Exécution d'un graphe cubique de tâches sur un réseau bi-dimensionnel et asymptotiquement optimal

Clémentin Tayou Djamegni

Recherche

Exécution d'un graphe cubique de tâches sur un réseau 2D et asymptotiquement optimal

Clémentin Tayou Djamegni

Laboratoire d'Informatique
Faculté des Sciences
Université de Dschang
Cameroun
dtayou@yahoo.com, dtayou@cril.univ-artois.fr

RÉSUMÉ. Cet article présente une stratégie d'ordonnancement des graphes de tâches associés à une fonction de temps linéaire dans le contexte de la programmation parallèle. Cette stratégie d'ordonnancement est utilisée pour exécuter un graphe cubique de tâches, dont les tâches ont la même durée d'exécution et les temps de communications inter-tâches sont négligés, sur un réseau de processeurs bi-dimensionnel et asymptotiquement optimal par rapport à la fonction de temps. Ce résultat améliore la meilleure borne précédemment connue.

ABSTRACT. This work proposes a scheduling strategy, based on re-indexing transformations, for task graphs associated with a linear timing function. This scheduling strategy is used to execute a cubical task graph, for which all the tasks have the same execution time and inter-tasks communication delays are neglected, on a two-dimensional array of processors which is asymptotically space-optimal with respect to the timing function.

MOTS-CLÉS : ordonnancement; graphe de tâches;calcul parallèle; fonction de temps linéaire; fonction d'allocation; ré-indexation; optimalité

KEYWORDS : scheduling; task graph;parallel processing; linear timing function; allocation function; re-indexation; optimality

1. Introduction

Le problème d'ordonnancement des tâches est l'un des problèmes les plus importants en parallélisme. Dans le contexte de la programmation parallèle, une application est généralement modélisée par un graphe de tâches dont les noeuds représentent les tâches à exécuter et les arcs représentent les contraintes de précédence entre tâches. Le problème est de trouver un ordonnancement qui minimise à la fois le temps total d'exécution et le nombre de processeurs (PEs). Un tel ordonnancement est dit optimal. Une importante activité de recherche s'est développée autour de ce problème d'optimisation [1, 2, 3, 5, 6, 17, 18, 21, 23]. À cet effet, il convient de noter que ce problème est NP-difficile même dans le cas où le temps d'exécution d'une tâche et le temps de communication inter-tâches sont de une unité de temps et le nombre de PEs disponibles est non borné [17, 18]. De ce fait plusieurs travaux de recherche ont été consacrés à des cas particuliers de graphes de tâches [1, 2, 3, 5, 6, 21, 23].

Dans cet article nous étudions le cas particulier de ce modèle classique d'ordonnancement où le graphe de tâches est cubique et d'ordre $N \times N \times N$, le temps d'exécution d'une tâche est de une unité de temps, les dates d'exécution des tâches sont données par une fonction de temps linéaire $t(i, j, k) = ai + bj + ck$, où a, b et c sont des constantes entières non nulles, les temps de communication inter-taches sont négligés et l'architecture cible est un réseau bi-dimensionnel de PEs identiques. L'intérêt des graphes de tâches dits cubiques provient de ce qu'ils modélisent plusieurs algorithmes. À ce propos il est possible de citer la fermeture transitive [15, 22], le problème dit du chemin algébrique [20, 21] ainsi que plusieurs algorithmes relevant du traitement de signal et de l'algèbre linéaire tels que la multiplication matricielle [11, 19, 21], l'élimination de Gauss et la décomposition LU [7, 22]. De plus, plusieurs travaux de recherche sur la parallélisation des nids de boucles concernent les graphes cubiques de tâches [7]. Les fonctions de temps dites affines jouent un rôle particulier en parallélisation automatique dans la mesure où elles exhibent le parallélisme d'une part et facilitent la re-écriture du code d'autre part [7, 10, 14].

Pour simplifier la présentation nous supposons que $1 \leq a \leq b \leq c$ et $pgcd(a, b, c) = 1$. Ce choix sera justifié dans la section 3.1.

Les premiers pas vers la résolution de ce problème d'ordonnancement ont été établis dans [5, 23]. Dans [5], Cappello propose une allocation optimale par rapport à la fonction de temps t pour le cas particulier où $a = b = c = 1$. Dans [23], Tsay et Chang proposent trois allocations optimales par rapport à t pour les trois cas suivants : $(a+b \leq c)$, $(a+b > c \wedge a = b)$ et $(a + b > c \wedge b = c)$.

Dans cet article, nous généralisons les résultats précédents [5, 23]. Pour ce faire, nous introduisons une nouvelle technique d'allocation basée sur un pré-traitement par ré-indexation qui transforme le graphe initial de tâches en un nouveau graphe de tâches

qui permet d'obtenir, par la méthode d'allocation dite de projection [16], un réseau bi-dimensionnel et asymptotiquement optimal par rapport à la fonction de temps linéaire t. Les transformations par ré-indexation ont déjà été utilisées pour atteindre d'autres objectifs d'optimisation, notamment en parallélisation automatique. En particulier, plusieurs travaux de recherche ont été consacrés à la caractérisation des transformations par ré-indexation qui permettent d'explorer le parallélisme dans les nids de boucles [7, 8, 10, 14]. Dans notre travail, les transformations par ré-indexation sont utilisées pour atteindre un autre objectif d'optimisation : augmenter le parallélisme potentiel de la méthode d'allocation dite de projection dans le cas des graphes cubiques de tâches.

Le reste cet article est organisé comme suit. Nous faisons quelques rappels et définitions dans la section 2. La section 3 est consacrée à notre contribution. La section 4 conclut cet article.

2. Définitions

Définition 1 *Un ordonnancement d'un graphe de tâches $G = (D, A)$, où $D \subset \mathbb{Z}^n$ est l'ensemble des tâches et A est l'ensemble des arcs, est défini par une fonction de temps t et une fonction d'allocation $alloc$. La fonction de temps t donne la date d'exécution de chaque tâche de manière à respecter les contraintes de précédence, c'est-à-dire que $t(z) < t(z\prime)$ pour tout arc $[z \rightarrow z\prime]$ de G. La fonction d'allocation $alloc$ affecte les tâches aux PEs de manière à ce que deux tâches distinctes ayant la même date d'exécution soient exécutées sur deux PEs différents, c'est-à-dire que $alloc(z) \neq alloc(z\prime)$ si $t(z) = t(z\prime)$ et $z \neq z\prime$.*

Comme dans plusieurs travaux de recherche [1, 2, 3, 7, 8, 9, 11, 21, 22], nous supposons que le graphe de tâches G est borné, i.e. l'ensemble des tâches $D \subset \mathbb{Z}^n$ est fini, que les tâches ont la même durée d'exécution et que les temps de communications inter-tâches sont négligés.

Définition 2 *Le parallélisme potentiel d'une fonction de temps est le nombre maximum de tâches ayant la même date d'exécution. Une fonction d'allocation est dite optimale par rapport à une fonction de temps t si elle nécessite un nombre de PEs égal au parallélisme potentiel de t. Une fonction de temps est dite optimale si elle minimise le temps total d'exécution, en admettant que le nombre de PEs est non borné.*

Plusieurs travaux de recherche considèrent les ordonnancements affines [7, 8, 9, 11, 21, 22], i.e. des ordonnancements basés sur des fonctions de temps et allocations affines.

Définition 3 *Une fonction de temps t est dite affine si elle est de la forme $t : z \rightarrow \vec{\lambda} \bullet z + \alpha$ où $\vec{\lambda}$ est un vecteur constant appelé vecteur d'ordonnancement, α est un constante entière*

et " $\bullet$ " *est le signe du produit scalaire. Si* $\alpha = 0$*, la fonction de temps* t *est dite linéaire.* t *est dite normalisée si son vecteur d'ordonnancement* $\vec{\lambda}$ *est unimodulaire, c'est-à-dire que les composantes de* $\vec{\lambda}$ *sont premiers entre eux. Une fonction d'allocation* $alloc$ *est dite affine (ou de projection) si elle est obtenue en projetant le domaine des tâches suivant un vecteur constant* $\vec{\xi}$ *appelé direction d'allocation (ou de projection).*

Le vecteur d'ordonnancement est orthogonal aux hyperplans de $\mathbb{Z}^n$ contenant uniquement des points ayant une même date d'exécution. Comme dans [16], ces hyperplans seront appelés *surfaces d'ordonnancement*.

Proposition 1 *[7, 21, 22] Soit* $alloc$ *une fonction d'allocation affine de direction de projection* $\vec{\xi}$*. On a* $alloc(z) = alloc(z\prime)$ *si et seulement si les points* z *et* $z\prime$ *appartiennent à une même droite de direction* $\vec{\xi}$*.*

3. Notre Contribution

3.1. Une Hypothèse pour Simplifier la Présentation

Notons que tout ordonnancement d'un graphe de tâches associé à une fonction de temps linéaire $t\prime(i,j,k) = a\prime i + b\prime j + c\prime k$, où $a\prime$, $b\prime$ et $c\prime$ sont des entiers non nuls, peut être transformé par des permutations [13] en un ordonnancement équivalent associé à une fonction de temps linéaire $t''(i,j,k) = a''i + b''j + c''k$ qui satisfait $1 \leq a'' \leq b'' \leq c''$ et qui laisse inchangé le temps total d'exécution et le nombre de PEs utilisés. De plus, tout ordonnancement associé à une fonction de temps affine $t\prime(i,j,k) = a\prime i + b\prime j + c\prime k + d\prime$ peut être transformé en un ordonnancement équivalent associé à une fonction de temps linéaire. Sans perte de généralité, nous supposons que $1 \leq a \leq b \leq c$. Par ailleurs, pour des raisons de simplicité, nous supposons aussi que la fonction de temps linéaire $t(i,j,k) = ai+bj+ck, 1 \leq a \leq b \leq c$, est normalisée, c'est-à-dire que $pgcd(a,b,c) = 1$.

3.2. Notations et Définitions

Nous donnons ici quelques notations et définitions liées à la stratégie d'allocation qui sera introduite à la section 3.3.

$< \vec{i}, \vec{j}, \vec{k} >$ désigne la base canonique de $\mathbb{Z}^3$. Le résultat de l'addition d'un point $z = (z1, z2, z3)$ par un vecteur $\vec{v} = (v1, v2, v3)^t$ est le point $z + \vec{v} = (z1 + v1, z2 + v2, z3 + v3)$.

Définition 4 *Soit* $\vec{s}$ *un vecteur unimodulaire et soit* $D \subset \mathbb{Z}^3$ *un domaine borné, la frontière de* D *suivant la direction* $\vec{s}$ *est* $Frt(D,\vec{s}) = \{z \in D \mid z+\alpha\vec{s} \notin D$ *pour tout entier* $\alpha > 0\} \subseteq D$*.*

Définition 5 *Un graphe de tâche $G = (D, A)$ associé à une fonction de temps linéaire t est dit cubique d'ordre $N \times N \times N$ si l'ensemble des tâches est $D = \{(i, j, k) \in \mathbb{Z}^3 \mid 0 \leq i, j, k \leq n\}$ où $n = N - 1$, l'ensemble des arcs est $A = \{[z \rightarrow z + \vec{d}] \mid z \in D,\ z + \vec{d} \in D,\ \vec{d} \in V\}$ où V est un ensemble de vecteurs constants appelés vecteurs de dépendance et $t(z) < t(z + \vec{d})$ pour tout couple $(z, d) \in D \times V$.*

Proposition 2 *[7, 21, 22] Nous avons $t(\vec{d}) > 0$ pour tout vecteur $\vec{d} \in V$.*

La définition 5 généralise la définition de grille tridimensionnelle donnée dans [2, 3]. Dans [2, 3], l'ensemble V des vecteurs de dépendance est égal à l'ensemble des vecteurs de la base canonique, c'est-à-dire que $V = \{\vec{i}, \vec{j}, \vec{k}\}$. Étant donné que les temps de communications inter-tâches sont négligés, ce cas particulier admet $t(i, j, k) = i + j + k$ comme fonction de temps optimale.

Définition 6 *Une fonction $\psi : D \rightarrow \psi(D)$, $D \subset \mathbb{Z}^n$, est une fonction de ré-indexation valide si elle correspond à une transformation bijective.*

Comme nous l'avons dit en introduction, la stratégie d'allocation qui sera présentée à la section 3.3 est basée sur la transformation du graphe initial de tâches, par ré-indexation, en un nouveau graphe de tâches qui se prête mieux à la méthode d'allocation dite de projection [7, 11, 21, 22]. Comme la méthode de projection admet que les tâches ayant une même date d'exécution appartiennent à un même hyperplan [7, 16, 21, 22], nous ne considérerons que les fonctions de ré-indexation qui préservent cette propriété.

3.3. Une Allocation Asymptotiquement Optimale

Dans cette section nous proposons une stratégie d'allocation optimale des tâches d'un graphe cubique de tâches d'ordre $N \times N \times N$ associé à une fonction de temps linéaire $t(i, j, k) = ai + bj + ck$. Cette stratégie peut être déroulée pas à pas en trois phases. Soit $D_0 = \{(i, j, k) \in \mathbb{Z}^3 \mid 0 \leq i, j, k \leq n\}$, où $n = N - 1$, le domaine des tâches. Soit $\beta = pgcd(a, c)$. Nous avons $a = \delta\beta$ et $c = \gamma\beta$. Pour simplifier la présentation nous supposons que $n \pmod{2b\beta\delta\gamma} = 0$.

3.3.1. Phase 1

Dans cette phase nous appliquons à D_0 une fonction de ré-indexation q qui transforme le vecteur d'ordonnancement initial $a\vec{i} + b\vec{j} + c\vec{k}$ en un nouveau vecteur $\vec{k}$. Pour ce faire nous posons :

$$q(i, j, k) = (i, j, ai + bj + ck) \qquad [1]$$

La fonction q est une bijection de D_0 vers D_1, où $D_1 = q(D_0) = \{(i, j, k) \in \mathbb{Z}^3 \mid 0 \leq i \leq n,\ 0 \leq j \leq n,\ 0 \leq -ai - bj + k \leq cn,\ (-ai - bj + k) \bmod c = 0\}$. Cette fonction

de ré-indexation conduit à un nouvelle fonction de temps linéaire $t\prime(i,j,k) = k$. Une propriété de cette nouvelle fonction de temps est que les surfaces d'ordonnancement sont maintenant parallèles au plan horizontal, en supposant que la direction de $\vec{k}$ est verticale.

On vérifie facilement la propriété suivante :

Proposition 3 *Si $z \in D_1$ et $z + \lambda\vec{i} \in D_1$ alors $\lambda \bmod \gamma = 0$.*

3.3.2. Phase 2

Dans cette phase les points de D_1 sont ré-indexés de manière à ramener tous les points de $Frt(D_1, -\vec{i})$ sur le plan d'équation cartésienne $i = 0$ et à compresser D_1, par un facteur de γ, suivant la direction $\vec{i}$. Pour ce faire nous utilisons la fonction de ré-indexation suivante :

$$h(i,j,k) = \begin{cases} (\lfloor \frac{i}{\gamma} \rfloor, j, k) & \text{si } -bj + k - cn \leq 0 \\ (\lfloor \frac{cn+ai+bj-k}{a\gamma} \rfloor, j, k) & \text{si } -bj + k - cn > 0 \end{cases} \quad [2]$$

La fonction h maintient chaque point de D_1 sur sa surface d'ordonnancement. Donc h laisse inchangée la fonction de temps linéaire $t\prime(i,j,k) = k$. Par ailleurs nous avons les propriétés suivantes :

Proposition 4 *La fonction de ré-indexation h est une bijection de D_1 vers $h(D_1)$.*

PREUVE. Soient (i,j,k) et $(i\prime,j\prime,k\prime)$ deux points de D_1. Supposons que $h(i,j,k) = h(i\prime,j\prime,k\prime)$. Ceci entraîne que

$$\begin{cases} j = j\prime \\ k = k\prime \\ \lfloor \frac{i}{\gamma} \rfloor = \lfloor \frac{i\prime}{\gamma} \rfloor \text{ ou } \lfloor \frac{cn+ai+bj-k}{a\gamma} \rfloor = \lfloor \frac{cn+ai\prime+bj\prime-k\prime}{a\gamma} \rfloor \end{cases}$$

Ceci entraîne que $|i - i\prime| < \gamma \vee |(cn + ai + bj - k) - (cn + ai\prime + bj\prime - k\prime)| < a\gamma$. Il vient que $|i - i\prime| < \gamma$. Donc $(i,j,k) = (i\prime,j\prime,k\prime) + \lambda\vec{i}$ pour un certain entier λ tel que $|\lambda| < \gamma$. Par ailleurs la proposition 3 entraîne que $\lambda = 0$. Ce qui signifie que $(i,j,k) = (i\prime,j\prime,k\prime)$. □

Proposition 5 *Nous avons $h(D_1) \subseteq D_2$ où $D_2 = \{(x,y,z) \in \mathbb{Z}^3 \mid 0 \leq \gamma x \leq n,\ 0 \leq y \leq n,\ (a\gamma)x + by - z \leq 0,\ (a\gamma)x - by + z \leq (a+c)n,\ (-by + z) \bmod \beta = 0\}$.*

PREUVE. Soient $(i,j,k) \in D_1$ et$(x,y,z) = h(i,j,k)$. Nous avons :

$$0 \leq i \leq n \quad [3]$$

$$0 \leq j \leq n \quad [4]$$

$$0 \leq -ai - bj + k \leq cn \quad [5]$$

$$-ai - bj + k \,(\mathrm{mod}\ c) \ = \ 0 \quad [6]$$

Il est clair que, (4) entraîne que $0 \leq y \leq n$ et (6) entraîne que $(-by+z) \mod \beta = 0$. D'après (2) nous considérons deux cas.

Cas 1 : $-bj+k-cn \leq 0$: (3) entraîne que $0 \leq \gamma x \leq n$, (5) entraîne que $(a\gamma)x+by-z \leq 0$ et $-bj+k-cn \leq 0$ entraîne que $(a\gamma)x - by + z \leq (a+c)n$.

Cas 2 : $bj - k + cn < 0$: (5) entraîne que $0 \leq \frac{cn+ai+bj-k}{a\gamma}$ qui à son tour entraîne que $0 \leq x$. Par ailleurs, $bj - k + cn < 0$ entraîne que $ai + bj - k + cn \leq an$ qui à son tour entraîne que $\gamma x \leq n$. On en déduit que $0 \leq \gamma x \leq n$.
Puisque $i = \gamma\frac{cn+ai+bj-k}{a\gamma} + \frac{-cn-bj+k}{a}$, (3) entraîne que $\gamma x + \frac{-cn-by+z}{a} \leq n$, c'est à dire que $(a\gamma)x - by + z \leq (a+c)n$. (5) entraîne que $cn + ai + bj - k \leq cn$ qui à son tour entraîne que $(a\gamma)x \leq cn$. Par ailleurs $cn + by - z < 0$. On en déduit que $(a\gamma)x + by - z \leq 0$. □

On vérifie facilement la propriété suivante :

Proposition 6 *Si $z \in D_2$ et $z + \lambda\vec{j} \in D_2$ alors $\lambda \bmod \beta \ = \ 0$.*

La projection de D_2 suivant la direction $\vec{k}$ conduit à un réseau rectangulaire d'ordre $(\lfloor\frac{n}{\gamma}\rfloor + 1) \times (n+1)$. Le nombre de PEs de ce réseau améliore, d'un facteur de γ, celui du réseau carré d'ordre $(n+1) \times (n+1)$ obtenu en projectant le domaine initial D_0 suivant la même direction $\vec{k}$. Cette amélioration en nombre de PEs justifie la ré-indexation.

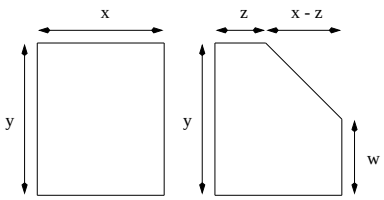
3.3.3. Phase 3

On peut encore réduire le nombre de PEs du réseau obtenu à la phase 2. Pour ce faire, les points de D_2 sont ré-indexés de manière à ramener les points de $Frt(D_2, -\vec{j})$ sur le plan d'équation cartésienne $j = 0$ et à compresser D_2 , par un facteur de β, suivant la direction $\vec{j}$. Pour ce faire, nous utilisons la fonction de ré-indexation suivante :

$$g(i,j,k) = \begin{cases} (i, \lfloor\frac{j}{\beta}\rfloor, k) & \text{si } (a\gamma)i + k - (a+c)n \leq 0 \\ (i, \lfloor\frac{-a\gamma i+bj-k+(a+c)n}{b\beta}\rfloor, k) & \text{si } (a\gamma)i + k - (a+c)n > 0 \end{cases} \quad [7]$$

La fonction de ré-indexation g laisse inchangée la fonction de temps linéaire $t\prime(i,j,k) = k$. Par ailleurs, on vérifie facilement la propriété suivante :

Proposition 7 *La fonction g est une bijection de D_2 vers $g(D_2)$.*



$$x = \lfloor \frac{n}{\gamma} \rfloor + 1 \quad y = \lfloor \frac{n}{\beta} \rfloor + 1 \quad z = \frac{(a-b+c)n}{2a\gamma} \quad w = \frac{(c-a)n}{b\beta}$$

Figure 1. *Formes du nouveau réseau*

Proposition 8 *Nous avons* $g(D_2) \subseteq D_3$ *où* $D_3 = \{(x,y,z) \in \mathbb{Z}^3 \mid 0 \leq \gamma x \leq n,\ 0 \leq \beta y \leq n,\ (2a\gamma)x + (b\beta)y \leq (a+c)n\}$.

PREUVE. Soient $(i,j,k) \in D_2$ et $(x,y,z) = g(i,j,k)$. Nous avons :

$$0 \leq \gamma i \leq n \qquad [8]$$

$$0 \leq j \leq n \qquad [9]$$

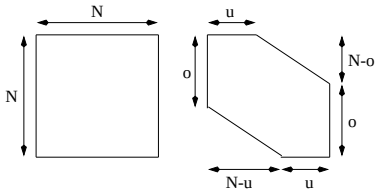
$$(a\gamma)i + bj - k \leq 0 \qquad [10]$$

$$(a\gamma)i - bj + k \leq (a+c)n \qquad [11]$$

Commençons par noter que (8) entraîne que $0 \leq \gamma x \leq n$. Sur la base de 7 nous considérons deux cas :

Cas 1 : $(a\gamma)i + k \leq (a+c)n$: (9) entraîne que $0 \leq \beta y \leq n$ et (10) entraîne que $(a\gamma)x + b\beta y - z \leq 0$. Par ailleurs, $(a\gamma)i + k \leq (a+c)n$. On en déduit que $(2a\gamma)x + (b\beta)y \leq (a+c)n$.

Cas 2 : $-(a\gamma)i - k \leq -(a+c)n$: $-(a\gamma)i - k \leq -(a+c)n$ entraîne que $-(a\gamma)i + bj - k + (a+c)n \leq bn$, et il vient que $\beta y \leq n$. Par ailleurs, (11) entraîne que $0 \leq y$. On en déduit que $0 \leq \beta y \leq n$.
Puisqu'on a $(a\gamma)i + bj - k = \frac{-(a\gamma)i+bj-k+(a+c)n}{b\beta} b\beta + (2a\gamma)i - (a+c)n$, (10) entraîne que $(b\beta)y + (2a\gamma)x - (a+c)n \leq 0$. □



(a) a + b <= c (b) a + b > c

$$o = \frac{(-a+b+c)n}{2b} \quad u = \frac{(a-b+c)n}{2a}$$

Figure 2. *La forme de* $u\prime(\frac{(a+b+c)n}{2})$

Considérons maintenant le réseau obtenu en projetant le domaine D_3 suivant la direction $\vec{k}$. La forme de ce réseau est illustrée à la figure 1. Le nombre de PEs est de l'ordre de

$$\frac{n^2}{c} + \Theta(n)$$

si

$$a + b \leq c$$

et

$$\left(\frac{1}{c} - \frac{(a+b-c)^2}{4abc}\right) n^2 + \Theta(n)$$

dans le cas contraire. Ce qui représente une réduction significative, d'un facteur de β, par rapport à la taille du réseau obtenu à la phase 2. De plus, nous avons le résultat de complexité suivant :

Theorème 1 *La projection de* $g(h(q(D_0)))$ *suivant la direction* $\vec{k}$ *conduit à un réseau asymptotiquement optimal en nombre de PEs.*

PREUVE. Nous allons montrer que le nombre de PEs est asymptotiquement optimal par rapport à la fonction de temps $t\prime$, c'est à dire que asymptotiquement, la fonction d'allocation induit autant de PEs que la fonction de temps linéaire $t\prime$. Pour ce faire, il suffit d'exhiber une date d telle que

$$|u(d)| = \begin{cases} \frac{n^2}{c} + \Theta(n) & \text{si } a+b \leq c \\ \left(\frac{1}{c} - \frac{(a+b-c)^2}{4abc}\right) n^2 + \Theta(n) & \text{si } a+b > c \end{cases}$$

où

$$u(d) = \{z \in D_1 \mid t\prime(z) = d\}.$$

Posons $u\prime(d) = \{(i,j,k) \in \mathbb{Z}^3 \mid 0 \leq i \leq n,\ 0 \leq j \leq n,\ 0 \leq -ai - bj + k \leq cn,\ k = d\}$. Partitionnons $u(d)$ et $u\prime(d)$ en une série de segments parallèles au vecteur $\vec{i}$, $u(d) = \bigcup_{f=0}^{f=n} u(d,f)$ et $u\prime(d) = \bigcup_{f=0}^{f=n} u\prime(d,f)$ où $u(d,f) = \{(i,j,k) \in u(d) \mid j = f\}$ et $u\prime(d,f) = \{(i,j,k) \in u\prime(d) \mid j = f\}$.

Montrons d'abord que :

$$|u(d,f)| = \begin{cases} 0 & \text{si } f \not\equiv f_0 \pmod{\beta} \\ \left\lfloor \frac{|u\prime(d,f)|}{\gamma} \right\rfloor & \text{si } f \equiv f_0 \pmod{\beta} \end{cases} \qquad [12]$$

où f_0 est tel que $-bf_0 + d \equiv 0 \pmod{\beta} \wedge |f_0| < \beta$.

Justifions par réfutation qu'un tel entier f_0 existe. Supposons que $(-bl+d) \not\equiv 0 \pmod{\beta}$ pour tout entier $l \in \{0,\ 1,\ 2,\ , \ldots , \beta - 1\}$. Ceci entraîne qu'il existe deux entiers distincts l_1 et l_2 tels que $(-bl_1 + d) \equiv (-bl_2 + d) \pmod{\beta}$ et $0 \leq l_1,\ l_2 \leq \beta - 1$. Il vient que $l_1 - l_2 \equiv 0 \pmod{\beta}$ puisque $pgcd(b,\beta) = pgcd(a,b,c) = 1$. Ce qui entraîne que $l_1 - l_2 = 0$ puisque $|l_1 - l_2| < \beta$. D'où la contradiction.

Pour montrer (12) nous considérons deux cas.

Cas 1 : $f \not\equiv f_0 \pmod{\beta}$: On vérifie facilement par réfutation que $u(d,f) = \emptyset$.

Cas 2 : $f \equiv f_0 \pmod{\beta}$: La proposition 3 entraîne que $|u(d,f)| \leq \left\lfloor \frac{|u\prime(d,f)|}{\gamma} \right\rfloor$. Il nous reste donc à montrer que $\left\lfloor \frac{|u\prime(d,f)|}{\gamma} \right\rfloor \leq |u(d,f)|$. Pour ce faire, il suffit de justifier que : Pour toute suite de γ points successifs $(i,f,d) + x\vec{i},\ x = 0,\ 1,\ 3,\ \ldots,\ \gamma - 1$ de $u\prime(d,f)$, un et un seul de ces points appartient $u(d,f)$. On a $f = \beta f_1 + f_0$ et $-bf_0 + d = \beta f_2$. On vérifie facilement que $(-\delta(i + x_0) - bf_1 + f_2) \equiv 0 \pmod{\gamma}$ pour un certain $x_0 \in \{0,\ 1,\ 3,\ \ldots,\ \gamma - 1\}$. Ceci entraîne que $(-\beta\delta(i+x_0) - \beta bf_1 + \beta f_2) \equiv 0 \pmod{\beta\gamma}$. Il vient que $(-a(i+x_0) - bf + d) \equiv 0 \pmod{c}$. On en déduit que $(i+x_0, f, d) \in u(d,f)$. L'unicité est une conséquence immédiate de la proposition 3.

(12) entraîne que

$$|u(d)| = \sum_{w=0}^{w=\lfloor \frac{n-f_0}{\beta} \rfloor} \left\lfloor \frac{|u\prime(d, f_0 + \beta w)|}{\gamma} \right\rfloor . \qquad [13]$$

Maintenant, considérons la date $d = (a+b+c)n/2$. La forme de $u\prime((a+b+c)n/2)$ est illustrée à la figure 2. (13) entraîne que :

$$\left|u\left(\frac{(a+b+c)n}{2}\right)\right| = \frac{1}{\beta\gamma}\left|u\prime\left(\frac{(a+b+c)n}{2}\right)\right| + \Theta(n).$$

$$= \begin{cases} \frac{n^2}{c} + \Theta(n) & \text{si } a+b \leq c \\ \left(\frac{1}{c} - \frac{(a+b-c)^2}{4abc}\right) n^2 + \Theta(n) & \text{si } a+b > c \end{cases}$$

□

3.3.4. Application

Pour le problème de calcul du produit $C = AB$ de deux matrices carrées A et B d'ordre N le graphe de tâches est un cube d'ordre $N \times N \times N$ [11, 19, 23, 22]. Ce problème admet $t(i,j,k) = i+j+k$ comme fonction de temps optimal, ce qui correspond au cas où $a+b > c$. De même, pour le problème dit du chemin algébrique [20] le graphe des tâches est également cubique, et $t(i,j,k) = i+j+3k$ est une fonction de temps optimale, ce qui correspond au cas où $a+b \leq c$.

4. Conclusion

Dans cet article nous avons étudié l'ordonnancement des graphes cubiques de tâches associés à une fonction de temps linéaire. En particulier, la borne obtenue améliore, en terme de nombre de PEs, la meilleure borne précédemment connue [23]. Ce meilleur réseau est obtenu en transformant le graphe initial de tâches par ré-indexation avant d'appliquer la méthode d'allocation dite affine. Le pré-traitement par ré-indexation améliore le parallélisme de la méthode de projection par rapport au domaine d'indexation initial. Cette nouvelle méthode d'allocation exploite des propriétés de convexité du graphe de dépendances. Comme cette stratégie d'allocation est basée sur des ré-indexations et allocations affines, on pourrait envisager son intégration dans des outils de parallélisation (semi)-automatique [4, 9, 11, 12]. À ce propos, il faudrait d'abord étudier la question suivante : Étant donné un graphe quelconque de tâches associé à une fonction de temps affine, comment construire des fonctions de ré-indexation qui conduisent à des allocations optimales ?

5. Bibliographie

[1] E. Bampis, J. -C. König, D. Trystram, Optimal parallel execution of complete binary trees and grids into most popular interconnection networks, *Theoretical Computer Science*, 147 (1995) 1-18.

[2] E. Bampis, C. Delorme, J. -C. König, Optimal schedules for d-D grid graphs with communication delays, *Parallel Computing*, 24 (11) (October 1998) 1653-1664

[3] E. Bampis, J. -C. König, D. Trystram, Minimizing the schedule length for a parallel 3D-grid precedence graph, *European Journal of Operational Research*, 95 (2) (1996) 427-438.

[4] M. Bednara, J. Teich, Interface Synthesis for FPGA Based VLSI Processor Arrays, In Proc. of The International Conference on Engineering of Reconfigurable Sytsems and Algorithms (ERSA02), Las Vegas, Nevada, U.S.A., (June 2002) 24-27.

[5] P. Cappello, A processor-time-minimal systolic array for cubical mesh algorithms, *IEEE Trans. on Parallel and Distributed Systems*, 3 (1) (January 1992) 4-13.

[6] P. Chretienne, C. Picouleau, Scheduling with communication delays : A survey, in Scheduling Theory and its Applications, John Wiley & Sons, (1995) 65-90.

[7] A. Darte, Y. Robert, F. Vivien, Scheduling and Automatic Parallelization, *Birkhäuser Boston*, 2000.

[8] A. Darte, Y. Robert, F. Vivien, Loop Parallelization Algorithms, *in volume 1808 of LNCS on Compiler Optimizations for Scalable Parallel Systems : Languages, Compilation Techniques, and Run Time Systems*, Dharma P. Agrawal and Santosh Pande editors, Springer Verlag, (2001) 141-172.

[9] F. Dupont de Dinechin, M. Manjunathaiah, T. Risset, M. Spivey, Design of Highly Parallel Architectures with Alpha, *Forum on Specifi cation & Design Langages, FDL*, Marseille, (September 2002).

[10] P. Feautrier, Dataflow analysis of array and scalar references, *International Journal Of Parallel Programming*, 20 (1) (1991) 23-53.

[11] A. C. Guillou, P. Quinton and T Risset, Hardware Synthesis for Systems of Recurrence Equations with Multi-Dimensionnal Schedule, *International Journal of Embedded Systems* (IJES), To appear (2006).

[12] M. Griebl, C. Lengauer, The loop parallilizer LooPo, In Proc. *Sixth Workshop On Compilers For Parallel Computers*, M. Gerndt, Ed. Konferenzen des Forschungszentrums Jülich, 21 (1996) 311-320.

[13] Y.C. Hou, J.C. Tsay, Equivalent transformations on systolic design represented by generating functions, *J. of Information Science and Eng.*, 5 (Mar. 1992) 229-250.

[14] A.W. Lim, M.S. Lam, Maximizing parallelism and minimizing synchronization with affine partitions, *Parallel Computing*, 24 (3-4) (May 1998) 445-475

[15] Igor Z. Milovanovic, Emina I. Milovanovic, B. M. Randjelovic, Computing Transitive Closure Problem on Linear Systolic Array, LNCS 3401 (2005) 416-423.

[16] C. Mongenet, Data compiling for systems of uniform recurrence equations, *Parallel Processing Letters*, 4 (3) (1994) 245-257.

[17] C. Picouleau, Etude des problèmes d'optimisation dans les systèmes distribués, *Thèse de Doctorat*, Université de Paris VI, France, (1992).

[18] V.J. Rayward-Smith, UET scheduling with unit interprocessor communication delays, *Discrete Applied Mathematics*, 18 (1987) 55-71.

[19] N.M. Stojanovic, E.I. Milovanovic, I. Stojmenovic, T. Milovanovic, T.I. Tokic, Mapping matrix multiplication algorithm onto fault-tolerant systolic array, *Computers & Mathematics with Applications*, 48 (1-2) (July 2004) 275-289.

[20] C. Tayou Djamegni, P. Quinton, S. Rajopadhye, T. Risset, Derivation of Systolic Algorithms For the Algebraic Path Problem by Recurrence Transformations, *Parallel Computing*, 26 (11) (2000) 1429-1445.

[21] C. Tayou Djamegni, Méthode d'optimisation pour la synthèse d'architectures régulières, *Thèse de Doctorat d'Etat*, Département d'Informatique, Université de Yaoundé I, (Dec. 2005).

[22] M. Tchuente, Parallel Computation on regular arrays, *Manchester University Press and John Wiley Sons*, (1992).

[23] J.C. Tsay, P.Y. Chang, Design of space-optimal regular arrays for algorithms with linear schedules, *IEEE Trans. on Computers*, 44 (5) (May 1995) 683-694.

Remerciements : Je voudrai remercier les lecteurs et l'éditeur pour leurs critiques et suggestions judicieux qui ont permis d'améliorer la présentation de cet article.

Clémentin Tayou Djamegni a obtenu le DEA, le Doctorat de Troisième Cycle et le Doctorat d'Etat au Département d'Informatique de la Faculté des Sciences de l'Université de Yaoundé I (Cameroun) respectivement en 1995, 1997 et 2005. Il est enseignant-chercheur à l'Université de Dschang (Cameroun) depuis 1996. Durant la période 1996-1998, il a été deux fois chercheur invité à l'IRISA (France) et durant la période 2001-2005 il a été plusieurs fois maître de conférences invité à la Faculté des Sciences Jean Perrin (France). Ses centres d'intérêt concernent la parallélisation d'algorithmes, les réseaux réguliers, l'analyse formelle de concepts, les algorithmes distribués et les systèmes distribués.