ESCALONAMENTO EM MÁQUINAS PARALELAS PARA MINIMIZAR O ATRASO PONDERADO DE TAREFAS DE TEMPOS IGUAIS

Mitre Costa Dourado[1] Rosiane de Freitas Rodrigues [2]
Jayme Luiz Szwarcfiter[3]

[1]Instituto de Matemática e NCE - UFRJ
[2]Departamento de Ciência da Computação, DCC - UFAM
[3]Instituto de Matemática, NCE, COPPE - UFRJ
Universidade Federal do Rio de Janeiro-UFRJ, RJ, Brasil
mitre@nce.ufrj.br, rosiane@dcc.ufam.edu.br, jayme@nce.ufrj.br.

Resumo

Este trabalho aborda problemas de escalonamento de tarefas em máquinas paralelas idênticas e uniformes, tarefas de tempos de processamento iguais e a função de minimização do atraso ponderado. São apresentados algoritmos para os problemas $P|p_j = p|\sum w_jT_j$ e $Q|p_j = p|\sum w_jT_j$, com n tarefas independentes e m máquinas paralelas idênticas e uniformes, respectivamente. Os problemas envolvem tarefas com tempos de processamento iguais (unitários ou não) e diferentes prioridades representadas por pesos distintos. O objetivo consiste em determinar escalonamentos ótimos que minimizem o atraso das tarefas tardias ponderadas. A melhor solução vigente para estes problemas, envolve a modelagem dos mesmos como um problema de alocação (*assignment problem*), resultando em um algoritmo de complexidade $O(n^3)$. Este trabalho apresenta algoritmos que resolvem tais problemas de maneira mais eficiente, em tempo $O(n^2)$ tanto para máquinas paralelas idênticas quanto paralelas uniformes.

Palavras-chave: atraso ponderado, escalonamento teórico, máquinas paralelas, tarefas de tempos iguais.
Área Principal: Otimização Combinatória.

Abstract

This work tackles problems of scheduling jobs on identical and uniform parallel machines, equal-time jobs and the minimization of the weighted tardiness. We present algorithms for $P|p_j = p|\sum w_jT_j$ and $Q|p_j = p|\sum w_jT_j$ problems, with n independent jobs and m identical or uniform parallel machines, respectively. The problems involve jobs with equal processing time (unitary or not) and different priorities represented by different weights. The objective is to determine optimal schedules that minimize the weighted tardiness of jobs. The best current solution to these problems involves the reduction of them as an assingment problem, resulting in an algorithm of complexity $O(n^3)$. This paper presents algorithms that solve such problems more efficiently in $O(n^2)$ time complexity, both for identical or uniform parallel machines.

Keywords: equal-time jobs, parallel machines, scheduling theory, weighted tardiness.
Main Area: Combinatorial Optimization.

1 Introdução

Escalonamento de tarefas em máquinas paralelas que envolvem a minimização do atraso ponderado, constitui uma das classes de problemas de escalonamento mais estudadas e com função de mais difícil tratamento dentre as funções objetivo regulares clássicas (Brucker (2004), Pinedo (2002)), que são aquelas baseadas no tempo de completude das tarefas, sempre monotonicamente crescente. Tal classe é abordada neste trabalho onde os problemas estudados envolvem o escalonamento de tarefas em um conjunto de máquinas em paralelo, possuindo características idênticas (máquinas paralelas idênticas) ou características distintas expressas em termos de velocidades diferentes de execução (máquinas paralelas uniformes). As tarefas possuem tempos de processamento iguais (unitários ou não) e prioridades distintas, estando disponíveis para serem processadas a qualquer momento, respeitando-se apenas as datas estipuladas para que a execução de cada uma termine. E, deseja-se minimizar o atraso das tarefas tardias ponderadas.

Neste contexto, dois grandes problemas são abordados, cujas representações em notação clássica de 3-campos (Graham et al. (1979)) são: $P|p_j = p|\sum w_j T_j$ e $Q|p_j = p|\sum w_j T_j$. Suas formulações são dadas como segue. O problema $P|p_j = p|\sum w_j T_j$, indica que são dadas n tarefas $J = \{J_1, ..., J_n\}$ a serem escalonadas em m máquinas paralelas idênticas $P = \{P_1, ..., P_m\}$, onde as máquinas possuem as mesmas características e as tarefas possuem tempos iguais de processamento $p_j = p$. O problema $Q|p_j = p|\sum w_j T_j$, indica que são dadas n tarefas $J = \{J_1, ..., J_n\}$ a serem escalonadas em m máquinas paralelas uniformes $Q = \{Q_1, ..., Q_m\}$, onde cada máquina Q_i possui uma velocidade específica q_i e cada tarefa J_j possui um tempo de processamento $p_j = p$. A execução da tarefa J_j na máquina Q_i requer $\frac{p_j}{q_j}$ unidades de tempo. Um caso especial é quando a velocidade para qualquer máquina Q_i é a mesma ($q_i = q$), onde tem-se o caso anterior envolvendo máquinas paralelas idênticas (P).

Para ambos os problemas descritos acima, a melhor solução encontrada na literatura se baseia na redução dos mesmos ao problema clássico de alocação (***assigment problem***), resolvendo-os em $O(n^3)$ (Brucker e Knust (2010), Durr(2010), Pinedo (2002)). Neste trabalho, um algoritmo mais eficiente está sendo proposto, de tal forma a resolver os problemas em complexidade de tempo de $O(n^2)$.

O restante deste artigo está estruturado da seguinte maneira. Na Seção 2, são apresentados os principais resultados para problemas de escalonamento em máquinas paralelas idênticas e uniformes. Na Seção 3, são definidos os problemas e apresentados os conceitos necessários. Na Seção 4 é apresentado um algoritmo geral para os problemas abordados. Por fim, considerações finais são feitas na Seção 5.

2 Resultados Relacionados

Esta seção apresenta os principais resultados da literatura sobre problemas de escalonamento envolvendo **máquinas paralelas uniformes**. a função objetivo de minimização de tarefas tardias, com tempos de processamento quaisquer ou os casos de tempos de processamento

iguais e, principalmente, tempos de processamento estritamente unitários.

Para **tempos de processamento arbitrários**, considerando funções objetivo regulares clássicas e escalonamento sem preempção, o único problema envolvendo máquinas paralelas uniformes que possui solução polinomial é o problema $Q||C_j$, que pode ser resolvido em tempo $O(mn^3)$ por técnicas de fluxo em redes. Na verdade, o caso mais geral envolvendo máquinas paralelas não-relacionadas, $R||C_j$, também é resolvido da mesma forma (Bruno et al (1978)). Os problemas $Q||C_{max}$ e $Q||w_jC_j$ são NP-difíceis, uma vez que os casos mais simples, envolvendo máquinas paralelas idênticas, $P||C_{max}$ e $P||w_jC_j$ respectivamente, são NP-difíceis (cf. Brucker (2002)). Isto significa que os problemas de escalonamento $Q||L_{max}$, $Q||U_j$, $Q||w_jU_j$, $Q||T_j$ e $Q||w_jT_j$, que envolvem outras funções regulares clássicas, são também NP-difíceis. Em escalonamentos onde se permite preempções (onde as tarefas podem ter ter suas execuções interrompidas para serem retomadas em um tempo futuro), o problema $Q|pmtn|C_{max}$ pode ser resolvido em tempo $O(n + m\log n)$ (Gonzalez&Sahni (1978)), e os problemas $Q|pmtn; r_j|C_{max}$, $Q|pmtn|C_j$ e $Q|pmtn|L_{max}$ podem ser resolvidos em tempo $O(n\log n + mn)$ (Labetoulle et al. (1984) e Dessouky et al. (1990), cf. Brucker (2002)).

Para **tempos de processamento iguais**,existem dois principais resultados envolvendo funções regulares do tipo $maxf_j(C_j)$ e $\sum f_j(C_j)$, que mostram que os problemas $Q|p_j = 1|maxf_j(C_j)$ e $Q|p_j = 1|\sum f_j(C_j)$ podem ser resolvidos em tempo polinomial (Brucker (2004), Leung et al. (2004), Durr (2010)). Para o problema $Q|p_j = 1|maxf_j(C_j)$,uma solução ótima pode ser encontrada através adaptando-se o algoritmo de Lawler em tempo $O(n^2)$. Para o problema $Q|p_j = 1|\sum f_j(C_j)$, uma solução pode ser encontrada resolvendo-o como um problema de alocação, em tempo $O(n^3)$. Alguns casos especiais podem ser resolvidos de maneira mais eficiente, como os problemas $Q|p_j = 1|\sum w_jC_j$ e $Q|p_j = 1|\sum T_j$ com tarefas sem pesos, bem como $Q|p_j = 1|L_{max}$ e $Q|r_j; p_j = 1|C_{max}$, todos estes problemas sendo resolvidos em tempo $O(n\log n)$. Vale ressaltar que para os problemas $Q|p_j = 1|\sum w_jU_j$ e $Q|p_j = p|\sum w_jU_j$, envolvendo tarefas ponderadas e a função de minimização da penalidade unitária, um algoritmo recente foi proposto pelos autores (Dourado et al, 2009) de complexidade de tempo $O(n\log n)$.

Neste trabalho são estendidos os conceitos e abordagens utilizados na resolução de problemas similares, de tal forma a ser proposto um algoritmo quadrático para os problemas $P|p_j = p|\sum w_jT_j$ e $Q|p_j = p|\sum w_jT_j$, com tarefas ponderadas.

3 Os Problemas $P|p_j = p|\sum w_jT_j$ e $Q|p_j = p|\sum w_jT_j$

O problema $Q|p_j = p|\sum w_jT_j$ é uma generalização do problema $P|p_j = p|\sum w_jT_j$. Sendo assim, a partir deste momento será considerado apenas o caso mais geral que envolve máquinas paralelas uniformes. Da mesma forma, o problema $Q|p_j = p|\sum w_jT_j$ é equivalente ao problema $Q|p_j = 1|\sum w_jT_j$ (Brucher e Kravchenko (2006)), onde os tempos de processamento p_j e as datas de término d_j podem ser escalados pelo fator $\frac{1}{p}$. Sendo assim, a partir deste ponto será considerado apenas o problema $Q|p_j = 1|\sum w_jT_j$.

O problema $Q|p_j = 1|\sum w_jT_j$, em notação de 3-campos (Graham et al. (1979)), pode ser

definido como segue. Dado um conjunto de n tarefas $J = \{J_1, ..., J_n\}$, todas disponíveis ao mesmo tempo para execução ($r_j = 0$), mas cada uma tendo datas de término não-obrigatórias d_j (do inglês *due date*, requerendo um tempo de execução unitário (*unit execution time*) $p_j = 1$ e tendo prioridades distintas representadas por pesos w_j. Tais tarefas devem ser escalonadas em m, $1 \leq m \leq n$, máquinas paralelas uniformes $Q = \{Q_1, ..., Q_m\}$, ou seja, cada máquina Q_i possuindo uma capacidade q_i distinta de processamento (velocidades diferentes). A execução da tarefa J_j na máquina Q_i requer $\frac{p_j}{q_i}$, onde para tarefas de tempos unitários tem-se $\frac{1}{q_i}$.

Assume-se que para cada tarefa, suas datas término e seus pesos são inteiros não-negativos. Cada máquina pode processar no máximo uma tarefa e cada tarefa pode ser processada em uma única máquina. Conseqüentemente, um escalonamento é uma função injetiva $\mathcal{S}$, a qual associa para cada tarefa $J_j \in J$ um número s_j, chamado tempo de início da tarefa J_j, e uma máquina específica $Q_y \in Q$. Dizemos que J_j tem sido escalonado para iniciar no tempo s_j na máquina Q_y. Considerando que a tarefa J_j seja escalonada na k-ésima posição da máquina Q_i, denota-se por $C_j = \frac{k}{q_i}$, o tempo de completude (*completion time*) desta tarefa J_j. Uma tarefa factível no tempo t, nestes casos, é uma tarefa que satisfaz $0 < C_j \leq d_j$. Um escalonamento $\mathcal{S}$ é factível quando cada tarefa J_j é factível para seus tempos de completude C_j. Para cada tarefa J_j é computado o seu atraso ponderado, dado por $T_j = max\{0, C_j - d_j\}$, e o objetivo consiste em minimizar a soma total do atraso ponderado das tarefas, função objetivo $\sum w_j T_j$.

Para a versão não-ponderada do problema, $Q|p_j = 1|\sum T_j$, uma estratégia simples resolve o problema, tendo como base o Lema 1 apresentado a seguir.

Lema 1 (cf. Pinedo (2002)). *Se em um escalonamento para (J, Q) existirem duas tarefas J_a e J_b tal que $d_a \leq d_b$ e $p_a \leq p_b$, então existe uma sequência ótima na qual a tarefa J_a aparece antes da tarefa J_b.*

Este tipo de resultado é conhecido como critério de eliminação ou dominância, muito útil para o desenvolvimento de métodos exatos para as versões NP-difíceis do problema, geralmente envolvendo tempos de processamento de valores arbitrários. O foco deste trabalho está em tarefas de tempos de processamento iguais, cujo problema geral, $Q|p_j = p|\sum T_j$, é equivalente a $Q|p_j = 1|\sum T_j$, onde as tarefas possuem tempos de processamento unitários. Ainda assim, tal resultado é útil para o desenvolvimento de uma estratégia simples de resolução do problema, tal como apresentado a seguir.

Um escalonamento para (J, Q) pode ser visto como sendo uma partição induzida (S, T), onde S representa o conjunto de tarefas em-tempo e T representa o conjunto de tarefas tardias.

Dado que os n possíveis tempos de completude podem ser gerados como indicado no Lema 1, o problema $Q|p_j = 1|\sum T_j$ pode ser resolvido aplicando-se primeiramente a regra EDD (do inglês, *Earliest Due Date first*), que ordena as tarefas em ordem não-descrescente de suas datas de términos sugeridas. Assim, considerando as tarefas nesta ordem, atribui-se a k-ésima primeira tarefa ao k-ésimo primeiro tempo de completude gerado. Ao final tem-se o conjunto $|S|$ de tarefas escalonadas em tempo e o conjunto $|T|$ de tarefas tardias. A Figura 1 ilustra a

estratégia descrita.

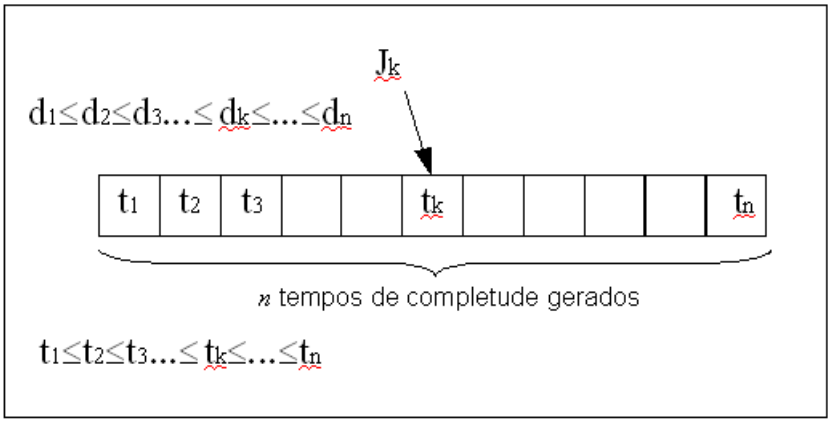

Figura 1: Esquema que representa a atribuição da k-ésima tarefa ao k-ésimo possível tempo de completude, dentre os n gerados.

Este procedimento possui complexidade de tempo $max\{n \log n, n \log m\}$, ou seja, $O(n \log n)$, sendo $O(n \log m)$ o custo para se gerar os n tempos de completude considerando-se m máquinas ($m \leq n$) e, $O(n \log n)$ para a ordenação EDD. Portanto, resolver a versão não-ponderada do problema mais geral em análise, $Q|p_j = 1|\sum T_j$ custa $O(n \log n)$, tal como para a versão mais simples $P|p_j = 1|\sum T_j$. Vale ressaltar que pelo afirmado no início da sessão, tal resultado vale também para os problemas de escalonamento $Q|p_j = p|\sum T_j$ e $P|p_j = p|\sum T_j$.

Para a resolução da versão ponderada do problema, $Q|p_j = 1|\sum w_j T_j$, com tarefas possuindo prioridades representadas por pesos de valores inteiros não-negativos, a estratégia acima não fornece a solução ótima. Desta forma, será apresentada uma caracterização baseada na generalização do Lema 1, que consiste no seguinte Lema adicional.

Lema 2 (cf. Pinedo (2002)). *Se em um escalonamento para (J, Q) existirem duas tarefas J_a e J_b tal que $d_a \leq d_b$, $p_a \leq p_b$ e $w_a \geq w_b$, então existe uma sequência ótima na qual a tarefa J_a aparece antes da tarefa J_b.*

Sendo assim, baseando-se no Lema 2, o Teorema 1 a seguir caracteriza uma solução ótima para o problema de escalonamento $Q|p_j = 1|\sum w_j T_j$.

Teorema 1 *Seja $\mathcal{S}$ um escalonamento para (J, Q) e (S, T) sua partição induzida. Então $\mathcal{S}$ minimiza $\sum w_j T_j$ se e somente se para qualquer $J_j \in T$:*

(i) não houver máquina disponível para escalonar J_j em tempo, e

(ii) $J_i \in \mathcal{S}$, tq $C_i < C_j \Rightarrow w_i T_i \geq w_j T_j$.

O Teorema 1 diz que escalonamentos ótimos para o problema $Q|p_j = 1|\sum w_j T_j$ são obtidos se dada a partição de tarefas (S, T), para toda tarefa tardia $J_j \in T$, (i) não existe uma máquina disponível (uma célula de tempo disponível em qualquer máquina do conjunto Q) na qual J_j possa ser escalonada, e (ii) para toda tarefa em tempo escalonada antes de J_j, seu atraso (*tardiness*) deve ser menor do que o atraso da tarefa tardia.

Uma estratégia ótima para o caso ponderado toma como base a solução fornecida para o caso não-ponderado e, a partir desta solução, tendo um conjunto inicial de tarefas em tempo e tarefas tardias, uma estratégia adicional é proposta para se obter um escalonamento ótimo, tal como dado a seguir.

4 Algoritmo para $Q|p_j = 1|\sum w_j T_j$

Para o caso do problema com tarefas de prioridades distintas, $Q|p_j = 1|\sum w_j T_j$, deseja-se um número mínimo de tarefas tardias (o mesmo número obtido para a versão não-ponderada do problema, tal que o peso total das tarefas tardias seja o menor possível. Sendo assim, apesar do número de tarefas tardias ser igual, o conjunto de tarefas pode ser diferente.

Sendo assim, a estratégia para a resolução do problema mais geral, $Q|p_j = 1|\sum w_j T_j$, consiste primeiramente em resolver a versão não-ponderada do problema ($Q|p_j = 1|\sum T_j$), resultando em um escalonamento com uma bipartição de tarefas (em tempo e tardias). Em seguida, faz-se trocas de tarefas em tempo por tarefas tardias de tal forma a minimizar o atraso total ponderado das tarefas, de acordo com o caracterizado na sessão anterior e descrito no pseudo-código do Algoritmo 1 apresentado a seguir.

A Figura 2 representa a busca, no k-ésimo passo, pelo menor peso dentre os pesos das tarefas em-tempo, que ainda assim seja menor do que o peso w_j da tarefa tardia J_j corrente, tal como descrito no Algoritmo 1, logo a seguir.

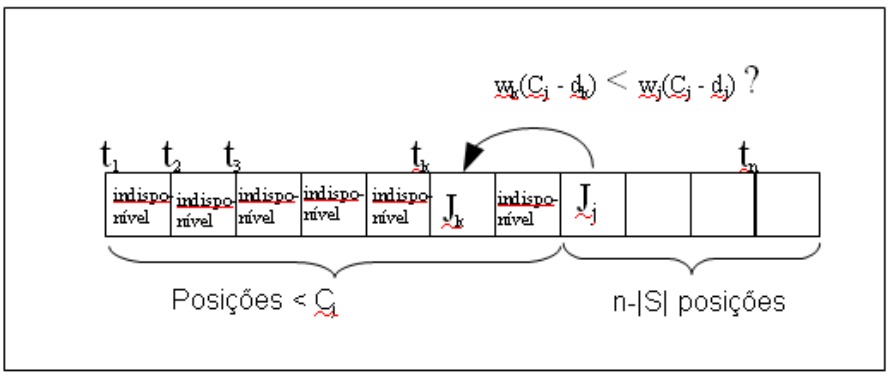

Figura 2: Esquema que representa a busca linear, no k-ésimo passo, pelo maior peso que seja menor do que o peso w_j da tarefa tardia J_j corrente,tal como descrito no Algoritmo 1.

O Algoritmo 1 apresenta o pseudo-código para a resolução do problema descrito acima: $Q|p_j = 1|\sum w_j T_j$.

Algoritmo 1 algoritmo para $Q|p_j = 1|\sum w_j T_j$

crie uma fila de prioridades fpQt, inserindo primeiramente os tempos $\frac{1}{q_i}$ de cada máquina Q_i.
faça $k = 1$, representando o índice do k-ésimo tempo de completude a ser previamente gerado.
repita
    remova o menor valor de fpQt e atribua a t_k. Assumindo que o mesmo seja $\frac{k}{q_i}$, insira em fpQt o tempo $\frac{k+1}{q_i}$.
    $k := k + 1$.
até ($k = n$ tempos de completude serem gerados)
ordene as tarefas aplicando a política EDD - $(d_1 \leq d_2 \leq ... \leq d_n)$.
para cada tarefa J_j **faça**
    se $d_j <= t_j$ **então**
        atribua J_j a t_j e indique-a como indisponível.
    senão
        defina J_j como tardia.
    fim se
fim para
escalone cada tarefa tardia no final em qualquer t_k restante.
para cada tarefa $J_j \in |T|$ **faça**
    verifique se $\exists\ J_k$, $w_k T_k^* = min_{J_l \in J}\{w_l(C_j - d_l) \mid C_l < C_j$ e $w_l < w_j\}$.
    se $\exists\ J_k$ **então**
        troque J_k por J_j no escalonamento $\mathcal{S}$.
    fim se
fim para
escalone cada tarefa tardia no final em qualquer unidade de tempo t restante.

A estratégia embutida no Algoritmo 1 pode ser realizada em tempo $O(n^2)$, tal como demonstrado a seguir.

Tal como para a versão não-ponderada do problema, a geração dos n tempos possíveis de execução acontece da mesma forma. Sendo assim, a ordenação das tarefas em ordem EDD custa $O(n \log n)$. Criar a fila de prioridades fpQt contendo inicialmente o primeiro tempo de completude possível em cada máquina, para uma tarefa de tempo de execução unitária, leva tempo $O(m)$. O primeiro laço de repetição é feito n vezes e dentro dele, a cada passo a fila de prioridades fpQt é manipulada, com uma remoção e uma inserção, seguida de reordenamento para devolver sempre seu menor valor dentre os m possíveis. Assim, tal repetição custa $O(n \log m)$.

Para o segundo laço de repetição, cada uma das n tarefas é considerada para ser escalonada (atribuída) a um dos tempos de completude gerados no passo anterior. Dentro deste laço, é verificado em tempo constante se a tarefa J_j em questão pode ser escalonada em tempo ou não.

Se a tarefa J_j puder ser escalonada em tempo, nada mais a fazer. Senão, verificar dentre as $|S|$ tarefas escalonadas (tarefas em-tempo), se existe alguma com peso menor do que J_j, pode ser feito usando-se uma outra fila de prioridades - fpQw - que forneça sempre o menor peso dentre as $|S|$ tarefas escalonadas. Isto pode ser feito em tempo $O(\log |S|)$. A troca é feita em tempo constante $O(1)$. Observa-se que a fila de prioridades fpQw deve ser inicializada de modo a validar as operações posteriores.

No terceiro e principal laço, as $|T|$ tarefas que ficaram definidas como tardias, foram alocadas nos $n - |S|$ tempos de completude restantes. Como o número de tempos de completude é invariante, ocorrendo apenas a busca e troca de tarefas associadas aos mesmos, se as condições de busca forem satisfeitas. Então, como para cada tarefa tardia J_j, é verificado dentre todas as tarefas escalonadas J_k onde $C_k < C_j$, se existe alguma que satisfaça a condição $w_k(C_j - d_k) < w_j(C_j - d_j)$ e escolhida a de menor valor, o custo de tais operações é de $O(n^2)$.

Sendo assim, a complexidade de tempo total do Algoritmo 1 é de $O(n \log m + n \log n + n^2)$ e, assumindo que $m \leq n$, a complexidade total é de $O(n^2)$.

5 Conclusões

Neste trabalho, foram apresentados algoritmos para os problemas $Q|p_j = 1| \sum w_j T_j$ e $Q|p_j = p| \sum w_j T_j$, encontram escalonamentos que minimizam o atraso ponderado das tarefas, em máquinas paralelas que possuem capacidades distintas de processamento (máquinas paralelas uniformes). Tais problemas são generalizações dos problemas $P|p_j = 1| \sum w_j T_j$ e $P|p_j = p| \sum w_j T_j$, respectivamente. Desta forma, tais problemas também podem ser resolvidos pela abordagem apresentada. O Algoritmo 1, calcula todos os possíveis tempos de completude das n tarefas e depois considera cada tarefa em ordem EDD para atribuí-las aos tempos gerados. Em seguida, realiza trocas entre tarefas tardias e tarefas em-tempo, de tal forma a minimizar a soma total das tarefas tardias ponderadas. O processo total possui complexidade de tempo $O(n^2)$.

A Tabela 1 apresenta uma relação dos problemas abordados e suas complexidades, comparando-os com os melhores resultados disponíveis na literatura.

Tais resultados apresentados neste artigo, juntamente com os propostos por Dourado et al. (2009), Rodrigues (2009) e Dourado et al. (2008), motivam a abordagem dos problemas $Q|p_j = 1; r_j| \sum U_j$ e $Q|p_j = 1; r_j| \sum T_j$, bem como suas versões com pesos, $Q|p_j = 1; r_j| \sum w_j U_j$ e $Q|p_j = 1; r_j| \sum w_j T_j$, que se encontram em aberto, dentre as pesquisas futuras.

Referências

Baptiste, P., Brucker, P., Knust, S. e Timbkovsky, V. (2004). “Ten Notes on Equal-Processing-

Tabela 1: Quadro comparativo dos resultados.

Problema	*Complexidade*	
	Literatura	*Resultados Propostos*
$Q\|p_j = 1\|\sum T_j$ $Q\|p_j = p\|\sum T_j$	$O(n^3)$	$O(n \log n)$
$Q\|p_j = 1\|\sum w_j T_j$ $Q\|p_j = p\|\sum w_j T_j$	$O(n^3)$	$O(n^2)$
$Q\|p_j = 1; r_j\|\sum T_j$ $Q\|p_j = 1; r_j\|\sum w_j T_j$	em aberto	em estudo

Time Scheduling: at the Frontiers of Solvability in Polynomial Time", *Quaterly Journal of the Belgian, French and Italian Operations Research Societies - 4OR*, Springer-Verlag, 2:111-127.

Brucker, P. e Knust, S. (2010). "Complexity Results for Scheduling Problems", *Disponível em http://www.mathematik.uni-osnabrueck.de/research/OR/class*. Acesso em abril de 2010.

Brucker, P. e Kravchenko, S. (2006). "Scheduling Equal Processing Time Jobs to Minimize the Weighted Number of Late Jobs", *Journal of Mathematical Modelling and Algorithms*. Springer-Verlag. Vol. 5, 2:143-165.

Brucker, P. (2004). "*Scheduling Algorithms*", 4rd ed., Springer-Verlag, New York.

Bruno J., Coffman, E. e Sethi, R. (1974). "Scheduling independent tasks t reduce mean finishing time", *In: Communications of the ACM*, Vol. 17, pp. 382-387.

Chrobak, M, Durr, C., Jawor, W., Kowalik, L. e Kurowski, M. (2006). "A Note on Scheduling Equal-Length Jobs To Maximize Throughput", *Journal of Scheduling*, 9: 71-73.

Dessouky M., Lageweg, B., Lenstra, J. e van del Velde, S. (1990). "Scheduling identical jobs on uniform parallel machines", *In: Statistica Neerlandica*, Vol. 44(3), pp. 115-123.

Dourado, M., Rodrigues, R. e Szwarcfiter, J. (2010). "Three algorithms to minimizing the number of tardy jobs in a classical scheduling problem", *submetido para publicação*.

Dourado, M., Rodrigues, R. e Szwarcfiter, J. (2009). "Scheduling unit time jobs with integer release dates to minimize the weighted number of tardy jobs", *In: Annals of Operations Research*, Springer Netherlands, 169(1),81–91, DOI: 10.1007/s10479-008-0479-y, ISSN: 0254-5330 (Print) 1572-9338 (Online).

Durr, C. (2010). "The Scheduling Zoo: a searchable bibliography on scheduling". *Disponível em http://www.lix.polytechnique.fr/ durr/query/*. Acesso em abril de 2010.

Gonzalez T., Sahni, S. (1978). "Preemptive scheduling of uniform processor systems", *In: Journal of the Association for Computing Machinery*, Vol. 25, pp. 92-101.

Graham, R., Lawler, E., Lenstra, J. e Rinnooy-Kan, A. (1979). "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey", *Annals of Discrete Mathematics*, 5: 287-326.

Lawler, E. e Moore, J. (1969). "A Functional Equation and its Application to Resource Allocation and Sequencing Problems", *Management Science*, 16:77-84.

Lenstra, J., Rinooy Kan, A., Brucker, P. (1977). "Complexity of machine scheduling problems", *Annals of Discrete Mathematics*, 1:343-362.

Leung, J. (ed.) (2004). "*Handbook of Scheduling: Algorithms, Models, and Performance Analysis", Computer and Information Science Series*, Chapman&Hall and CRS Press.

Moore, J. (1968). "A n Job, One Machine Sequencing Algorithm For Minimizing the Number of Late Jobs", *Management Science*, 15:102-109.

Pinedo, M. (2002), "Scheduling: Theory, Algorithms, and Systems", *Prentice-Hall*, 2a. ed.

Rodrigues, R. (2009). "Caracterizações e Algoritmos para Problemas Clássicos de Escalonamento", *Doctoral Thesis - Computer and System Engineering*, COPPE, UFRJ.