Formulations pour résoudre le problème de ordonnancement "modulo" sous contraintes de ressources

Maria AYALA*

Directeur(s) de thèse : Christian ARTIGUES

Laboratoire d'accueil :
LAAS-CNRS
7, avenue du Colonel Roche 31077 Toulouse Cedex 4

Établissement d'inscription :
Université de Toulouse
118, Route de Narbonne
F-31062 Toulouse cedex 9

Résumé

Le problème d'ordonnancement cyclique consiste à ordonner dans le temps l'exécution répétitive (boucle) d'un ensemble d'opérations liées par des contraintes de précédence, en utilisant un nombre limité de ressources. L'ordonnancement modulo est une structure d'ordonnancements cycliques 1-périodiques avec période entière qui crée des liens entre les itérations successives d'une boucle. Dans ce travail, nous montrons les differents formulations de programmation linéaire en nombres entiers et de leur resolution par des techniques de séparation et génération de colonnes pour trouver un ordonnancement valide pour une boucle afin qu'elle puisse être superposée plusieurs fois dans un intervalle constant, que nous appelons intervalle d'initiation ou période λ.

Mots-clés

ordonnancement d'instructions, ordonnancement "modulo", Programmation linéaire en nombres entiers, génération de colonnes.

1 INTRODUCTION

Beaucoup d'études dans le domaine de la compilation pour les processeurs superscalaires modernes et architectures VLIW (Very Large Instruction Word) sont focalisées sur le problème d'ordonnancement d'instructions. Ce problème est défini par un ensemble d'opérations à ordonnancer, un ensemble de dépendances entre ces opérations et la micro architecture du processeur cible qui définit des contraintes des ressources complexes. Une opération est considérée comme une instance d'une instruction dans un programme. L'ordonnancement d'instructions par boucles internes est connu sous le nom de pipeline logiciel. Le pipeline logiciel est une méthode efficace pour l'optimisation de boucles qui permet la réalisation en parallèle des opérations des différentes itérations de la boucle. Aujourd'hui, les compilateurs commerciaux utilisent des méthodes basées sur les algorithmes d'ordonnancement modulo. L'ordonnancement "modulo" est une structure d'ordonnancements cycliques 1-périodique avec période entière. Cela rend le modèle plus simple que le modèle classique d'ordonnancement cyclique et aussi plus facilement implémentable. Le modulo-ordonnancement est une technique qui crée des liens entre les itérations successives d'une boucle. Le but est de trouver un ordonnancement valide pour une boucle pour qu'elle puisse être supperposée plusieurs fois

*mayala@laas.fr

dans un intervalle constant, que nous appelons intervalle d'initiation ou période (λ). L'algorithme d'ordonnancement modulo doit prendre en compte les contraintes de dépendances des opérations, les contraintes de ressources (et la taille des registres qui sont ignorés dans cette étude). Il doit aussi considérer les critères d'optimisation tels que la minimisation de l'intervalle d'initiation de l'ordonnancement λ ou la minimisation de la durée de l'ordonnancement d'une itération de la boucle.

Le problème d'ordonnancement modulo consiste à trouver les dates de début d'un ensemble d'opérations génériques de telle sorte que dans le contexte cyclique, les instances des opérations génériques sont répétées toutes les λ unités de temps. La date de début d'une instance d'opération est donc définie par deux valeur, la date de début σ_i dans une période générique, variant de 0 à $\lambda - 1$ et le numéro k_i de la période. L'aspect modulo permet de limiter le nombre d'opérations à considérer aux seules opérations génériques. Les ressources sont disponibles en quantités limitées et chaque opération demande une quantité positive ou nulle de chaque ressource. Les demandes cumulées des opérations en cours d'exécution à chaque instant modulo λ sur chaque ressource ne doivent pas dépasser sa limitation. Les contraintes de dépendances sont définies relativement aux dates de début "effectives" $\sigma_i + k_i\lambda$.

Dans le contexte de la programmation linéaire en nombres entiers (PLNE) pour l'ordonnancement modulo avec contrainte de ressources, différentes formulations ont été proposées, basées sur des généralisation de la formulation classique non préemptive et indexée par le temps de Pritsker et al. [3] pour le RCPSP (Resource Constrained Project Scheduling Problem). Eichenberger et Davidson [1] proposent une formulation comportant deux ensembles de variables : une variable binaire $y_{i\tau}$ donnant la date de début de l'opération i dans la période générique (telle que $\sigma_i = \sum_{\tau=0}^{\lambda-1} \tau y_{i\tau}$) et la variable entière k_i. Dupont-de-Dinechin et al. [2] donnent une formulation comportant un seul type de variable binaire x_{it} tel que $y_{i\tau} = \sum_{k=0}^{K-1} x_{i,\tau+k\lambda}$ et $k_i = \sum_{k=1}^{K-1} \sum_{t=k\lambda}^{K\lambda-1} x_{it}$ avec K et λ donnés. Dans les deux cas, suivant l'approche de Christofides et al. [5], des contraintes de précédence structurées et non structurées sont présentées. Mingozzi et al. [4] présentent une formulation 0-1 pour le RCPSP qui nécessite un grand nombre de variables correspondant à des sous-ensembles d'activités qui peuvent être exécutées simultanément sans violer les contraintes de ressources ou de précédence. Dans ce travail, nous proposons une comparaison expérimentale des deuse formulations ainsi qu'une extensions de la formulation de Mingozzi pour le RCMSP (Problème de l'ordonnancement modulo sous contraintes de ressources). Nous proposons un schéma de génération de colonnes pour résoudre la relaxation du PLNE. Les résultats expérimentaux pour comparer les formulations sont réalisés en utilisant des instances de RCMSP provenant du compilateur commercial ST200.

2 Problème de l'ordonnancement modulo sous contraintes de ressources (RCMSP)

L'ordonnancement modulo est une structure d'ordonnancements cycliques 1-périodiques avec période entière $\lambda \geq 1$. Un ordonnancement 1-périodique satisfait :

$$\sigma_i^k = \sigma_i^0 + k\lambda, \forall i \in [1, n], \forall k \geq 0 \tag{1}$$

Si E représente les dépendances entre opérations et $\sigma_i = \sigma_i^0$, chaque contrainte de dépendence uniforme est exprimée comme :

$$\sigma_i + \theta_i^j - \omega_i^j \lambda \leq \sigma_j, \forall (i, j) \in E \tag{2}$$

Où θ_i^j, la latence, représente le longueur de la dépendance et ω_i^j, la distance, représente le nombre d'itérations qui séparent les instances de i et j

Chaque contrainte de ressource s'exprime comme :

$$\sum_{O_i \in A(t)} b_i^s \leq B_s, \forall s \in [1,m], \forall t \in [0,\lambda[\tag{3}$$

où $A(t) = \{O_i | \exists c \in R_i^s, \sigma_i = (t-c) mod \lambda\}$. b_i^s est la quantité de ressource s qui est nécessaire pour l'exécution de l'opération i et B_s donne le nombre d'unites de la ressource s disponibles.

Le problème de minimisation est toujours une fonction des dates de debut. Si, w_i représente le coût associé à chaque opération i, nous considerons le problème de minimisation de la somme pondérée des dates d'exécution des opérations à durées unitaires pour un intervalle d'initiation donné :

$$min \sum_{i \in [1,n]} w_i \sigma_i \tag{4}$$

Si nous introduisons une opération $n+1$ qui représente la fin de l'ordonnancement avec $\sigma_{n+1} \geq \sigma_i + p_i,\ i = 1, ..., n$, l'objectif peut être aussi, une fonction du temps de fin de l'opération $n+1$, c'est à dire la durée totale ou Makespan (noté Cmax).

3 Programmation linéaire en nombres entiers pour l'ordonnancement modulo sous contraintes de ressources

Pritsker [3] et ensuite Christofides [5] proposent une nouvelle formulation pour l'ordonnancement de projet sous contraintes de ressources. Cette formulation a été étendue par Eichenberger [1] et Dupont-de-Dinechin [2] pour le problème de RCMSP.

3.1 Formulations d'Eichenberger et al. Formulation décomposée (FD) [1]

Cette formulation utilise la décomposition $\sigma_i = k_i * \lambda + \tau_i$ où $k_i \in N$ est le numéro du cycle dans lequel chaque opération est placée. $\tau_i \in [0, \lambda - 1]$ est la date de debut dans l'intervalle $[0, \lambda - 1]$. Cette formulation est basée sur les variables binaires y_i^τ tell que, $\tau_i = \sum_{\tau=0}^{\lambda-1} \tau y_i^\tau$, $i = 1, ..., n$. La formulation (FD) s'exprimée par :

$$\sum_{\tau=0}^{\lambda-1} y_i^\tau = 1, \forall i \in [1,n] \tag{5}$$

$$\sum_{\tau=0}^{\lambda-1} \tau y_i^\tau + k_i \lambda + \theta_i^j - \lambda \omega_i^j \leq \sum_{\tau=0}^{\lambda-1} \tau y_j^\tau + k_j \lambda, \forall (i,j) \in E \tag{6}$$

$$\sum_{i=1}^{n} y_i^\tau b_i^s \leq B_s, \forall s \in m, \tau \in [0,\lambda) \tag{7}$$

Eichenberger propose une nouvelle contrainte de dépendance appelée contrainte de dépendance structurée (FDS) :

$$\sum_{x=\tau}^{\lambda-1} y_x^i + \sum_{x=0}^{(\tau + l_{i,j} - 1) mod \lambda} y_x^j + k_i - k_j \leq \omega_i^j - \lfloor \frac{\tau + l_{i,j} - 1}{\lambda} \rfloor + 1, \forall \tau \in [0,\lambda), \forall (i,j) \in E \tag{8}$$

Cette nouvelle formulation donne une meilleure relaxation de la programmation linéaire car (8) $\Rightarrow$ (6).

3.2 Formulation de Dupont-de-dinechin et al. Formulation directe (FDI) [2]

Cetta formulation est basée sue les variables binaires x_i^t tell que $\sigma_i = \sum_{t=0}^{Tmax} tx_i^t$, où $Tmax$ représente l'horizon de temps. La formulation s'exprimée par :

$$min \sum_{i\in[1,n]} w_i(\sum_{t=0}^{Tmax} tx_i^t) \tag{9}$$

$$\sum_{t=0}^{Tmax} x_i^t = 1 \tag{10}$$

$$\sum_{t=0}^{Tmax} tx_i^t + \theta_i^j - \lambda\omega_i^j \leq \sum_{t=0}^{Tmax} tx_j^t, \forall(i,j) \in E \tag{11}$$

dans ce cas : $\sigma_i = k_i * \lambda + row_i = tx_i^t$ and $\sigma_j = k_j * \lambda + row_j = tx_j^t$

$$\sum_{i=1}^{n} \sum_{k=0}^{\lfloor \frac{Tmax}{\lambda} \rfloor} x_i^{\tau+k\lambda} b_i^s \leq B_s, \forall\tau \in [0, \lambda-1], s \in [1, m) \tag{12}$$

dans ce cas, une nouvelle formulation appelée désagrégée (FDD) est présentée en substituant la contrainte de dépendance pour :

$$\sum_{h=t}^{Tmax} x_i^h + \sum_{h=0}^{t+l_{i,j}-\lambda\omega_i^j-1} x_j^h \leq 1, \forall t \in [0, Tmax], (i,j) \in E \tag{13}$$

Cette nouvelle formulation donne une meilleure relaxation de la programmation linéaire car (13) $\Rightarrow$ (11).

3.3 Formulation de Mingozzi et al pour le RCPSP acyclique. [4]

Cette formulation est basée sur le concept d'ensembles de tâches réalisables pour le RCPSP acyclique. Un sous-ensemble l de tâches en cours d'exécution à l'instant t est réalisable si les contraintes de ressource sont satisfaites et il n'existe pas contraintes de dependance entre $(i) \in l$ et $(j) \in l$, $i \neq j$. Un sequence de sous-ensembles réalisables représente une solution si elle satisfait les conditions :

1. Chaque tâche est exécutée sans interruption.
2. Les date de debut des tâches satisfaitont les contraintes de dependance.

Dans cette formulation, si R_l un sous ensemble réalisables, la variable y_l^t représente une variable binaire $(0-1)$ égale à 1 ssi toutes les tâches du sous-ensemble réalisables R_l sont en cours d'exécution à la date t. La variable x_i^t est une variable binaire $(0-1)$ égale à 1 ssi la tâche i commence à la date t coome dans les modéles précédentes.

4 Extensions de la formulation de Mingozzi pour le RCMSP (ME)

Nous présentons une extension de la formulation directe, soit la une variable binaire $y_l^\tau = 1$, $\tau \in [0, \lambda-1]$ si les tâches de l'ensemble R_l sont en cours d'exécution à l'instant τ de la période générique. Soit R l'ensemble de tous les ensembles réalisables, soit $a_i^l = 1$ si $i \in R_l$ et 0 sinon, la formulation est exprimée par :

$$min \sum_{i=1}^{n} w_i \sum_{t=0}^{Tmax-1} tx_i^t \tag{14}$$

$$\sum_{t=0}^{Tmax-1} x_i^t = 1, i = 1, ..., n \tag{15}$$

$$\sum_{l \in R} (\sum_{\tau=0}^{\lambda-1} a_i^l y_l^\tau) = 1, i = 1, ...n \tag{16}$$

$$\sum_{l \in R} y_l^\tau \leq 1, \tau \in [o, \lambda - 1] \tag{17}$$

$$\sum_{t=0}^{Tmax-1} tx_i^t + \theta_i^j - \lambda \omega_i^j \leq \sum_{t=0}^{Tmax-1} tx_j^t, (i,j) \in E \tag{18}$$

$$\sum_{l \in R} a_i^l y_l^\tau = \sum_{t=0}^{Tmax-1; t mod \lambda = \tau} tx_i^t, 1 = 1, ...n, \tau \in [o, \lambda - 1] \tag{19}$$

Dans cette extension l'ordonnancement se fait sur la période générique λ alors que dans le modèle de Mingozzi l'ordonnancement se fait pour tout l'horizon de temps. Le contrainte (16) impose à la solution d'executer les sous ensembles contenant la tâche i exactement 1 unité de la période générique pour chaque tâche $i = 1, ..., n$. La contrainte (17) assure qu'au plus un sous-ensemble réalisable est en cours d'execution à chaque unité de la période générique. La contrainte (18) est la contrainte de dépendance et la contrainte (19) donne le lien entre x_i^t et y_l^τ. Comme le nombre de variables binaires peut être grand ($\lambda \times R_l$) nous proposons un schéma de génération de colonnes pour résoudre cette relaxation. La contrainte duale liée à la variable y_l^τ, s'écrit :

$$\sum_{i=1}^{n} a_i^l \sigma_i + \beta_\tau + \sum_{i=1}^{n} a_i^l \delta_i^\tau > 0. \tag{20}$$

et le sous-problème de recherche du plus petit coût réduit d'une variable hors base y_l^τ associé à (20) est exprimée par :

$$max \sum_{i=1}^{n} (\sigma_i - \delta_i^\tau) a_i \tag{21}$$

SC

$$\sum_{i=1}^{n} a_i b_i^s \leq B_s, \tau = 0, ..., \lambda - 1, s \in [i, m] \tag{22}$$

$$a_i \in 0, 1 \tag{23}$$

Le sous-problème correspond à un problème de sac à dos multidimensionnel (λ sous-problèmes).

5 Resultats expérimentaux et conclusions

Les instances proviennent du Compilateur ST200 de l'entreprise Stmicroelectronics avec six ressources disponibles.

La table 1 décrit pour 4 instances représentatives, le λ minimum non prouvé irréalisable par la résolution des relaxations ainsi que la borne inférieure du Cmax. On observe que les formulations (FDS) et (FDD) donnent de meilleures relaxations que (FD)et (FDI). On

Instance	lambda	FD	FDS	FDI	FDD
adpcm-st31.1	29	29	29.5	29	29.5
adpcm-st31.2	38	41	42	41	42
gsm-st-231.1	24	32	33	32	33
gsm-st-231.2	36	31.5	31.5	31.5	31.5

TABLE 1 – Borne inférieure pour CMAX (relaxation)

observe aussi une équivalence entre la formulation decomposée et directe qu'il foudra établir théoricament.

La table 2 présente les mémes résultats en nombres entiere avec un temp de résolution limité à 500 s. La formulation décomposée obtient de meilleurs temp de calcul et permet ainsi de résoudre en temps raisonables certains problèmes industriels réputés difficiles.

Instance	lambda	FD	FDS	FDI	FDD
adpcm-st31.1	21	30	30	30	30
adpcm-st31.2	42	42	42	42	NS
gsm-st-231.1	24	33	33	33	33
gsm-st-231.2	26	32	32	32	NS

TABLE 2 – Borne inférieure pour Cmax (entiere 500 seg)

Les expérimentation concernant à la generation de colonnes sont en cours.

6 Les références

[1] A Eichenberger, E. S. Davidson .Efficient Formulation for Optimal Modulo Schedulers. SIGPLAN Conference on Programming Language Design and Implementation - PLDI'97 (1997).

[2] B.Dupont-De-Dinechin. C. Artigues. S. Azem. Resource Constrained Modulo Scheduling. In C. Artigues, S. Demassey and E. Néron. Resource-Constrained Project Scheduling Models, algorithms, extensions and applications. November (2007).

[3]A. Pritsker, L.Watters and P. Wolfe. Multi-project scheduling with limited resources : a zero-one programming approach. Management Science 16, 93-108, (1969).

[4] A. Mingozzi and V. Maniezzo. An Exact Algorithm For The Resource Constrained Project Scheduling Problem Based on a New Mathematical Formulation. Management Science 44, 714-729, (1998).

[5] N. Christofides , R. Alvarez-Valdés and J. Tamarit. Project scheduling with resource constraints : a branch and bound approach. European Journal of Operational Research 29(3), 262-273, (1987).