Historique des travaux autour du probl P||Cmax

Introduction

Prntation du probl.

Parallsme.

Le parallsme est un type d'architecture informatique dans lequel plusieurs processeurs extent ou traitent une application ou un calcul simultannt. IL aide fectuer de grands calculs en divisant la charge de travail entre plusieurs processeurs, qui fonctionnent tous en m temps.

Il existe quatre types de parallsmes, dnis par la taxonomie de Flynn[(1)]. Cette classification est basur deux notions : le flot d'instructions (simple ou multiple), et le flot de donn (simple ou multiples) ; un algorithme est un flot d'instructions ter sur un flot de donn.

Instructions / Donn	Simple	Multiple
Simple	SISD premiers PC machine de Von Neumann Obsol, car tous les PC sont drmais multi-cœur.	SIMD Machines synchrones Pipeline Extion d'une instruction unique sur des donn diffntes.
Multiple	MISD Machines vectoriels Tableau de processeurs Exte plusieurs instructions sur une m donn	MIMD Multi processeurs moire distribu Multi processeurs moire partagmulti-cœur). Multi Ordinateur.

Taxonomie de Flynn

Les premis machines paralls ient des raux d'ordinateurs, et des machines vectorielles (faiblement paralls, troteuses), telles que l'IBM 360, les Cray1. La plupart des machines paralls contemporaines sont drmais MIMD.

On peut dnir une machine parall comme un ensemble de processeurs qui coopnt et communiquent.

BiblioPCmaxLatex-img1.jpg

IBM 360-91 (le plus rapide et le plus puissant en service en 1968) NASA. Centre de vols de Greenbelt (Md)

Ordonnancement

Sur une machine non parall, les tes sont ext sentiellement, les unes apres autres. Certaines tes, ou jobs peuvent demander plus de temps que d'autres pour e entiment trait.

Lorsque plusieurs ressources (processeurs, machines, cœurs) sont disponibles, ou que des jobs a exter ne sont pas indndants (m traitur un seul processeur), se pose alors, un probl d'ordonnancement.

Celui-ci consiste ganiser, dans le temps, les jobs ter, en les affectant e ressource donn de mani tisfaire un certain nombre de contraintes, tout en optimisant un ou des objectifs.

L'ordonnancement, fait partie de la catrie des probls d'optimisation combinatoire.

Les probls qui s'y rattachent sont trari

Premiment, la nature des machines paralls doit e conside. Celles-ci peuvent e identiques (Le m temps de traitement sera nssaire, d'une machine autre) ; uniformes (un quotient de vitesse qi propre e machine est pliquer pour chaque te affecttte machine pour drminer le temps de traitement nssaire) ; ou indndantes (les temps de traitements des tes sont ni uniformes ni proportionnels d'une machine autre).

Ensuite, des contraintes peuvent affecter les jobs eux-ms. Dans le cas d'un probl prptif, les taches peuvent e interrompues, et reprises ulteurement. Il est possible que les jobs soient indndants, ou au contraire, e li par des relations de prdence. Ces jobs ne sont disponibles qu'rtir d'une certaine date. Ou encore, e de durgale, ou tous de duriffnte.

Pour finir, l'objectif de l'ordonnancement est d'optimiser un crit. Par exemple, minimiser la somme des dates de fin, la somme des retards, le nombre de tes en retard, ou simplement, le retard total. Mais le plus habituel, est de chercher nimiser le temps total de traitement de tous les jobs, i.e minimiser le makespan.

Enonc $\mathbf{P||C_{max}}$

Ces diverses possibilitnissent divers probls d'ordonnancements diffnts, recenst classifipar Graham et al. [1], qui introduit la notation trois-champs $\alpha|\beta|\gamma$.

Le probl $\mathrm{P_m||C_{max}}$ se dnit alors ainsi :

- $\alpha = \alpha 1\ \alpha 2$, drmine l'environnement machines.

α = P : Les machines sont paralls et identiques : Un job, une te prendra le m temps de traitement qu'il soit extr une machine ou une autre. Le nombre de machine (m) est variable.

- β c { $\beta 1$, $\beta 2$, $\beta 3$, $\beta 4$, $\beta 5$, $\beta 6$}, drmine les caractstiques des jobs, ou des tes.

β est vide. Ce qui signifie que la prption n'est pas autorisles jobs doivent e ext'une traite, sans interruption ni coupure) et qu'il n'y a pas de relation entre les jobs (ils sont indndants).

- γ drmine le crit timiser.

$\gamma = \mathrm{C_{max}}$: on cherche timiser le makespan, i.e le temps de traitement total.

$\mathrm{P_m||C_{max}}$ consiste anifier un ensemble J = {1,2,. . .,n) de n jobs simultan pour e traitar m machines identiques et paralls. Chaque job, qui requi une option, peut e traitr une des m machines. Le temps de traitement de chaque job ($\mathrm{P_i}$ avec i $\in$ N) est connu avance. Un job commenct compl sans interruption. Les jobs, indndants, sont extar une seule machine, et une machine ne peut traiter qu'un seul job fois.

Probltique

Comme l'ont dntrrey et Johnson, $\mathrm{P_2||C_{max}}$ est un probl NP-Difficile [4], et $\mathrm{P||C_{max}}$ est un probl NP-Difficile au sens fort [3]. Cependant, $\mathrm{P_m||C_{max}}$ devient un probl NP-Difficile, du moment que le nombre de machines est fix], comme l'a montrthkopf [5], qui a prnt algorithme de programmation dynamique.

Donner la solution optimale probl d'ordonnancement (dans notre cas $\mathrm{P_m||C_{max}}$ n'est pas riste. M pour un probl de taille modeste, la rlution de celui-ci demanderait un temps excessif et donc ribitoire.

La rlution du probl d'ordonnancement va reposer sur des modes d'approche, qui consistent lculer en temps polynomial, une solution assez proche de la valeur optimale.

Dans la littture, l'de d'ordonnancement est triche et abondante. Le but nt d'amorer le temps de calcul, et d'approcher le rltat optimal.

Rudre le probl

//****************************

// Phrase choc d'introduction

Comme qudemment, l'existence d'une solution qui rut le probl n'est // // pas pensable (ins que P=NP ...). De fait, les solutions propos en temps polynomial // sont approch. Il est impossible de prnter une liste exhaustive des solutions propos tant la recherche dans ce domaine est soutenue. .

Il existe plein de solutions blabla LS, ... Reseaux de neurones, genetique...

Petit plan

Ici on presente les methodes les plus utilis, et quelques algo par methode.

Aprn rappel des notations utilis, methode basur LP, des heuristiques bas sur LS, Bin pacqking, pour finir avec un PTAS, et un partitionnement.

/****************************

Notations utilis

Chaque document utilise sa propre notation, mais les notions sont les ms.

Soient les donn du probl Pm||Cmax,

Un ensemble de n jobs J = {1,2,...,n)

Chaque job j a un temps de traitement pj P= $\{p_1, p_2, \ldots, p_n\}$

m machines paralls identique Mi avec (i=1,2,...,m)

C_m^A (J) : le rltat de l'ordonnancement d'un ensemble J de jobs, sur m machines paralls, identiques, obtenu par un algorithme A.

$C_m^?$ (J) : le makespan optimal.

r(A) = C_m^A (J) / $C_m^?$ le ratio d'approximation atteint par l'algorithme A au pire cas.

Heuristiques

Les heuristiques prntent plusieurs avantages. Leur complexitt rite, et obtiennent de bonnes performances. Elles reprntent la plus grande partie des recherches concernant le probl d'ordonnancement, m si leurs performances, au pire cas, n'est pas garantie.

Sont abord ici les heuristiques les plus prntes dans la littture.

Bas (List scheduling)

L'id'une LS est de stocker l'ensemble des jobs dans celle-ci, les trier dans un ordre particulier, avant de les affecter e machine selon des res dnies.

Algorithme LPT rule graham 1969 Graham propose [8] Longest Processing Time (LPT) rule.

____

Algorithme LPT

____

Input : instance de Pm||Cmax, avec m machines, n jobs et leur temps d'extion

1 : trie les jobs de l'ensemble J dans l'ordre doissant de leur temps d'extion et rdexe l'ensemble, de telle mani tenir :

$P_1 >= P_2 >= \ldots >= P_n$

2 : Parcours la liste, et affecte chaque job machine la moins charg, moment-l

____

Exemple.
Soit P = (13,10,7,6,6,5,3,2) l'ensemble des Pj d trians l'ordre doissant
pliquer sur 4 machines paralls identiques.

[Warning : Image ignored]

Nous obtenons C_4^{lpt} (J) = 14

Le tri puis l'affectation s'effectuent en **O(n log n + n log m)**

Le ratio d'approximation **r(LPT)** $<=$ $\frac{4}{3}$**-** $\frac{1}{3*m}$

Algorithme LPT-REV (Croce et Scatamacchia, 2018) Ce ratio d'approximation obtenu par LPT (r(LPT) = $\frac{4}{3}$- $\frac{1}{3*m}$) est une borne supeure que LPT peut atteindre, mais qu'il ne dssera jamais. Chaque utilisation de LPT donnera un rltat qui oscillera entre 1 et = $\frac{4}{3}$- $\frac{1}{3*m}$.

Exemple de pire cas.
Soit P = (7,7,6,6,5,5,4,4,4) l'ensemble des Pj d trians l'ordre doissant
pliquer sur 4 machines paralls identiques.

- LPT donne le rltat suivant :

[Warning : Image ignored]

C_4^{lpt} (J) = 15

- Un bon ordonnancement aurait donnitemize [Warning : Image ignored]

$C_4^{?}$ (J) = 12

o Soit une marge d'erreur de 15/12
Pour m = 4
$\frac{4}{3}$- $\frac{1}{3*m}$= $\frac{16}{12}$- $\frac{1}{12}$ = $\frac{15}{12}$
Ce cas reprnte un pire cas pour LPT

Croce et Scatamacchia [9] en examinant le comportement de LPT rule , notamment au niveau du ratio d'approximation, constatent que celui-ci peut e rit selon certaines configurations, ou instances du probl, et rgent le th suivant :

LPT a un rapport d'approximation non supeur 4_{3}- $\frac{1}{3*(m-1)}$ pour m $\geq$3 et n $<>$ 2m + 1.
LPT atteint la limite de Graham de $\frac{4}{3}$- $\frac{1}{3*m}$ pour m $\geq$2 et uniquement dans les cas o n = 2m + 1, et la machine critique traite 3 jobs, tandis que les autres en traitent 2.
La machine critique est la machine qui exte le job critique
Le job critique (not) est le job qui drmine le makespan.

Le rapport d'approximation augmente moins vite (en fonction du nombre de machines) que le prdent ndition d'ter certaines instances.

NB
L'exemple prdant (pire cas) a les caractstiques suivantes :
Nombre de jobs n = 2m +1
La machine critique exte 3 jobs
Les autres extent 2 jobs
Un rapport d'approximation de $\frac{4}{3}$- $\frac{1}{3*m}$

Une modification algorithme LPT est apportour placer le probl $P_m||C_{max}$ toujours dans une instance o le ratio d'approximation est $\frac{4}{3}$- $\frac{1}{3*(m-1)}$. Cette modification consiste anifier en premier, le job critique sur une machine M1.

Algorithm *LPT-REV*

Input: $P_m||C_{max}$ instance with n jobs and m machines.

1: Apply LPT yielding a schedule with makespan z_1 and $k-1$ jobs on the critical machine before job j'.
2: Apply $LPT' = LPT(\{j'\})$ with solution value z_2.
3: **If** $m = 2$ **then** apply $LPT'' = LPT(\{(j'-k+1), \ldots, j'\})$ with solution value z_3 and return $\min\{z_1, z_2, z_3\}$.
4: **Else** return $\min\{z_1, z_2\}$.

Le ratio d'approximation **r(LPT-REV)** $<=$ $\frac{4}{3}$**-** $\frac{1}{3*(m-1)}$

Basn-Packing

Le probl Bin-packing, est semblable au probl $P_m||C_{max}$. Il consiste nger des objets dans des bacs de taille similaires, tout en minimisant le nombre de boites.

L'ensemble des n jobs J = {1,2,. . .,n) et de leur temps de traitement pj P= $\{p_1, p_2, \ldots, p_n\}$ peuvent e vus respectivement, comme :
un ensemble d'objets $\mathbf{T = \{T_1, T_2, \ldots, T_n\}}$,
et leur taille **L(Ti)**
Une taille maximale **C** des bacs (ou bos) est donn

Un packing, est une partition $P<P_1, P_2, \ldots, Pm>$ de T telle que $L(P_j) <= C$ (avec $1<=j<=m$)
Le but est de placer les T_i dans des bacs P_j de taille C, de mani nimiser le nombre de bacs m.

L'idst d'utiliser le probl envers, pour approcher une solution au probl d'ordonnancement.

Algorithme MULTIFIT Coffman, Garey, et Johnson [10] se sont intss l'algorithme FFD (First Fit Decreasing), un outil de rlution du probl bin-packing , pour l'adapter au probl d'ordonnancement.
FFD(T,C) renvoie le nombre de bacs de taille C non vides nssaires, et l'arrangement correspondant de l'ensemble T d'objets.

Soit $T_m^?$ = min{C :FFD(T,C) <= m} la plus petite valeur de C (taille des bacs) qui permet 'e pacquns m (ou moins) bacs.

Le but de Multifit, est donc, de rire la valeur de C, faire tourner FFD(T,C), jusqu' que le nombre m de bacs alors, devenu insuffisant, augmente 1.
Cette valeur charni de C est $T_m^?$, qui correspond au makespan minimum recherche l'ordonnancement de l'ensemble T de jobs sur m machines paralls identiques.

[Warning : Image ignored]
Fonctionnement de FFD et principe de MULTIFIT

La recherche de $T_m^?$ s'effectue par dichotomie.
La borne supeure Cu[T,m] = max{(2/m)*L(T), $\max_i${L(Ti)}}
La borne infeure Cl[T,m] = max{(1/m)* L(T), $\max_i${L(Ti)}}

Multifit ret les parames suivants
T, un ensemble de jobs
m, un nombre de processeurs
k, un nombre d'ittions maximal (pour la recherche dichotomique)

Apr ittions, Multifit renvoie Cu(k) qui correspond plus petite valeur C pour laquelle FFD[T,C] <= m

Le tri puis k FFD s'effectuent en **O(n log n + kn log m)**
Ratio [11] **r(MF) <= 1.220 + 2^{-k}**

Gralement, Multifit donne un rltat tratisfaisant avec k=7.

Algorithme COMBINE Lee et Massey [11] ont l'id'utiliser LPT pour rire les bornes de drt de Multifit dans un algorithme nommMBINE.

Soient la moyenne des poids des jobs par processeur A = $\sum_{i=1}^{n} \frac{Pi}{m}$
M = C_m^{lpt} (J)
Et M* = $C_m^?$ (J)

Si M >= 1.5 A alors M* = M

Algorithme COMBINE

Input : instance de Pm||Cmax, avec m machines, n jobs et un coefficient d'arrα (0.005)
1 : A = $\sum_{i=1}^{n} \frac{Pi}{m}$
M [F0DF ?] C_m^{lpt} (J)
Si M >= 1.5 A alors M* = M
Sinon aller en 2 :
2 : Appliquer Multifit avec
Cu = M
Cl = max{ (M / ($\frac{4}{3}$- $\frac{1}{3*m}$)), P1, A} (P1 : job le plus long)
Arrr lorsque Cu − Cl <= αA

Complexit **O(nlogn + knLogm)**
Ratio [12] **r(CB) <= 13/11 + 2^{-k}**

Avec k le nombre d'ittions de recherches dichotomiques.
Concernant la complexit
Gralement les ittions Combine k = 6 (lorsque Cu − Cl <= αA), mais il a d exte fois LPT, donc, gralement, k = 7.

Input: n, m, p_i for $i = 1, \ldots, n$.

Step 1. Let $r = 1$, $q = 1$, and $C_{\max} = C_{\max}(LPT)$, the makespan obtained by the LPT algorithm. Go to step 2.

Step 2. Let $\phi = \emptyset$, $A = \{1, \ldots, n\}$; and $B = \emptyset$. Let ω_r be the sequence of jobs in job-list A ordered according to ordering r and go to step 3.

Step 3. Let $\alpha = C_{\max}(M)$ be the makespan obtained by using algorithm MULTIFIT with $\sigma = \phi_q \omega_r$ in step 1 of algorithm M. If $C_{\max} > \alpha$, set $C_{\max} = \alpha$, $\gamma_h = \pi_h$ for $h = 1, 2, \ldots, m$. If $A \neq \emptyset$, go to step 4; otherwise go to step 5.

Step 4. Remove the last job of ω_r and place it into B. Update A, ϕ_q and ω_r. Let $\sigma = \phi_q \omega_r$ and return to step 3.

Step 5. If $r < 2$, set $r = r + 1$ and return to step 2; otherwise, go to step 6.

Step 6. If $q < 2$, set $q = q + 1$, $r = 1$, and return to step 2; otherwise stop. The schedule where jobs in γ_h are processed on machine h is an approximate solution of the $P||C_{\max}$ problem with makespan $C_{\max}$.

Algorithme LISTFIT Gupta et Ruiz-Torres [12], ont aussi l'id'utiliser Multifit, afin de riser l'algorithme Listfit.
Celui-ci sre la liste des travaux en 2 sous-listes, trait soit dans un ordre LPT (longest Time Processing), soit dans un ordre SPT (Shortest Time Processing). Puis, Listfit combine ces deux sous-listes en appliquant MultiFit aque ittion.

Algorithme Listfit

Complexit **$O(n^2 \log n + kn^2 b \log m)$**
Ratio **$r(MF) <= 13/11 + 2^{-k}$**

O k est le nombre d'ittions pour Multifit.

Aproche gloutonne

Algorithme SLACK (Croce et Scatamacchia, 2018) Croce et Scatamacchia [9], en effectuant la preuve d'une borne d'approximation pour le dloppement de LPT-Rev, ont mis en dence l'importance des diffnces de temps entre les jobs, ainsi que le regroupement de ceux-ci en sous-ensembles.
Notamment, pour l'instance suivante

Nombre de jobs n = 2m+1
Avec P_{2m+1} >= P_1 - P_m.
O ils ont planifi job 2m+1, puis le sous ensemble tri ... m}, puis le sous ensemble tri+1 ... 2m}.

En rlte la strate gloutonne suivante

Algorithme SLACK

---

1 : Trier la liste des jobs dans l'ordre doissant des temps nssaires de traitement
2 : Rdexer les jobs, de mani tenir P_1 >= P_2 >= ... >= P_n
3 : Duper l'ensemble en n/m tuples de m Jobs (ajout de jobs dummy de taille nulle pour le dernier tuple, si n n'est pas un multiple de m).
4 : Considr chaque tuple avec la diffnce de temps entre le premier job du tuple, et le dernier, appellack.
{ { 1, ... , m} {m+1 , ... , 2m} ... }
P1 – Pm Pm+1 – P2m ...
5 : Trier les tuples par ordre doissant de Slack, et ainsi former un nouvel ensemble
{{m+1 , ... , 2m} { 1, ... , m}} (si Pm+1 – P2m > P1 – Pm)
6 : Appliquer L'ordonnancement (Affectation machine la moins charg ce moment-l l'ensemble ainsi obtenu

---

Programmation Linre

L'ordonnancement, et plus particuliment $P_m||C_{max}$ s'inscrit parfaitement dans l'ncun probl de programmation linre. En effet la fonction objectif, minimiser le makespan, ainsi que les contraintes sont des fonctions linres.
Toutefois, les variables, et le rltat attendus sont discrets, ce qui rend la rlution du probl nettement plus difficile comparne programmation linre riables continues.
Ces algorithmes donnent une solution faisable, exacte.

Algorithme PA de Mokotoff. Makatoff [6] prnte un algorithme basr la formulation de la programmation linre, en utilisant des variables boolnes d'affectation des jobs e machine.

La minimisation du makespan peut e posinsi :

Min y tel que

$\sum_{j=1}^{m} xij = 1$ pour 1<= i <= n
Sur toutes les machines, au moins un et un seul x_i est 1
Un job est affectne et une seule machine.

Y - $\sum_{i=1}^{n} Pixij \geq 0$ pour 1<= j <= m
Pour une machine donn la somme des temps <= y

O la valeur optimale de y est le Cmax
et X_{ij} = 1 si le job i est affecta machine j
= 0 si le job i n'est pas affecta machine j

Le programme linre est donc compos
n*m + 1 variables (les variables x_{ij} et la variable y)
n+m contraintes

La zone F peut e dnie ainsi :

F = { (x,y) : x ∈ B^{n*m}, y $\in R_+$: $\sum_{j=1}^{m} xij = 1 \quad \forall i$; Y - $\sum_{i=1}^{n} Pixij \geq 0, \forall j\}$

avec B = $\begin{bmatrix} x11 & \cdots & xn1 \\ \vdots & \ddots & \vdots \\ x1m & \cdots & xnm \end{bmatrix}$

Le polytope P, relatif st dni ainsi
P = { (x,y) : x ∈ R_+^{n*m}, y $\in R_+$: $\sum_{j=1}^{m} xij = 1 \quad \forall i$; Y - $\sum_{i=1}^{n} Pixij \geq 0, \forall j\}$

Il est possible de construire un ensemble fini d'**inlit**

Ax + Dy <= $\bar{b}$ telles que min{y : (x,y) ∈ F} = min{y : x ∈ R_+^{n*m} , y ∈ R_+, Ax+Dy <= $\bar{b}$ }
NB : Une solution (x,y) ∈ P doit e exclue (n'est pas un vecteur entier) si (x,y) ∉ F

Des **Inlitransitoires** peuvent e gr (nombre maxi de job par machine)
$\sum_{i \in Sj}^{\square} xij$ <= Lj (Lj = h-1 [F0F3 ?] S_{jh} > Lb et $S_{j(h-1)}$ <= Lb)
Lb : Borne infeure

Pour un probl Pm||Cmax, m de taille modeste, le nombre de variables, contraintes, et trmportant, dont certaines sont inutiles. L'algorithme va donc utiliser la mode des plans snts (Cutting Planes Technique). A chaque ittion, des inlitalides sont gr, puis une relaxation est ext Jusqu'obtention d'une solution faisable.

____
Algorithme PA
____
Drmination de la borne infeure (Lb) suivant l'algorithme de McNaughton [7].
Drmination de la borne supeure Ub (juste pour la nommer) suivant l'heuristique LPT.
Si Lb coide avec Ub, alors la solution optimale est trouv

Sinon, le processus ittif drre :
Dans chaque ittion, un programme de relaxation linre est rlu dans lequel Cmax doit e l borne infeure actuelle Lb.
Si la solution obtenue est enti, donc faisable, l'algorithme s'arr et la solution actuelle est optimale.
Sinon, des nouvelles inlitinlitt/ou inlitransitoires), sont ajout relaxation linre. Le nouveau programme linre est rlu et l'algorithme s'arr si la solution est enti.
Si la relaxation n'est pas possible, la limite infeure Lb est augment'une unit le processus, redrre.
Par contre, Si les inlite peuvent pas e gr, un algorithme Branch&Bound prend le relais pour rudre le probl.

Approximation

Une catrie d'algorithmes fournit une garantie d'approche. C'est le cas, notamment, des PTAS (Sch d'Approximation en Temps Polynomial).

Principe PTAS (programmation dynamique, approximation ε . . .

- o ***Algorithme Using dual approximation Algorithm for Scheduling problems :*** Theorical and Practical Results (Hochbaum et Shmoys 1987)

Autres approches

LDM Transition
LPT fait toujours rrence pour comparer chaque algorithme dlopp
..
Les heuristiques, font plus l'objet de recherches . . ..

Synth

Rpitulatif complexitorne d'approximation. . . thiques
Quelques rltats comparxpmentaux
Avantages inconvents

Conclusion

Utilisation des algorithmes Slack est utilisns . . .

1.1.1. Point de vue personnel

1. Recherche documentaire

Document LDM n'explique pas comment les partitions sont construites (semble utiliser LPT)
Difficult trouver des applications aux algorithmes
Beaucoup de documents indisponibles, payant. . .
Acteurs prolifiques Graham, Mokotoff, . . .

1.1.1. Futur ? . . .

Remarques
(1). A trouvssi sous le nom de taxonomie de Tanenbaum

Un site effectue rliment, un top 500 des machines les plus puissantes :
https://www.top500.org/

Des sites internet rassemblent des documents concernant les probls d'ordonnancement :
http://www.mathematik.uni-osnabrueck.de/research/OR/class/
http://schedulingzoo.lip6.fr/

Rrences

[1]
Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979).Optimization and approximation in deterministic sequencing and scheduling : a survey. In P. L. Hammer, E. L. Johnson, & B. H. Korte (Eds.), Discrete optimization II, annals of discrete mathematics (Vol. 5, pp. 287–326).

[2]
Chen B.,Potts C.N.,and Woeginger G.J.(1999).Areviewofmachine scheduling : Complexity, algorithms and approximability. In D. Z. Du&P.M.Pardalos(Eds.),Handbookofcombinatorialoptimization : Volume 1–3. New York : Springer.

[3]*
M.R. Garey and D.S. Johnson, Computers and Intractability : A Guide to the TheoryofNP-Completeness,Freeman,SanFrancisco,1979.

[4]*
M.R. Garey and D.S. Johnson, Strong NP-completeness results : motivation, examples and implications, Journal of the Association for Computing Machinery 25 (1978), 499-508.

[5]*
M.H. Rothkopf, Scheduling independent tasks on parallel processors, Management Science 12 (1966), 437-447.

[6]
E. Mokoto[FB00 ?], Scheduling to minimize the makespan on identical parallel machines : An LP-based algorithm, Investigacioon Operativa 8(1999)97–108.

[7]
McNaughton, R., "Scheduling with deadlines and loss function", Management Science 6, 1959, 1-12.

[8]
R.L. Graham, Bounds for certain multiprocessing anomalies, Bell System TechnicalJournal 45 (1966), 1563-1581.

[9]
F. Della Croce and R. Scatamacchia, "The Longest Processing Time rule for identical parallel machines revisited," Journal of Scheduling, 2018.

[10]
Coffman E.G Jr., Garey M. R., & Johnson, D. S. (1978). An application of bin-packing to multiprocessor scheduling. SIAM Journal on Computing, 7, 1–17.

[11]
Lee, C. Y., & Massey, J. D. (1988). Multiprocessor scheduling : Combining LPT and MULTIFIT. Discrete Applied Mathematics, 20(3), 233–242.

[12]
Gupta J. N. D., & Ruiz-Torres, A. J. (2001). A list[FB01 ?]t heuristic for minimizing makespan on identical parallel machines. Production Planning & Control, 12(1), 28–36.