme une application de Q dans Q. $B_i(u)$ sera l'application de Q dans Q définie par $(B_i(u))(q) = \delta(q, u)$.

(3) $\Phi(f_1, \ldots, f_k) = 1 \Leftrightarrow f_k(\ldots(f_1(q_0)\ldots) \in F$.

On a bien, en notant u_i pour $u[X_i, L_i - 1]$:

$$\begin{aligned} \Phi(B_1(u_1), \ldots, B_k(u_k)) = 1 &\Leftrightarrow \delta(\ldots(\delta(q_0, u_1), u_2, \ldots, u_k)\ldots) \in F \\ &\Leftrightarrow \delta(q_0, u_1 \ldots u_k) \in F \\ &\Leftrightarrow u \in L. \end{aligned}$$

Si l'on note A_k la classe des problèmes sur les mots qui sont k-parallélisables, on a :

PROPOSITION 2.3.5 :

(1) *A_k est fermée par les opérations booléennes.*

(2) *$A_{k.k'} \subset A_k \subset A_1$ (les inclusions étant strictes) pour tout kk' tels que $k.k' \neq 1$.*

Preuve : C'est un simple corollaire de la proposition 2.3.2 et de la proposition 2.2.4.

2.4. Additions de n nombres en base b

Si $u \in \{0, \ldots, b-1\}^*$ on notera $\bar{u}$ l'entier représenté par u dans la base b. Par convention on pose : $\bar{\varepsilon}=0$.

Soit $u^1, u^2, \ldots, u^n, v$ $n+1$ mots de $\{0, \ldots, b-1\}^*$, le problème est de tester si $\bar{u}^1 + \ldots + \bar{u}^n = \bar{v}$.

On va considérer deux $k-a$ décompositions pour le problème de l'addition : la première sera une $k-2$ décomposition optimale avec recouvrement et la deuxième sera une $k-n^2+1$ décomposition optimale sans recouvrement.

On va exposer l'idée de la première k-décomposition en se restreignant au cas de l'addition de deux nombres.

Exemple :

pour l'addition :

$$\begin{array}{r} 101293 \\ +\ 88916 \\ \hline 190209 \end{array}$$

Entrée des processus

Processus 1		Processus 2		Processus 3
10		012		293
8		889		916
19		902		209

Le processus 2 rendra comme résultat 1 : il ne s'occupe pas de la validité de $2+9=2$ (ce qui sera assuré par le processus 3), mais déduit de $2+9=2$ qu'il y a une retenue, et vérifie que les autres résultats sont cohérents ($01+88+1=90$). De même, les autres processus ont la valeur 1.

Chaque processus i aura comme entrées les mots $u_i^1\, x_i^1$, $u_i^2\, x_i^2$, vy. Où x_i^1 et x_i^2 et y sont des lettres de $\{0, \ldots, b-1\}$. Le processus i testera si $\overline{u_i^1} + \overline{u_i^2} + r = \overline{sv}$ avec :

- s égal à 1 ou 0 (retenue propagée ou pas de retenue);
- $r=0$ (absence de retenue à droite) si $\bar{y} = \overline{x_i^1} + \overline{x_i^2}$ ou $\bar{y} = \overline{x_i^1} + \overline{x_i^2} + 1$;
- $r=1$ (cas d'une retenue supposée à droite) si $\overline{1y} = \overline{x_i^1} + \overline{x_i^2}$ ou $\overline{1y} = \overline{x_i^1} + \overline{x_i^2} + 1$.

Ainsi chaque processus teste si la partie de l'addition qui le concerne est possible sans s'occuper de la validité réelle des chiffres les plus à droite dans la partie qui le concerne. Ces chiffres les plus à droite étant « partagés » avec le processus voisin, c'est ce voisin qui dans son test permettra d'assurer que l'addition est bien valide.

La deuxième méthode, consistera à coder dans le résultat des processus l'information nécessaire pour savoir si l'addition est valide.

Entrée des processus

Processus 1		Processus 2		Processus 3
10		12		93
8		89		16
19		02		09

Le processus 1 retournera (0, 1) pour indiquer que le résultat est vrai si on a une retenue.

Le processus 2 retournera (1, 1) pour indiquer que le résultat est vrai si on a une retenue et qu'il y a de plus une retenue à transmettre à gauche.

Le processus 3 retournera (1, 0) pour indiquer que le résultat est vrai et qu'il y a de plus une retenue à transmettre à gauche.

Chaque processus, donnera comme résultat un couple (r_1, r_2) avec $r_j = 0$ ou 1, si $\overline{u_i^1} + \overline{u_i^2} + \overline{r^2} = \overline{r_i v_i}$ et comme résultat *faux* si un tel couple n'existe pas. En fait chaque processus donne comme résultat d'une part la nécessité d'une retenue à droite et d'autre part le fait qu'il a généré une retenue à gauche.

PROPOSITION 2.4.1 : *Pour tout k et pour tout n, l'addition de n nombres en base b est* $k-2$ *parallélisable avec un recouvrement d'au plus* $2\lceil \mathrm{Log}_b(n) \rceil (n+1)$.

Preuve : Par commodité d'indiçage, on écrira $\vec{\mathbf{X}}_i$, $\vec{\mathbf{L}}_i$ au lieu de $\vec{\mathbf{X}}_{k-i}$, $\vec{\mathbf{L}}_{k-i}$. Soit $\langle d(\vec{\mathbf{N}}) = (\vec{\mathbf{X}}_i, \vec{\mathbf{L}}_i)_{1 \leqq i \leqq k}, B_{i_1 \leqq i \leqq k}, \Phi \rangle$ (où $\mathbf{N}$ est un $n+1$ vecteur) la $k-2$ décomposition suivante :

(1) Les k couples de $n+1$ vecteurs $(\vec{\mathbf{X}}_i, \vec{\mathbf{L}}_i)_{1 \leqq i \leqq k}$ sont définis pour $1 \leqq j \leqq n+1$ par :

$$(\vec{\mathbf{X}}_i)_j = (\vec{\mathbf{N}})_j - (i-1)m$$

$$(\vec{\mathbf{L}}_i)_j = m + \lceil \mathrm{Log}_b(n) \rceil$$

avec

$$m = \left\lceil \frac{\vec{\mathbf{N}}.\vec{\mathbf{1}}}{k(n+1)} \right\rceil.$$

(2) Avant de définir les B_i remarquons que l'unicité de la division euclidienne montre que pour tout x entier relatif, pour tout i entier naturel, il existe un unique couple (α, β) d'entiers naturels tel que

$$x = \beta b^i - \alpha \qquad \text{avec} \quad 0 \leqq \alpha < b^i.$$

Les B_i pour $1 \leqq i \leqq k$ seront définis par :

Cas $i = 1$:

$$B_1(v^1 x^1, \ldots, v^n x^n, wz) = 1 \qquad \text{avec} \quad |z| = |x^i| = \lceil \mathrm{Log}_b(n) \rceil$$

si et seulement si $\overline{v^1} + \ldots + \overline{v^n} + \beta = \overline{w}$ où β est solution de :

$$\overline{x^1} + \ldots + \overline{x^n} - \bar{z} = \beta\, b^{\lceil \mathrm{Log}_b(n) \rceil} - \alpha \qquad \text{avec} \quad 0 \leqq \alpha < b^{\lceil \mathrm{Log}_b(n) \rceil}.$$

Cas général :

$$B_i(v^1 x^1, \ldots, v^n x^n, wz) = 1 \qquad \text{avec} \quad |z| = |x^i| = \lceil \mathrm{Log}_b(n) \rceil$$

si et seulement si il existe γ tel que $\bar{\gamma} < n$ tel que :

$$\overline{v^1} + \ldots + \overline{v^n} + \beta = \overline{\gamma\, w}$$

où β est solution de :

$$\overline{x^1} + \ldots + \overline{x^n} - \bar{z} = \beta\, b^{\lceil \mathrm{Log}_b(n) \rceil} - \alpha \qquad \text{avec} \quad 0 \leqq \alpha < b^{\lceil \mathrm{Log}_b(n) \rceil}$$

Cas $i = k$:

$$B_i(v^1, \ldots, v^n, w) = 1$$

si et seulement si il existe γ tel que $\bar{\gamma} < n$ tel que :

$$\overline{v^1} + \ldots + \overline{v^n} = \overline{\gamma\, w}.$$

(3) $\varphi(x_1, \ldots, x_k) = 1 \Leftrightarrow x_i = 1$ pour tout i $1 \leqq i \leqq k$.

Notons $\vec{\mathbf{u}}_i$ le $n+1$ vecteur de mots $\vec{\mathbf{u}}[\vec{\mathbf{X}}_i, \vec{\mathbf{L}}_i - 1]$ qui sert d'entrée au B_i. On va montrer que la k-décomposition donnée est correcte c'est-à-dire que :

$$\varphi(B_1(\vec{\mathbf{u}}_1), \ldots, B_k(\vec{\mathbf{u}}_k)) = 1 \quad \Leftrightarrow \quad \overline{(\vec{\mathbf{u}})_1} + \ldots + \overline{(\vec{\mathbf{u}})_n} = \overline{(\vec{\mathbf{u}})_{n+1}}.$$

Pour cela, on montre par induction sur i que :

Pour tout $i < k$:

$$(B_{k-i}(\vec{\mathbf{u}}_{k-i}) = 1 \text{ et } \ldots \text{ et } B_k(\vec{\mathbf{u}}_k) = 1)$$

$$\Leftrightarrow \quad \text{il existe } \alpha\, 0 \leqq \alpha < n \sum_{j=1}^{j=n} \overline{(\vec{\mathbf{u}}_{k-i} \ldots \vec{\mathbf{u}}_k)_j} = \overline{(\alpha\, \vec{\mathbf{u}}_{k-i} \ldots \vec{\mathbf{u}}_k)_{n+1}}$$

pour $i = 0$:

C'est la définition même de B_k

pour $i = l + 1$:

Soit

$$\vec{\mathbf{u}}_{k-l-1} = \langle v^1 x^1, \ldots, v^n x^n, wz \rangle \qquad \text{avec} \quad |z| = |x^i| = \lceil \mathrm{Log}_b(n) \rceil,$$

on a

$$\vec{\mathbf{u}}_{k-l} \ldots \vec{\mathbf{u}}_k = \langle x^1 s^1, \ldots, x^n s^n, zt \rangle \text{ pour certains } s^i \text{ et } t.$$

D'après la définition de B_i on aura :

$B_{k-(l+1)}(\vec{\mathbf{u}}_{k-(l+1)}) = 1$ et $\ldots$ et $B_k(\vec{\mathbf{u}}_k) = 1$

$\Leftrightarrow$ il existe δ $0 \leqq \delta < n$ $\overline{x^1 s^1} + \ldots + \overline{x^n s^n} = \overline{\delta z t}$ (hypothèse d'induction) (A)

et il existe γ $0 \leqq \gamma < n$ tel que :

$$\overline{v^1} + \ldots + \overline{v^n} + \beta = \overline{\gamma w}$$

où β est solution de :

$$\overline{x^1} + \ldots + \overline{x^n} - \overline{z} = \beta\, b^{\lceil \mathrm{Log}_b(n) \rceil} - \alpha \qquad \text{avec} \quad 0 \leqq \alpha < b^{\lceil \mathrm{Log}_b(n) \rceil}.$$

De (A) on déduit qu'il existe un $r < n$ (correspondant à la « retenue ») tel que

$$\overline{s^1} + \ldots + \overline{s^n} = \overline{t} + r b^{|t|} \qquad \text{et} \qquad \overline{x^1} + \ldots + \overline{x^n} + r = \overline{\delta z}.$$

Soit encore :

$$\overline{x^1} + \ldots + \overline{x^n} = \overline{\delta} b^{\lceil \mathrm{Log}_b(n) \rceil} + \overline{z} - r \qquad \text{avec} \quad 0 \leqq r < n.$$

Ce qui montre que $\beta = \bar{\delta}$ et $\alpha = r$, et donc :

Il existe $\gamma\, 0 \leqq \bar{\gamma} < n$ tel que :

$$\begin{aligned}\overline{v^1 x^1 s^1} + \ldots + \overline{v^n x^n s^n} &= \overline{v^1 b^{|zt|}} + \ldots + \overline{v^n b^{|zt|}} + \overline{x^1 s^1} + \ldots + \overline{x^n s^n} \\ &= \overline{\gamma\, w b^{|zt|}} - \bar{\delta} b^{|zt|} + \overline{\delta z t} \\ &= \overline{\gamma\, w z t}.\end{aligned}$$

Le recouvrement induit par cette k-décomposition est

$$R(d(\vec{\mathbf{N}})) = 2\lceil \mathrm{Log}_b(n) \rceil (n+1)$$

puisque chaque élément de la décomposition recouvre le précédent d'un facteur gauche de longueur $\lceil \mathrm{Log}_b(n) \rceil$.

D'autre part on a :

$$\theta(d(\vec{\mathbf{N}})) = (n+1)\left(\left\lceil \frac{\vec{\mathbf{N}}.\vec{\mathbf{1}}}{k(n+1)} \right\rceil + \lceil \mathrm{Log}_b(n) \rceil\right) \leqq \left\lceil \frac{\vec{\mathbf{N}}.\vec{\mathbf{1}}}{k} \right\rceil + (n+1)\lceil \mathrm{Log}_b(n) \rceil + 1.$$

Et donc

$$\lambda(d(\vec{\mathbf{N}})) = k\,\theta(d(\vec{\mathbf{N}})) \leqq k\left\lceil \frac{\vec{\mathbf{N}}.\vec{\mathbf{1}}}{k} \right\rceil + k + k(n+1)\lceil \mathrm{Log}_b(n) \rceil \leqq \vec{\mathbf{N}}.\vec{\mathbf{1}} + C.$$

PROPOSITION 2.4.2 : *Pour tout k et pour tout n, l'addition de n nombres en bases b est k-(n^2+1) parallélisable avec un recouvrement nul.*

Preuve : Par commodité d'indiçage, on écrira $\vec{\mathbf{X}}_i$ (resp. $\vec{\mathbf{L}}_i$, $\vec{B}_i$) au lieu de $\mathbf{X}_{k-i}$, ($\mathbf{L}_{k-i}$, B_{k-i}). Soit la $k-2$ décomposition $\langle d(\vec{\mathbf{N}}) = (\vec{\mathbf{X}}_i, \vec{\mathbf{L}}_i)_{1 \leqq i \leqq k}, B_{i_{1 \leqq i \leqq k}}, \Phi \rangle$ (où $\vec{\mathbf{N}}$ est un $n+1$ vecteur) définie par :

(1) Les k couples de $n+1$ vecteurs $(\vec{\mathbf{X}}_i, \vec{\mathbf{L}}_i)_{1 \leqq i \leqq k}$ sont définis pour $1 \leqq j \leqq n+1$ par :

$$\begin{aligned}(\vec{\mathbf{X}}_i)_j &= (i-1)m \\ (\vec{\mathbf{L}}_i)_j &= m\end{aligned}$$

avec

$$m = \left\lceil \frac{\vec{\mathbf{N}}.\vec{\mathbf{1}}}{k(n+1)} \right\rceil.$$

(2) La sortie de B_i sera considérée comme un couple (α, β) d'entiers avec $0 \leqq \alpha < n$ et $0 \leqq \beta < n$ ou *faux*. On aura :

$B_k(\vec{\mathbf{u}}) = (0, \beta)$ où β est l'unique solution inférieure à n (si elle existe) de $\sum_{i=1}^{i=n} \overline{(\vec{\mathbf{u}})_i} + \beta = \overline{(\vec{\mathbf{u}})_{n+1}}$.

$B_k(\vec{\mathbf{u}}) = faux$ si une telle solution n'existe pas.

$B_i(\vec{\mathbf{u}}) = (\alpha, \beta)$ pour $1 < i < k$ où $\alpha\beta$ sont les uniques solutions $0 \leqq \alpha < n$ et $0 \leqq \beta < n$ (si elles existent) de $\sum_{i=1}^{i=n} \overline{(\vec{\mathbf{u}})_i} + \beta = \overline{\alpha(\vec{\mathbf{u}})_{n+1}}$.

$B_i(\vec{\mathbf{u}}) = faux$ si de telles solutions n'existent pas.

$B_1(\vec{\mathbf{u}}) = (\alpha, 0)$ où α est l'unique solution $0 \leqq \alpha < n$ (si elle existe) de $\sum_{i=1}^{i=n} \overline{(\mathbf{u})_i} = \overline{\alpha(\mathbf{u})_{n+1}}$.

$B_1(\bar{\mathbf{u}}) = faux$ si une telle solution n'existe pas.

(3) $\varphi((\alpha_1, \beta_1), \ldots, (\alpha_k, \beta_k)) = 1 \Leftrightarrow \alpha_k = 0$ et $\alpha_i = \beta_{i+1}$ *pour* $0 < i < k$ et $\beta_1 = 0$

et $\varphi(x_1, \ldots, x_k) = 0$ si un des x_i est égal à *faux*.

On vérifie aisément que $d(\vec{\mathbf{N}})$ est bien une k-décomposition pour le problème de l'addition.

Pour la multiplication, la situation est différente : on peut aisément montrer que la multiplication de deux nombres n'est k-parallélisable que pour $k = 1$ (c'est-à-dire que l'optimalité n'est conservée que dans le cas de la séquentialité pure).

3. PARALLÉLISATION DES PROBLÈMES SUR LES MATRICES

Les résultats précédents s'étendent à des algorithmes dont les entrées ne sont pas des mots.

Il est toujours possible de rendre une entrée linéaire, seulement une linéarisation arbitraire peut compliquer un algorithme.

La définition précédente de la $k-a$ décomposition s'étend directement aux problèmes dont les entrées ont une dimension quelconque. On peut par analogie avec la topologie discrète décomposer l'entrée en une union de pavés. Cependant faire ce travail quel que soit la dimension des entrées semble peu intéressant pour les problèmes étudiés par l'informatique.

Nous nous limiterons donc à des entrées à deux dimensions en pensant à tous les problèmes où les données sont des matrices.

Une $k-a$ décomposition consistera à décomposer chaque matrice d'entrée en k sous matrices. Chaque processeur prendra en entrée une sous-matrice et calculera une certaine fonction B_i sur cette entrée, en composant leurs résultats, on obtiendra celui du problème.

3.1. Notations

– $\vec{\mathbf{N}}$ sera un $2-p$-vecteur d'entiers.

– Si A est un alphabet, $\vec{\mathbf{u}}$ sera un p-vecteur de matrices sur A.

– Si $\mathbf{u}$ est une matrice $\mathbf{u}(m, n)$ est la lettre située sur la m-ième ligne et la n-ième colonne de $\mathbf{u}$ si elles existent et 1 sinon; $\mathbf{u}[X, Y, L, M]$ sera la matrice $\mathbf{u}'$ telle que $\mathbf{u}'(i, j) = \mathbf{u}(i+X-1, j+Y-1)$ pour tout $1 \leqq i \leqq L$ et $1 \leqq j \leqq M$.

C'est-à-dire que l'on définit parfaitement une sous matrice en donnant la position dans la matrice initiale de sa première ligne et première colonne ainsi que son nombre de lignes et de colonnes.

– Pour $\vec{\mathbf{u}}$ on notera $\vec{\mathbf{u}}[\vec{\mathbf{X}}, \vec{\mathbf{N}}]$ le p-vecteur de matrices $\langle \mathbf{v}_1, \ldots, \mathbf{v}_p \rangle$ tel que

$$\mathbf{v}_i = (\vec{\mathbf{u}})_i(((\vec{\mathbf{X}})_1)_i, ((\vec{\mathbf{N}})_1)_i, ((\vec{\mathbf{N}})_2)_i)$$

$\vec{\mathbf{X}}$ représente les coordonnées des premiers éléments des sous matrices, et $\mathbf{N}$ les dimensions de celles-ci.

– $A^{\vec{\mathbf{N}}}$ est l'ensemble des p-vecteurs $\langle \mathbf{u}_1, \ldots, \mathbf{u}_p \rangle$ de matrices sur A tels pour tout i les dimensions de $\mathbf{u}_i$ soient $((\vec{\mathbf{N}})_1)_i$, $((\vec{\mathbf{N}})_2)_i$.

3.2. Définitions et propriétés

DÉFINITION 3.2.1 : Soit $\vec{\mathbf{N}}$ un $2-p$ vecteur d'entiers, une k-décomposition, d sera une application qui à $\vec{\mathbf{N}}$ associe k $2-p$ vecteurs d'entiers $(\vec{\mathbf{X}}_i, \vec{\mathbf{L}}_i)$ pour $1 \leqq i \leqq k$ avec pour $j = 1, 2$ $1 \leqq (\vec{\mathbf{X}}_i)_j \leqq (\vec{\mathbf{X}}_i)_j + (\vec{\mathbf{L}}_i)_j \leqq \mathbf{N}_j$.

Comme dans le cas des vecteurs de mots, les k $2-p$ vecteurs d'entiers $(\vec{\mathbf{X}}_i, \vec{\mathbf{L}}_i)_{1 \leqq i \leqq k}$ tels que pour $j = 1, 2$ *on a* $1 \leqq (\vec{\mathbf{X}}_i)_j \leqq (\vec{\mathbf{X}}_i)_j + (\vec{\mathbf{L}}_i)_j \leqq \vec{\mathbf{N}}_j$ seront appelés *k-décomposition* de $\vec{\mathbf{N}}\vec{\mathbf{R}}$.

DÉFINITION 3.2.2 : Soit Π un problème, une $k-a$ décomposition de Π sera une application qui à tout $\vec{\mathbf{N}}$ associera un triplet :

$\langle d(\vec{N}) = (\vec{X}_i, \vec{L}_i)_{1 \leq i \leq k}, B_{i_1 \leq i \leq k}, \Phi \rangle$ où :

(1) $d(\vec{N})$ est une k-décomposition de $\vec{N}$;

(2) les B_i sont des fonctions : $A^{\vec{L}_i} \to \{0, \ldots, a-1\}$;

(3) Φ est une fonction : $\{0, \ldots, a-1\}^k \to \{0, 1\}$ telle que pour tout $\vec{u} \in A^{\vec{N}}$

$$\Pi(\vec{u}) = 1 \quad \Leftrightarrow \quad \Phi(B_1(\vec{u}[\vec{X}_1, \vec{L}_1]), \ldots, B_k(\vec{u}[\vec{X}_k, \vec{L}_k])) = 1$$

(**u** est bien un vecteur de matrices).

On a sur la $k-a$ décomposition les propriétés suivantes de manière triviale.

Propriété 3.2.1 : Tout problème Π admet une $k-a$ décomposition pour tout k et pour tout a.

Propriété 3.2.2 : Une $k-a$ décomposition de Π est aussi une $k-a'$ décomposition de Π pour $a' > a$.

Ici encore se pose le problème de trouver un critère qui permettra de choisir les « bonnes » $k-a$ décompositions.

On évalue le *temps* des B_i par la taille maximale des entrées.

DÉFINITION 3.2.3 : Soit d: $d(\vec{N}) = (\vec{X}_i, \vec{L}_i)_{1 \leq i \leq k}$ le *temps* de d est

$$\theta(d(\vec{N})) = \max_{1 \leq i \leq k} ((\vec{L}_i)_1 . (\vec{L}_i)_2).$$

DÉFINITION 3.2.4 : Soit d: $d(\vec{N}) = (\vec{X}_i, \vec{L}_i)_{1 \leq i \leq k}$ la *performance* de d est

$$\lambda(d(\vec{N})) = k\,\theta(d(\vec{N})).$$

On a les bornes supérieures suivantes :

$$\theta(d(\vec{N})) \leq (\vec{N})_1 . (\vec{N})_2$$
$$\lambda(d(\vec{N})) \leq k(\vec{N})_1 . (\vec{N})_2.$$

Ces bornes supérieures sont atteintes pour la $k-a$ décomposition triviale où le premier élément de la décomposition est toute l'entrée et où les autres éléments sont quelconques.

On supposera dans ce qui suit que toutes les entrées sont utiles dans le sens donné au paragraphe 2, définition 2.2.5.

On a donc les bornes inférieures suivantes :

$$\theta(d(\vec{\mathbf{N}})) \geqq \left\lceil \frac{(\vec{\mathbf{N}})_1 \cdot (\vec{\mathbf{N}})_2}{k} \right\rceil$$

$$\lambda(d(\vec{\mathbf{N}})) \geqq k \left\lceil \frac{(\vec{\mathbf{N}})_1 \cdot (\vec{\mathbf{N}})_2}{k} \right\rceil.$$

Ces bornes inférieures ne sont pas toujours atteintes car les décompositions demandent des sous matrices et non pas des surfaces quelconques.

Exemple : On suppose que l'on a une seule entrée en l'occurrence une matrice de dimension $(2p+1, 2q+1)$ et on cherche une 2-décomposition.

Le *mieux* que l'on puisse faire est de partager la plus grande des entrées en deux, $(2p+1)$, par exemple.

La taille des entrées sera :

- $p(2q+1)$ pour le premier élément de la décomposition;
- $(p+1)(2q+1)$ pour le deuxième et dernier élément de la décomposition.

On aura donc :

$$\theta(d(2p+1, 2q+1)) = (2q+1)(p+1)$$

soit :

$$\theta(d(2p+1, 2q+1)) = p(2q+1) + 2q + 1$$

alors que :

$$\left\lceil \frac{(\vec{\mathbf{N}})_1 \cdot (\vec{\mathbf{N}})_2}{k} \right\rceil = \left\lceil \frac{(2p+1)(2q+1)}{2} \right\rceil$$

$$= p(2q+1) + q + 1.$$

Soit q éléments de plus imposés par la décomposition en sous matrices.

DÉFINITION 3.2.5 : Soit d une k-décomposition : $d(\vec{\mathbf{N}}) = (\vec{\mathbf{X}}_i, \vec{\mathbf{L}}_i)_{1 \leqq i \leqq k}$, on dira que d est *optimale* si il existe une constante C telle que

$$\lambda(d(\vec{\mathbf{N}})) \leqq (\vec{\mathbf{N}})_1 \cdot (\vec{\mathbf{N}})_2 + C \max_{1 \leqq j \leqq p} (\min(((\vec{\mathbf{N}})_1)_j, ((\vec{\mathbf{N}})_2)_j)).$$

On dira que Π est $k-a$ *parallélisable* si il existe une $k-a$ décomposition $\langle d, (B_i)_{1 \leqq i \leqq k}, \Phi \rangle$ pour laquelle d est optimale.

Remarque : On retrouve la définition 2.2.6 à partir de 3.2.5.

– de la k-décomposition avec

$$(\vec{\mathbf{N}})_2 = (1, 1, \ldots, 1) = \mathbf{1}$$

et pour tout i

$$(\vec{\mathbf{X}}_i)_2 = (\vec{\mathbf{L}}_i)_2 = (1, 1, \ldots, 1) = \vec{\mathbf{1}}$$

– de θ car

$$\theta(d((\vec{\mathbf{N}})_1, \vec{\mathbf{1}})) = \max_{1 \leqq i \leqq k} ((\vec{\mathbf{L}}_i)_1 . (\vec{\mathbf{I}}_i))$$

– de l'optimalité car $\max_{1 \leqq j \leqq p} (\min(((\vec{\mathbf{N}})_1)_j, ((\vec{\mathbf{N}})_2)_j)) = 1.$

Dans l'exemple précédent la définition explicitée était bien minimale en effet :

$$\lambda(d(2p+1, 2q+1)) = (2q+1)(2p+1) + 2q + 1$$

soit

$$\lambda(d(\vec{\mathbf{N}})) = (\vec{\mathbf{N}})_1 . (\vec{\mathbf{N}})_2 + ((\vec{\mathbf{N}})_2)_1 .$$

La constante vaut ici 1 [on a supposé que $((\vec{\mathbf{N}})_2)_1 = 2q+1$ était la plus petite des entrées].

Les propriétés suivantes restent vraies.

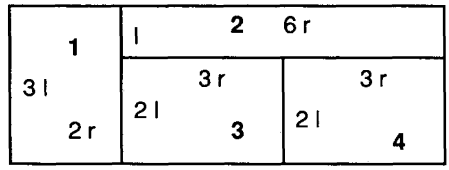
Propriété 3.2.3 : Π est toujours $1-a$ parallélisable pour ($a \geqq 2$).

Propriété 3.2.4 : Si Π est $k-a$ parallélisable alors Π est $k-a'$ parallélisable pour $a' > a$.

Par contre si Π est $k-a$ parallélisable on ne peut rien dire de la $k'-a^{k/k'}$ parallélisation de Π. Car la méthode qui permettait de passer d'une $k-a$ décomposition à une $k'-a^{k/k'}$ décomposition pour k' diviseur de k n'est plus applicable dans ce contexte. En effet la réunion de deux matrices ne donne pas, en général, une matrice. Pour reformer une matrice, on est amené à introduire des recouvrements qui détruisent la minimalité de la décomposition.

Exemple : Soit Π un problème à une entrée. Les dimensions de la matrice d'entrée sont ($3l$, $8r$);

pour $k=4$, on a la décomposition suivante :



La 4-décomposition de la figure est minimal

$$\lambda(d(3l, 8r))=4(3l.2r)=3l.8r.$$

Une 2-décomposition obtenue en réunissant les matrices précédentes (et on ne peut faire autrement sans connaître la nature du problème traité) donne au mieux : (1) pour le premier élément de la décomposition (2, 3, 4) pour le second.

$$\theta(d(3l, 8r))=6r.3l$$
$$\lambda(d(3l, 8r))=2.(3l.6r)=3l.8r+12lr$$

et

$$12lr \neq C\min(3l, 8r).$$

Pour que cette propriété puisse toujours être vérifiée, il faudrait imposer des conditions de régularité sur la décomposition.

Une autre mesure de l'optimalité des décompositions est le nombre d'occurrences des entrées de Π qui servent d'entrée à plusieurs processus.

DÉFINITION 3.2.6 : Pour la l-ième entrée du i-ième élément de la décomposition d, $\rho(d, l, i)$ représente le nombre d'éléments en entrée partagés avec les autres éléments de la décomposition d.

DÉFINITION 3.2.7 :

$$\rho(d, i)=\sum_{l=1}^{p}\rho(d, l, i)$$

$$\rho(d)=\max_{1\leqq i\leqq k}(\rho(d, i))$$

$\rho(d)$ représente le *recouvrement i.e.* le nombre maximal, pour un élément de la décomposition, d'entrées partagées avec les autres.

Propriété 3.2.5 : Si d est optimal alors il existe C telle que pour tout $\vec{\mathbf{N}}$:

$$\rho(d(\vec{\mathbf{N}})) \leqq C \max_{1 \leqq j \leqq p} (\min(((\vec{\mathbf{N}})_1)_j, ((\vec{\mathbf{N}})_2)_j)).$$

Preuve : On écrira dans cette preuve d pour $d(\vec{\mathbf{N}})$.

$$\rho(d) = \max_{1 \leqq i \leqq k} \left(\sum_{l=1}^{p} \rho(d, l, i) \right)$$

d'où

$$\rho(d) \leqq \sum_{i=1}^{k} \sum_{l=1}^{p} \rho(d, l, i)$$

$$\rho(d) \leqq \sum_{l=1}^{p} \sum_{i=1}^{k} \rho(d, l, i)$$

or

$$\sum_{i=1}^{k} \rho(d, l, i) = \sum_{i=1}^{k} ((\vec{\mathbf{L}}_i)_1)_l ((\vec{\mathbf{L}}_i)_2)_l - ((\vec{\mathbf{N}})_1)_l ((\vec{\mathbf{N}})_2)_l$$

d'où

$$\rho(d) \leqq \sum_{l=1}^{p} \sum_{i=1}^{k} ((\vec{\mathbf{L}}_i)_1)_l ((\vec{\mathbf{L}}_i)_2)_l - \sum_{l=1}^{p} ((\vec{\mathbf{N}})_1)_l ((\vec{\mathbf{N}})_2)_l$$

$$\rho(d) \leqq \sum_{i=1}^{k} (\vec{\mathbf{L}}_i)_1 . (\vec{\mathbf{L}}_i)_2 - (\vec{\mathbf{N}})_1 . (\vec{\mathbf{N}})_2$$

$$\rho(d) \leqq k \max_{1 \leqq i \leqq k} (\vec{\mathbf{L}}_i)_1 . (\vec{\mathbf{L}}_i)_2 - (\vec{\mathbf{N}})_1 . (\vec{\mathbf{N}})_2 = \lambda(d) - (\vec{\mathbf{N}})_1 . (\vec{\mathbf{N}})_2$$

si d est optimal alors :

$$\rho(d) \leqq C \max_{1 \leqq j \leqq p} (\min(((\vec{\mathbf{N}})_1)_j, ((\vec{\mathbf{N}})_2)_j)).$$

On ne peut obtenir ici un résultat général sur l'élimination du recouvrement par codage. On montrera dans l'exemple qui suit que pour certaines décompositions, et pour ce problème particulier, on peut passer d'une $k-a$ décomposition avec recouvrement à une $k-a'$ décomposition sans recouvrement avec a' de l'ordre de $a \,.\, \rho(d)$.

3.3 Exemples

3.3.1. *Recherche de figures*

On étudie ici un problème avec une seule entrée.

Un problème classique est la recherche de figures dans une image. Pour simplifier l'écriture, la matrice d'entrée se composera de 0 et de 1.

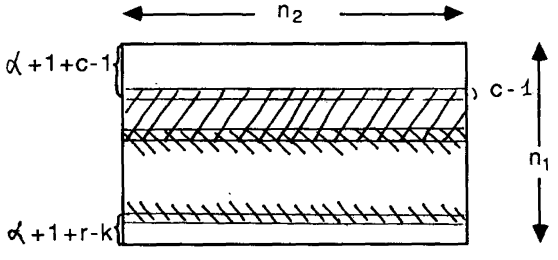
Elle a pour dimension (n_1, n_2).

Π a pour résultat vrai si il y a un carré de 1 de côté c dont les axes sont ceux de la matrice.

Propriété 3.3.1.1. – Π est $k-2$ parallélisable pour tout k.

Preuve : On fait un découpage « en bande », pour éviter le problème du carré situé sur deux bandes contigus, on fera se « chevaucher » les bandes de $c-1$ éléments.

Supposons que $(n_1 \geqq n_2)$. [On a un découpage symétrique quand $(n_1 < n_2)$.]



On a :

$$n_1 = \alpha k + r \qquad \text{où} \quad 0 < r \leqq k.$$

On décompose la matrice initiale en :

– $k-1$ sous-matrices caractérisées par

$$(1+u.(\alpha+1),\ 1,\ \alpha+c,\ n_2) \qquad \text{pour} \quad 0 \leqq u \leqq k-2$$

– la dernière étant :

$$(1+(k-1)(\alpha+1),\ 1,\ \alpha+1+r-k,\ n_2).$$

Soit $dd(n_1, n_2)$ cette décomposition, que l'on abrège ici par *dd*.

Chaque fonction B_i étudie s'il y a un carré de 1 de côté c dans sa sous matrice d'entrée. Φ est simplement une disjonction sur les B_i. On veut montrer que la décomposition est minimale, pour cela on étudie la valeur de λ.

$$\lambda(dd) = kn_2(\alpha + c) = n_2(n_1 - r + kc)$$

soit

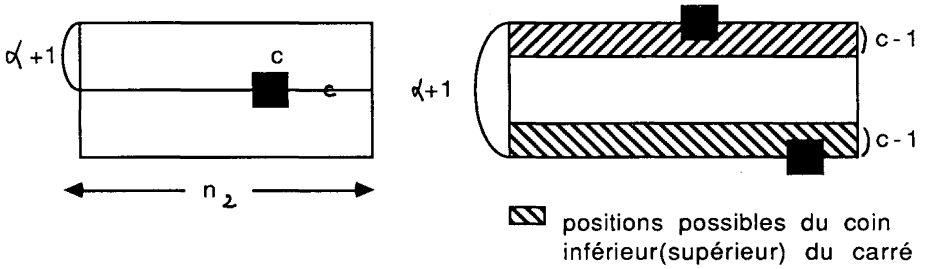
$$\lambda(dd) = n_2 n_1 + (kc - r) n_2$$

or $-k \leqq -r < 0$, donc il existe une constante S telle que :

$$\lambda(dd) \leqq n_2 n_1 + S n_2.$$

Remarque : Le recouvrement est $\rho(d) = (c-1) n_2$.

Si on voulait éliminer ce recouvrement pour chaque élément de la décomposition, il faut pouvoir coder d'une part le fait d'avoir ou non trouvé le carré dans son d'entrée d'autre part le fait d'avoir ou non un carré potentiel et donc il faut coder les positions possibles d'un carré chevauchant deux « bandes ».



On doit coder sur la bande supérieur le coin supérieur gauche du carré et sur la bande inférieur le coin inférieur droit.

Le nombre de positions possibles d'un coin est : $(c-1)(n_2 - c - 1)$.

On ne peut faire l'élimination du recouvrement par codage indépendamment de la taille des entrées.

3.3.2. *Problème de changement d'échelle*

On dispose d'une image qui est représentée par une matrice (n, n) d'entiers. On veut agrandir cette image, d'un facteur 2 par exemple et produire ainsi une image représentée par une matrice de taille $(2n, 2n)$. Pour ce faire on a

plusieurs possibilités :

– Faire une approximation par le plus proche voisin, en donnant au nouveau point la valeur de son voisin de gauche ($v=1$).

– Faire une interpolation linéaire, c'est-à-dire donner au nouveau point la moyenne des valeurs de ses voisins. On utilise deux voisins $v=2$.

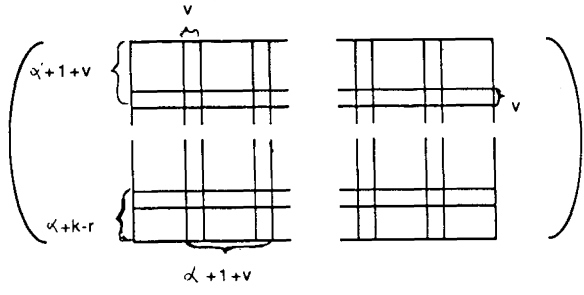
– Utiliser une convolution cubique, c'est-à-dire une approximation du nouveau point en utilisant ses quatre voisins ($v=4$).

Π_v a pour résultat vrai si la matrice M' est un agrandissement de la matrice M en utilisant l'approximation paramétrée par v.

Propriété 3.3.2.1 : Π_v est $k-2$ parallélisable pour tout k.

Preuve : Chaque fonction B_i étudie si sa deuxième entrée est bien un agrandissement de la première, en utilisant la méthode paramétrée par v. Il est bien entendu que si les dimensions de M' ne sont pas $(2n, 2n)$, on décompose M' de manière quelconque, et B_i aura comme réponse 0. Φ est simplement une conjonction sur les B_i.

On peut ici encore faire un découpage « en bande », avec un recouvrement de v pour M et sans recouvrement pour M'. Cette décomposition se traite de la même manière que précédemment. Aussi on choisit d'étudier un découpage en « carré » lorsque $k=K^2$



soit

$$n=\alpha K+r \qquad \text{où} \quad 0<r\leqq K$$

Les éléments de la décomposition sont donnés par :

– pour les sous matrices « centrales », pour $0\leqq u\leqq K-2$ et $0\leqq w\leqq K-2$

sur M

$$(1+u(\alpha+1),\ 1+w(\alpha+1),\ \alpha+1+v,\ \alpha+1+v)$$

sur M'

$$(1+u(2\alpha+1),\ 1+w(2\alpha+1),\ 2\alpha+1,\ 2\alpha+1)$$

– pour les sous matrices situées sur le bord inférieur, pour $0 \leqq w \leqq K-2$
pour M

$$(1+(K-1)(\alpha+1),\ 1+w(\alpha+1),\ \alpha+1-K+r+v,\ \alpha+1+v)$$

pour M'

$$(1+(K-1)(2\alpha+1),\ 1+w(2\alpha+1),\ 2\alpha+1-K+2.r,\ 2\alpha+1)$$

– pour les sous matrices situées sur le bord droit, pour $0 \leqq u \leqq K-2$
pour M

$$(1+u(\alpha+1),\ 1+(K-1)(\alpha+1),\ \alpha+1+v,\ \alpha+1-K+r+v)$$

pour M'

$$(1+u(2\alpha+1),\ 1+(K-1)(2\alpha+1),\ 2\alpha+1,\ 2\alpha+1-K+r)$$

– et enfin pour le dernier élément en bas à droite :
pour M

$$(1+(K-1)(\alpha+1),\ 1+(K-1)(\alpha+1),\ \alpha+1-K+r,\ \alpha+1-K+r+v)$$

pour M'

$$(1+(K-1)(2\alpha+1),\ 1+(K-1)(2\alpha+1),\ 2\alpha+1-K+r,\ 2\alpha+1-K+r).$$

Soit de cette décomposition. On va montrer que dc est minimal.

$$\theta(dc)=(\alpha+1+v)^2+(2\alpha+1)^2.$$

On a alors :

$$\lambda(dc)=k(\alpha+1+v)^2+k(2.\alpha+1)^2$$
$$\lambda(dc)=(n-r+K+v\,K)^2+(2n-2r+K)^2$$

en développant λ :

$$\lambda(dc) = n^2 + (2n)^2 + 2n(4K - 6r) + (K - r + vK)^2 + (K - 2r)^2$$

or $-K \leqq -r \leqq 0$, donc il existe une constante S telle que :

$$\lambda(dd) \leqq (n, 2n).(n, 2n) + S.2n.$$

4. CONCLUSION

On vient de donner un cadre général pour l'étude de la parallélisation d'un problème, on peut reprocher à ce modèle le manque de communication entre processus, il est donc intéressant d'étendre ce modèle en prenant en compte une architecture particulière permettant plus d'échanges entre processus (communication ou partage de mémoire). Une autre extension de ce travail consiste à s'intéresser à d'autres types de problèmes classiques (graphes, arbres...).

BIBLIOGRAPHIE

[1] A. Borodin, *On Relating Time and Space to Size and Depth*, S.I.A.M. J. Comput, vol. 6, n° 4, décembre 1977.

[2] S. A. Cook, *A Taxonimy of Problems with Fast Parallel Algorithms*, Information and Control, vol. 64, p. 2-22.

[3] S. A. Cook, C. Dwork et R. Reischuk, *Upper and Lower Time Bounds for Parallel Random Acces Machines without Simultaneous Writes*, S.I.A.M. J. Comput, vol. 15, n° 1, février 1986.

[4] W. D. Hillis, *The Connection Machine*, M.I.T., Artificial Intelligence Laboratory, Memo n° 646, septembre 1981.

[5] « IMS T424 transputer », INMOS (1984).

[6] J. Schwarz, *Ultracomputers*, TOPLAS 2, vol. 4, 1980, p. 454-521.

[7] *Introduction to Data Level Parallelism*, Thinking Machine Technical Report 86.14, avril 1986.

[8] U. Vishkin, *Synchronous Parallel Computation, Asurvey*, Courant Institut, New York University, avril 1983.

[9] U. Vishkin, *A Parallel-Design Distributed-Implementation (PDDI) General-Purpose Computer*, T.C.S. 32, 1984, p. 157-172.

[10] L. L. Welty et P. C. Patton, *Hypercube Architectures*, AFIP 85, vol. 54, 1985.