SEQUENCIAMENTO DE TAREFAS EM MÁQUINAS PARALELAS COM TEMPOS DE PREPARAÇÃO E PRECEDÊNCIA ENTRE AS TAREFAS: MODELAGEM E HEURÍSTICAS CONSTRUTIVAS

Felippe Moreira Faêda, José Elias C. Arroyo, André Gustavo dos Santos
Departamento de Informática, Universidade Federal de Viçosa (UFV)
Viçosa - Minas Gerais - MG - Brasil - 36570-000
felippe.faeda@ufv.br,{jarroyo,andre}@dpi.ufv.br

Michele Monaci
Università degli Studi di Padova
Via Gradenigo 6/A, 35131, Padova - Itália
monaci@dei.unipd.it

RESUMO

Este trabalho aborda o problema de sequenciamento de tarefas em máquinas paralelas não-relacionadas considerando restrição de precedência entre as tarefas e tempos de preparação dependentes da sequência. Este problema tem como objetivo minimizar o tempo máximo de conclusão do sequenciamento, conhecido como *makespan*. Considerando a restrição de precedência, nenhuma tarefa pode iniciar seu processamento sem que todas as suas tarefas predecessoras tenham sido finalizadas. Para resolver este problema, apresentamos dois modelos diferentes de programação linear inteira mista e propomos uma heurística que utiliza diferentes regras de prioridade, obtendo diferentes heurísticas construtivas. Experimentos computacionais são realizados a fim de comparar o desempenho dos modelos matemáticos, que são resolvidos através do software CPLEX, e comparar o desempenho das heurísticas construtivas. Os resultados apontaram que um dos modelos é mais eficiente na geração de soluções para instâncias maiores em um determinado tempo de CPU. Os resultados também mostraram que existem diferenças significativas entre as heurísticas construtivas.

PALAVRAS CHAVE. Máquinas paralelas, heurísticas, regras de prioridade, otimização combinatória

Área Principal: (OC - Otimização Combinatória, PM - Programação Matemática)

ABSTRACT

This paper addresses the unrelated parallel machine scheduling problem with precedence constraints between the jobs and sequence-dependent setup times. This problem consists to minimize the maximum completion time of the jobs, called *makespan*. In this problem, we consider precedence constraints. The precedence constraints force a job not to be started before all its predecessors jobs are finished. To solve this problem, we provide two different mixed integer programming model and we propose a heuristic that using various priority rules, getting different constructive heuristics. Computational experiments are conducted to compare the performance of mathematical models optimally solved using CPLEX software and to compare the performance of constructive heuristics. The results present that one of the models is more efficient in generating solutions for large instances on a given CPU time. The results showed that there are significant differences between the heuristics.

KEYWORDS. Parallel machine, heuristics, priority rules, combinatorial optimization

Main Area: (OC - Combinatorial Optimization, PM - Mathematical Programming)

1. Introdução

Este trabalho aborda o problema de sequenciamento de tarefas em máquinas paralelas não-relacionadas considerando restrição de precedência entre as tarefas e tempos de preparação dependentes da sequência. Neste problema, n tarefas precisam ser alocadas em m máquinas paralelas não-relacionadas (R_m) com o objetivo de minimizar o tempo máximo de conclusão do sequenciamento, também conhecido como *makespan* (C_{max}). Com a restrição de precedência ($prec$), nenhuma tarefa pode iniciar seu processamento sem que todas as suas tarefas predecessoras tenham sido finalizadas. Além disso, são considerados tempos de preparação (S_{ijk}) dependentes do sequenciamento das tarefas e dependentes das máquinas.

De acordo com a classificação de Graham et al. (1979), esse problema pode ser referenciado como $R_m \mid prec,\ S_{ijk} \mid C_{max}$. Johnson e Garey (1979) provam que o problema básico de sequenciamento em máquinas paralelas idênticas ($P_m \mid\mid C_{max}$) é NP-difícil a partir de $m = 2$ máquinas. Como o problema $R_m \mid prec,\ S_{ijk} \mid C_{max}$ é uma generalização do problema $P_m \mid\mid C_{max}$, então também pode ser considerado um problema NP-difícil.

Segundo Rabadi (2006), o problema de sequenciamento em máquinas paralelas não-relacionadas considerando tempo de preparação é muito comum em indústrias de tecido, vidro, semicondutores, produtos químicos e algumas indústrias de serviço. Liu (2013), também cita que o problema, considerando restrição de precedência, ocorre tipicamente em escritórios ou ambientes de gerenciamento de projetos. Hassan Abdel-Jabbar, Kacem e Martin (2014) abordam o mesmo problema, porém, aplicado ao contexto de computação em nuvem.

Na literatura são encontrados trabalhos que consideram três tipos de máquinas paralelas: idênticas, uniformes e não-relacionas. A seguir são citados trabalhos da literatura que consideram restrições de precedências entre as tarefas para diferentes tipos de máquinas.

Para problemas com máquinas paralelas idênticas, Ramachandra e Elmaghraby (2006) propõem uma formulação matemática e um algoritmo de programação dinâmica para o problema, onde as tarefas são sequenciadas em duas máquinas idênticas e consideram a minimização do tempo de conclusão ponderado. Aho e Mäkinen (2006) consideram o problema com tempo de processamento igual a 1 para todas as tarefas e propõem um algoritmo polinomial para minimização do *makespan*. O algoritmo resolve o problema de forma exata respeitando algumas propriedades do grafo de precedência formado pelas restrições de precedência. Gacias, Artigues e Lopez (2010) propõem um algoritmo *branch-and-bound* e uma busca local para o problema considerando tempos de preparação entre as tarefas e minimização do tempo de conclusão total. Driessel and Mönch (2011) introduzem variações da meta-heurística *variable neighborhood search* (VNS) para minimização do atraso total ponderado considerando tempos de preparação, restrição de precedência e tempo de liberação das tarefas.

Para problemas com máquinas paralelas uniformes, Brucker et al. (1999) mostram que o problema de sequenciamento de tarefas com restrição de precedência em duas máquinas uniformes com o objetivo de minimizar o *makespan* pode ser resolvido em tempo linear. Kim (2011) propõe uma heurística baseada em programação linear para o problema considerando minimização do tempo de conclusão total ponderado. Van Zuylen (2011) aborda o problema onde as máquinas são controladas por agentes autônomos (*selfish agents*). Esse autor propõe um algoritmo de aproximação $O(\log m)$ para o problema considerando restrição de precedência e minimização do *makespan*.

Para problemas com máquinas paralelas não-relacionadas, Herrmann, Proth e Sauer (1997) introduzem algumas heurísticas para o problema considerando restrição de precedência em cadeia entre as tarefas com objetivo de minimizar o *makespan*. Kumar et al. (2009) apresentam um algoritmo de aproximação para o problema considerando restrição de precedência, minimização do *makespan* e tempo de conclusão total. Liu e Yang (2011) propõem uma heurística construtiva baseada em regra de prioridade (SS) para o problema. Liu (2013) aborda um algoritmo genético híbrido para minimização do atraso total para o problema com restrição de precedência arbitrária.

Este trabalho aborda uma variação do problema de máquinas paralelas não-relacionadas,

onde é utilizada restrição de precedência arbitrária entre as tarefas e tempos de preparação dependentes da sequência. A ausência de trabalhos na literatura para esta variação do problema, sua complexidade e sua importância prática, são as principais motivações deste trabalho. Para determinar soluções ótimas para algumas instâncias do problema, são propostos dois modelos matemáticos de programação linear inteira mista. Também são apresentadas sete heurísticas construtivas utilizando diferentes regras de prioridade. Segundo Arroyo e Armentano (2004), heurísticas construtivas são comumente utilizadas para gerar soluções iniciais de diferentes metaheurísticas e o desempenho das metaheurísticas, geralmente, depende da qualidade da solução inicial.

Este artigo está organizado da seguinte forma: na seção 2, o problema é definido e dois modelos matemáticos são apresentados. Na seção 3, são apresentadas as heurísticas construtivas e as regras de prioridade utilizadas. Nas seções 4 e 5, são descritos os experimentos computacionais realizados e as conclusões, respectivamente.

2. Definição do Problema

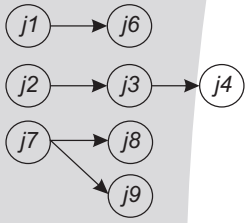
No problema de sequenciamento de tarefas em máquinas paralelas não-relacionadas existem um conjunto $J = \{1, ..., n\}$ de n tarefas e um conjunto $M = \{1, ..., m\}$ de m máquinas, onde cada tarefa $j \in J$ deve ser processada uma única vez por apenas uma máquina $i \in M$. As máquinas são consideradas não-relacionadas, ou seja, elas possuem diferentes capacidades de processamento. Portanto, cada tarefa j possui um tempo de processamento p_{ij} para cada máquina i. Além disso, são considerados tempos de preparação. O tempo de preparação (S_{ijk}) é o tempo gasto para preparar uma máquina i para processar uma tarefa k alocada imediatamente após a tarefa j. Os tempos de preparação são dependentes da sequência e dependentes da máquina, isto é, $S_{ijk} \neq S_{ikj}$ e $S_{ijk} \neq S_{hjk}$, para duas diferentes tarefas j e k, e duas diferentes máquinas i e h.

O problema abordado também considera a restrição de precedência entre as tarefas. Nesta restrição é dado um conjunto $A = \{a_1, ..., a_p\}$ de p restrições de precedência, onde a i-ésima restrição $a_i = (j, k)$ representa que a tarefa j deve preceder a tarefa k, ou seja, o tempo de início do processamento da tarefa k não deve ser menor que o tempo de conclusão da tarefa j. Essa restrição garante que uma tarefa não inicie seu processamento até que todas as suas tarefas predecessoras tenham sido finalizadas. Considerando as tarefas contidas em A como sendo vértices de um grafo, e as precedências como sendo as arestas, o grafo formado pelo conjunto A pode ser visto no exemplo da Figura 1(b). Para que uma instância seja válida, o grafo de precedências não pode conter ciclos. O problema abordado tem como objetivo encontrar o sequenciamento das n tarefas nas m máquinas a fim de minimizar o tempo máximo de conclusão das tarefas, chamado de *makespan* (C_{max}).

Como exemplo para o problema, são considerados um conjunto de $n = 10$ tarefas, um conjunto de $m = 3$ máquinas paralelas não-relacionadas (M1, M2, M3) e um conjunto A representado pelas Figuras 1(a) e 1(b). A Figura 2 mostra uma solução ótima para o exemplo.

Predecessora	Sucessora
j1	*j6*
j2	*j3*
j3	*j4*
j7	*j8*
j7	*j9*

(a) Exemplo do conjunto A



(b) Grafo de Precedência

Figura 1: Exemplo de representação da restrição de precedência

De acordo com o conjunto A da Figura 1(a), a tarefa $j7$ é predecessora das tarefas $j8$ e $j9$. Com isso, pode ser observado na Figura 2 que a tarefa $j9$, alocada na máquina M3, e a tarefa $j8$, alocada na máquina M1, iniciam em um instante de tempo superior ao tempo de conclusão da tarefa $j7$, finalizada no instante de tempo 6. As demais restrições de precedência podem ser visualizadas pelas setas vermelhas contidas na Figura 2. Os blocos pretos representam os tempos de preparação das máquinas. O valor da função objetivo ou *makespan* deste exemplo é $C_{max} = 25$.

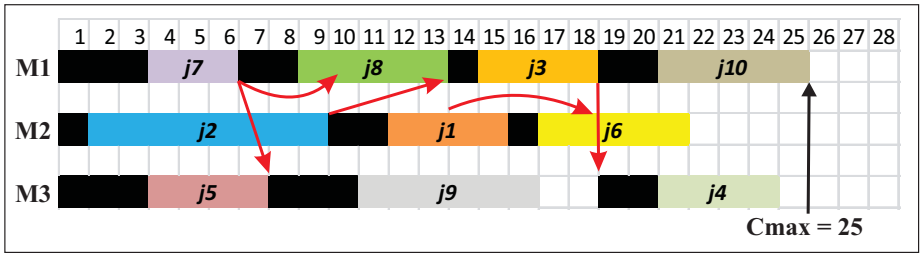


Figura 2: Exemplo de sequenciamento para o problema.

2.1. Modelagem Matemática

Nesta seção serão apresentados dois modelos de programação linear inteira mista (PLIM) para o problema abordado. Ambos os modelos envolvem os seguintes parâmetros de entrada:

- J: conjunto de n tarefas, $J = \{1, ..., n\}$;
- M: conjunto de m máquinas, $M = \{1, ..., m\}$;
- p_{ij}: tempo de processamento da tarefa j na máquina i;
- S_{ijk}: tempo de preparação da máquina i para processar a tarefa k depois da tarefa j;
- A: conjunto de restrições de precedência, $A = \{a_1, ..., a_p\}$, onde a i-ésima restrição $a_i = (j, k)$ representa que a tarefa j deve preceder a tarefa k;
- J_0: conjunto de tarefas J incluindo a tarefa fictícia 0. A tarefa 0 será utilizada para viabilizar restrições que precisam consultar as tarefas antecessoras e sucessoras a uma determinada tarefa. A tarefa 0 será a antecessora de todas as primeiras tarefas e também a sucessora de todas as últimas tarefas de cada máquina.
- B: um número inteiro suficientemente grande.

2.1.1. Modelo 1

A seguir será apresentado um modelo matemático ($Modelo^1$) adaptado do trabalho de Rabadi et al. (2006), onde abordaram um problema com as mesmas características, exceto as restrições de precedência. Este modelo utiliza uma variável de decisão binária que indica se uma tarefa k está sequenciada imediatamente após a tarefa j na máquina i. Além disso, algumas máquinas podem não possuir tarefas alocadas. As variáveis de decisão do modelo são:

- x_{ijk}: igual a 1, se a tarefa k é processada imediatamente após a tarefa j na máquina i, caso contrário, igual a zero;
- C_j: tempo de conclusão da tarefa j;
- C_{max}: valor do *makespan*.

O $Modelo^1$ é demonstrado nas equações de (1) a (10):

$$Min \quad C_{max} \tag{1}$$

$$st. \quad \sum_{j=0, j\neq k}^{n} \sum_{i=1}^{m} x_{ijk} = 1, \quad \forall k \in J \tag{2}$$

$$\sum_{j=0, j\neq h}^{n} x_{ijh} = \sum_{k=0, k\neq h}^{n} x_{ihk}, \quad \forall h \in J, \ \forall i \in M \tag{3}$$

$$\sum_{k=0}^{n} x_{i0k} \leq 1, \quad \forall i \in M \tag{4}$$

$$C_j + \sum_{i=1}^{m} (S_{ijk} + p_{ik}) x_{ijk} + B(\sum_{i=1}^{m} x_{ijk} - 1) \leq C_k, \forall j \in J_0, \ \forall k \in J \tag{5}$$

$$C_k - C_j \geq \sum_{i=1}^{m} \sum_{l=0}^{n} (S_{ilk} + p_{ik}) \cdot x_{ilk}, \qquad \forall (j, k) \in A \tag{6}$$

$$C_0 = 0 \tag{7}$$
$$C_{max} \geq C_j, \quad \forall j \in J \tag{8}$$
$$C_j \geq 0, \quad \forall j \in J \tag{9}$$
$$x_{ijk} \in \{0,1\}, \quad \forall j,k \in J_0, \ \forall i \in M. \tag{10}$$

A Equação (1) representa a função objetivo que é a minimização do *makespan*. As restrições (2) garantem que cada tarefa será processada apenas uma vez e por uma única máquina. As restrições (3) garantem que cada tarefa deve ter somente uma tarefa imediatamente predecessora e uma imediatamente sucessora. As restrições (4) fazem com que no máximo uma tarefa seja sequenciada na primeira posição de cada máquina. As restrições (5) são usadas para calcular o tempo de conclusão de cada tarefa e também não permitem que uma tarefa seja predecessora e sucessora de uma mesma tarefa. As restrições (6) são as restrições de precedência. Essas restrições garantem que cada tarefa não pode ser inicializada antes que todas as suas tarefas predecessoras tenham sido finalizadas. A restrição (7) obriga a tarefa 0 ter tempo de conclusão igual a zero, portanto, não influencia no cálculo do *makespan*. As restrições (8) definem o *makespan*. Finalmente, as restrições (9) e (10), respectivamente, garantem que o tempo de conclusão das tarefas tenha valor positivo e define que as variáveis x_{ijk} sejam binárias.

2.1.2. Modelo 2

A seguir será apresentado um modelo matemático ($Modelo^2$) adaptado de Liu (2013), onde o autor abordou o problema de minimização do atraso total considerando máquinas paralelas não-relacionadas com precedência entre as tarefas. Este modelo se difere do $Modelo^1$ pelo fato das restrições considerarem as posições das máquinas. Neste modelo é utilizado o parâmetro ϕ, que representa o número máximo de posições em cada máquina onde as tarefas serão alocadas. Além disso, cada máquina pode ter no máximo n posições, sendo que algumas máquinas podem não possuir tarefas alocadas. As variáveis de decisão do modelo são:

x_{ijr}: igual a 1, se a tarefa j está alocada na posição r da máquina i, caso contrário, igual a zero;
σ_{itkr}: igual a 1, se a tarefa k é processada logo após a tarefa t na posição r da máquina i, caso contrário, igual a zero;
C_j: tempo de conclusão da tarefa j;
C_{max}: valor do *makespan*.

O $Modelo^2$ é demonstrado nas equações de (11) a (24):

$$Min \quad C_{max} \tag{11}$$
$$st. \quad \sum_{i=1}^{m}\sum_{r=1}^{\phi} x_{ijr} = 1, \quad \forall j \in J \tag{12}$$
$$\sum_{j=1}^{n} x_{ijr} \leq 1, \quad \forall r \in \{1,...,\phi\}, \ \forall i \in M \tag{13}$$
$$\sum_{k=1}^{n} x_{ikr} - \sum_{j=1}^{n} x_{ij(r-1)} \leq 0, \quad \forall i \in M, \ \forall r \in \{2,...,\phi\} \tag{14}$$
$$C_k \geq (S_{i0k} + p_{ik})\, x_{ik1}, \quad \forall k \in J, \ \forall i \in M \tag{15}$$
$$C_k - C_j + B\,(2 - x_{ikr} - x_{ij(r-1)}) \geq S_{ijk} + p_{ik}, \quad \forall j,k \in J, j \neq k, \ \forall i \in M, \ \forall r \in \{2,...,\phi\} \tag{16}$$
$$C_k - C_j \geq \sum_{i=1}^{m}(S_{i0k} + p_{ik}) \cdot x_{ik1}, \quad \forall (j,k) \in A \tag{17}$$
$$C_k - C_j \geq \sum_{t=1}^{n}\sum_{i=1}^{m}\sum_{r=2}^{\phi}(S_{itk} + p_{ik}) \cdot (x_{it(r-1)} \cdot x_{ikr}), \quad \forall (j,k) \in A \tag{18}$$

$$C_j \geq 0, \quad \forall j \in J \tag{19}$$
$$C_{max} \geq C_j, \quad \forall j \in J \tag{20}$$
$$x_{ijr} \in \{0,1\}, \quad \forall i \in M,\ \forall j \in J\ \forall r \in \{1,...,\phi\} \tag{21}$$

Para a linearização das restrições em (18), considere $\sigma_{itkr} = x_{it(r-1)} \cdot x_{ikr}$. A seguir estão as restrições para linearização:

$$-x_{it(r-1)} + \sigma_{itkr} \leq 0, \tag{22}$$
$$-x_{ikr} + \sigma_{itkr} \leq 0, \tag{23}$$
$$x_{it(r-1)} + x_{ikr} - \sigma_{itkr} \leq 1, \tag{24}$$
$$\forall k,t \in J, i \neq t,\ \forall i \in M,\ \forall r \in \{2,...,\phi\}$$

A Equação (11) representa a minimização do *makespan*. As restrições (12) garantem que cada tarefa j seja designada para uma única posição de alguma máquina. As restrições (13) garantem que no máximo uma tarefa será designada para cada posição. As restrições em (14) garantem que cada tarefa, partindo da posição $r = 2$, deve ter uma tarefa na posição anterior $(r-1)$. As restrições em (15) e (16) garantem que o tempo de conclusão de uma tarefa k em uma máquina i seja maior ou igual ao tempo de conclusão da tarefa antecessora j mais o tempo de processamento e tempo de preparação da tarefa k. As restrições (17) e (18) obedecem as relações de precedência entre as tarefas. As restrições de (19-21) definem o tipo das variáveis de decisão. As restrições de (22-24) são responsáveis pela linearização necessária provocada pela restrição (18).

3. Heurística Construtiva Proposta

Nesta seção será apresentado o funcionamento de uma heurística construtiva (HC) adaptada de Liu (2013), onde a heurística utiliza uma regra de prioridade para selecionar uma tarefa a ser alocada e sequenciada em uma determinada máquina. A heurística HC gera uma solução viável de boa qualidade e com pouco tempo computacional após n iterações. Em cada iteração, um conjunto D contendo todas as tarefas disponíveis para o sequenciamento é formado e então uma tarefa $j^* \in D$ é selecionada de acordo com a regra de prioridade. Depois uma máquina $i^* \in M$ é escolhida para sequenciar a tarefa j^*. As variáveis usadas pela heurística são descritas a seguir:

- t: instante de tempo t do sequenciamento;
- H: conjunto contendo todas as tarefas já sequenciadas pela heurística;
- F: conjunto contendo todas as tarefas sequenciadas e finalizadas até o instante de tempo t;
- D: conjunto de tarefas não sequenciadas e disponíveis para sequenciamento, onde todas as suas tarefas predecessoras pertencem ao conjunto de tarefas concluídas F;
- I_j: tempo de início de processamento da tarefa $j \in J$;
- C_j: tempo de conclusão da tarefa $j \in J$;
- R_i: tempo de conclusão da última tarefa alocada na máquina i;
- k: última tarefa alocada em uma determinada máquina i;
- z: tempo de conclusão máximo das tarefas predecessoras de j^*;
- (j^*, i^*): tarefa $j^* \in D$ e máquina $i^* \in M$ selecionados pela regra de prioridade.

No Algoritmo 1 é descrito o funcionamento da heurística construtiva. Primeiramente, as variáveis u, H e R_i são inicializadas. Em cada iteração u do algoritmo, uma tarefa j é sequenciada. O algoritmo termina quando todas as tarefas do conjunto J são sequenciadas. Na linha 4, T é o conjunto que contém todos os tempos de conclusão de cada máquina $i \in M$. Em seguida, na linha 5, o instante de tempo t do sequenciamento é determinado pelo menor valor contido em T, ou seja, o menor valor de tempo de conclusão de todas as máquinas. Nas Linhas 6 e 7, são calculados, respectivamente, o conjunto F contendo as tarefas finalizadas até o instante de tempo t e o conjunto D, contendo as tarefas não sequenciadas e disponíveis para sequenciamento.

Conforme linha 6, uma tarefa j está concluída em um instante de tempo t, se o seu tempo de conclusão C_j é menor ou igual a t, portanto, passa a pertencer ao conjunto F. E uma tarefa

j não alocada está disponível para sequenciamento em um instante de tempo t, se todas as suas tarefas predecessoras estão concluídas, ou seja, pertencem ao conjunto F. Nas linhas de 8 a 13, se o conjunto de tarefas D estiver vazio, isto é, quando não existem tarefas disponíveis no instante de tempo t, o valor de t passa a valer o próximo menor valor de tempo de conclusão contido em T e o conjunto D é recalculado. Esse processo se repete até que o conjunto D não fique vazio.

Na linha 14, será aplicada a regra de prioridade para selecionar uma tarefa $j^* \in D$ que será alocada na última posição de uma máquina $i^* \in M$ também selecionada pela regra. Na linha 15, o valor de z é calculado como sendo o tempo máximo de conclusão das tarefas predecessoras de j^*. Em seguida, são calculados o tempo de conclusão C_{j^*} e o tempo de início I_{j^*} da tarefa selecionada. O tempo de conclusão da máquina i^* (R_{i^*}) passa a ser o tempo de conclusão da tarefa j^* e a tarefa entra para o conjunto H, onde estão todas as tarefas já sequenciadas. Por fim, a variável u é incrementada para a próxima iteração. Quando todas as tarefas já estiverem sequenciadas, o algoritmo retorna o maior valor do tempo de conclusão das tarefas $j \in J$, que é o *makespan*.

Na próxima seção serão apresentadas sete regras de prioridade a serem utilizadas na heurística HC. As regras de prioridade são denotadas de $R1$ a $R7$.

Algoritmo 1: HC

1 **início**
2 $u \leftarrow 0, H \leftarrow \emptyset, R_i \leftarrow 0, \forall i \in M$;
3 **enquanto** $u \leq n$ **faça**
4 $T = \{R_i \mid i \in M\}$;
5 $t = min\{r \mid r \in T\}$;
6 $F = \{j \mid C_j \leq t, \forall j \in H\}$;
7 $D = \{j \mid j \in J \; e \; j \notin H, k \in F, \forall k \in A_{(k,j)}\}$
8 **enquanto** $D = \emptyset$ **faça**
9 $T = T \setminus \{t\}$;
10 $t = min\{r \mid r \in T\}$;
11 $F = \{j \mid C_j \leq t, \forall j \in H\}$;
12 $D = \{j \mid j \in J \; e \; j \notin H, k \in F, \forall k \in A_{(k,j)}\}$
13 **fim**
14 (j^*, i^*) = Regra_de_Prioridade(D, M); //seleciona uma tarefa $j^* \in D$ e uma máquina $i^* \in M$;
15 $z = max\{C_l \mid \forall l \in A_{(l,j^*)}\}$;
16 $C_{j^*} = max\{z, R_{i^*}\} + s_{i^*kj^*} + p_{j^*i^*}$
17 $I_{j^*} = C_{j^*} - (s_{i^*kj^*} + p_{j^*i^*})$;
18 $R_{i^*} = C_{j^*}$; $\quad H = H \cup \{j^*\}$;
19 $u = u + 1$;
20 **fim**
21 **fim**
22 **retorna** $C_{max} = max\{C_j \mid j \in J\}$.

3.1. Regras de Prioridade

Em cada iteração da heurística HC, uma regra de prioridade é utilizada para selecionar uma tarefa $j^* \in D$ e uma máquina $i^* \in M$ na qual a tarefa j^* será alocada. Neste trabalho, sete regras de prioridade ($R1$ - $R7$) são adaptadas de trabalhos da literatura que abordam o problema de sequenciamento em máquinas paralelas. Essas regras foram adaptadas a fim de considerar as restrições de precedência e os tempos de preparação abordados neste trabalho. A seguir são descritas as regras de prioridade utilizadas:

- $R1$: adaptada de Weng, Lu e Ren (2001), a regra utilizada seleciona uma tarefa $j^* \in D$ que obtiver a menor média do tempo de processamento somado ao tempo de preparação.

Depois será selecionada uma máquina $i^* \in M$ que gerar menor tempo de conclusão para sequenciar a tarefa j^*. Dado que k é a última tarefa alocada em uma máquina i, R_i é o tempo de conclusão da tarefa k sequenciada na máquina i e z é o tempo de conclusão máximo entre as tarefas predecessoras de j^*, a regra se resume nas seguintes equações:

$$j^* = argmin\{p_j : j \in J\}, \quad p_j = \frac{\sum_{i=1}^{m} S_{ikj} + p_{ij}}{m}, \quad \forall j \in D \tag{25}$$

$$i^* = argmin\{max(z, R_i) + S_{ikj^*} + p_{ij^*} : i \in M\} \tag{26}$$

- $R2$: adaptada de Weng, Lu e Ren (2001), a regra utilizada seleciona uma tarefa j^* da mesma forma que a Regra 1. No entanto, de acordo com a equação a seguir e considerando k sendo a última tarefa alocada em uma máquina i, a máquina i^* selecionada será a máquina onde a tarefa j^* possuir menor tempo de processamento acrescido do tempo de preparação.

$$i^* = argmin\{S_{ikj^*} + p_{ij^*} : i \in M\} \tag{27}$$

- $R3$ e $R4$ são similares a $R1$ e $R2$, respectivamente, no entanto, a tarefa $j^* \in D$ será a tarefa que tiver menor tempo de processamento acrescido do tempo de preparação, definido pela equação:

$$j^* = argmin\{S_{ikj} + p_{ij} : j \in J, i \in M\} \tag{28}$$

- $R5$: utilizada por Lin, Pfund e Fowler (2011), a regra utilizada seleciona a tarefa $j^* \in D$ com menor tempo de conclusão se alocada no final de cada máquina $i \in M$. Depois, a tarefa j^* será alocada na máquina i^* que gerar o menor tempo de conclusão para j^*. Dado que R_i é o tempo de conclusão da última tarefa k alocada em i, os valor de j^* e i^* são definidos por:

$$j^* = argmin\{max(z, R_i) + S_{ikj} + p_{ij} : j \in J, i \in M\} \tag{29}$$

$$i^* = argmin\{max(z, R_i) + S_{ikj} + p_{ij^*} : i \in M\} \tag{30}$$

- $R6$: adaptada de Liu e Yang (2011), nesta regra, a tarefa $j^* \in D$ será a tarefa com maior desvio padrão do tempo de processamento mais o tempo de preparação nas máquinas $i \in M$. A máquina i^* será a que gerar o menor tempo de conclusão para a tarefa j^*, conforme Equação (30). A seguir são apresentados o cálculo da média aritmética e do desvio padrão:

$$E_j = \frac{1}{|M|}\sum_{i=1}^{m}(p_{ij} + S_{ikj}), \quad \sigma_j^2 = \frac{1}{|M|}\sum_{i=1}^{m}((p_{ij} + S_{ikj}) - E_j)^2, \quad \forall j \in J \tag{31}$$

- $R7$: adaptada de $R5$, esta regra consiste em criar um conjunto L contendo uma certa porcentagem α de tarefas com menor tempo de conclusão nas máquinas de acordo com a equação (29). A tarefa selecionada $j^* \in L$, será a tarefa que preceder o maior número de tarefas de acordo com as restrições de precedência. A máquina i^* será a que gerar menor tempo de conclusão para j^*, conforme Equação (30).

A regra $R7$, além de conceder prioridade às tarefas de menor tempo de conclusão, seleciona a tarefa que preceder o maior número de tarefas. Quando uma tarefa, de acordo com as precedências, tem maior número de tarefas sucessoras e é sequenciada primeiro, maior será a chance de aumentar o número de tarefas disponíveis em D nas próximas iterações do algoritmo. Com isso, tarefas com menor tempo de conclusão poderão ser sequenciadas primeiro. A heurística HC e as regras de prioridade apresentadas geram sete heurísticas construtivas denotadas por $HC1$ - $HC7$.

4. Experimentos Computacionais

Nesta seção são comparados os dois modelos matemáticos propostos, também é analisado o desempenho das sete heurísticas e, por fim, é feita uma comparação entre a melhor heurística com o melhor modelo. As heurísticas foram codificadas em C++ e os modelos foram desenvolvidos utilizando a biblioteca Concert Technology. Com esta biblioteca foi possível implementar os modelos matemáticos usando a linguagem C++ e resolvê-los através do software CPLEX versão 12.4. Os

experimentos foram realizados em um computador com processador Intel Core i7 com 4.00 GHz de clock, 32 GB de memória e sistema operacional Windows 8 64 bits.

Os experimentos utilizam três conjuntos de instâncias geradas que variam de acordo com o número de tarefas (n), número de máquinas (m), intervalo de tempo de preparação (s) e nível de densidade das restrições de precedência (d). O primeiro conjunto é formado por instâncias pequenas com $n \in \{6, 8, 10, 12\}$ e $m \in \{2, 3, 4, 5\}$. O segundo conjunto é formado por instâncias de tamanho médio com $n \in \{15, 20, 25, 30\}$ e $m \in \{2, 5, 10\}$. E o terceiro conjunto é formado por instâncias grandes com $n \in \{30, 60, 90, 120, 150, 180\}$ e $m \in \{5, 10, 15\}$. Em ambos os conjuntos, as instâncias foram geradas usando $d \in \{0.2, 0.4, 0.6\}$ e intervalos de tempo de preparação $s \in \{[0, 9], [0, 49], [0, 99]\}$. Foram geradas 5 instâncias para cada combinação de n, m, d e s. No total de instâncias, foram geradas 720 pequenas, 540 intermediárias e 810 grandes.

A geração das instâncias é similar ao apresentado por Liu (2013). O tempo de processamento de cada tarefa é um número inteiro gerado pela distribuição uniforme dentro do intervalo de [1,100]. As restrições de precedência são geradas usando a probabilidade P_{ij} de uma tarefa i ter relação de precedência com a tarefa j. Para uma dada densidade d, a probabilidade é definida por:

$$P_{ij} = \frac{d \times (1-d)^{j-i-1}}{(1-d) \times (1-(1-d)^{j-i-1})}, \forall\, 1 \leq i < j \leq n, i \neq j. \tag{32}$$

4.1. Calibração do parâmetro α usado pela heurística $HC7$

A heurística $HC7$ foi testada utilizando diferentes valores para o parâmetro α, que é utilizado para definir o tamanho do conjunto L de tarefas candidatas à seleção. Os valores testados foram: $\alpha(\%) = \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. A heurística foi executa uma vez para o conjunto de instâncias grandes. A métrica utilizada para avaliar os resultados dos experimentos é o Desvio Percentual Relativo (RPD) que é obtido pela seguinte fórmula:

$$RPD\% = \frac{f - f_{best}}{f_{best}} \times 100, \tag{33}$$

onde f corresponde ao valor da função objetivo obtido pela heurística e f_{best} corresponde à melhor solução encontrada a partir da execução da heurística para todos os valores do parâmetro.

Para validar e comprovar a diferença estatisticamente significativa entre os resultados, foi utilizada a Análise de Variância (ANOVA) paramétrica. Os testes apontam que há diferenças significativas para o uso dos diferentes valores de α e é possível observar que para $\alpha = 60\%$, obtém-se a menor média. Portanto, nos próximos experimentos, a heurística $HC7$ será executada usando $\alpha = 60\%$. A Figura 3 mostra o gráfico de intervalos HSD de *Tukey* com nível de confiança de 95% para a comparação da heurística $HC7$ com os diferentes valor para α.

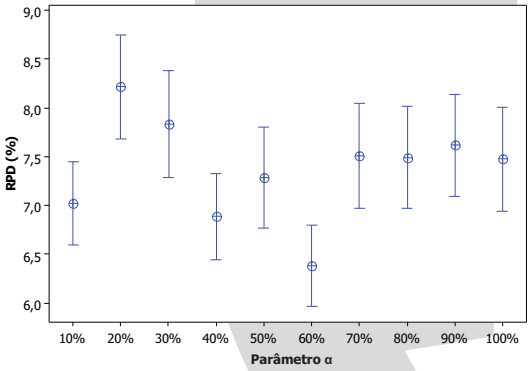


Figura 3: Gráfico de Intervalos HSD de *Tukey* para Calibração do Parâmetro α.

4.2. Experimento 1: Comparação entre os modelos matemáticos

Para este experimento, foi utilizado o conjunto de instâncias pequenas e para resolver os modelos matemáticos foi utilizado o software CPLEX 12.4 com limite máximo de 3600 segundos

de tempo de CPU para cada instância. No total, foram testadas 720 instâncias, onde o $Modelo^1$ conseguiu encontrar 718 soluções ótimas e 2 viáveis, já o $Modelo^2$ conseguiu encontrar 518 soluções ótimas e 202 viáveis. O $Modelo^2$ não conseguiu encontrar soluções melhores que o $Modelo^1$. Além disso, foi observado que o $Modelo^1$ consegue encontrar soluções viáveis dentro do tempo estipulado para instâncias de até 30 tarefas e o $Modelo^2$ consegue encontrar para instâncias de até 12 tarefas. Na Tabela 1 é apresentada a comparação entre os dois modelos. Como pode ser observado, o $Modelo^1$ obteve menores médias de *makespan*. No GAP entre o *Lower Bound* obtido e a solução encontrada pelos modelos para cada instância (GAP_{LB}), o $Modelo^1$ obteve média total de 0,01% contra 18,71% do $Modelo^2$. Para os grupos de instâncias onde o GAP_{LB} é igual a 0%, significa que todas as soluções encontradas no grupo são ótimas. E por fim, o tempo de CPU gasto pelos dois modelos são: 17,42 segundos para o $Modelo^1$ e 1.285,83 para o $Modelo^2$. Com esses resultados, é possível concluir que o $Modelo^1$ é significativamente melhor que o $Modelo^2$.

Tabela 1: Comparação entre o $Modelo^1$ e $Modelo^2$.

		$Modelo^1$			$Modelo^2$		
n	m	Média C_{max}	GAP_{LB}	Tempo	Média C_{max}	GAP_{LB}	Tempo
6	2	**207,53**	**0,00**	**0,10**	**207,53**	**0,00**	0,39
6	3	**159,02**	**0,00**	**0,11**	**159,02**	**0,00**	0,62
6	4	**135,73**	**0,00**	**0,11**	**135,73**	**0,00**	1,13
6	5	**113,60**	**0,00**	**0,13**	**113,60**	**0,00**	1,71
8	2	**259,29**	**0,00**	**0,32**	**259,29**	**0,00**	13,65
8	3	**184,42**	**0,00**	**0,25**	**184,42**	**0,00**	26,99
8	4	**164,36**	**0,00**	**0,20**	**164,36**	**0,00**	44,20
8	5	**128,71**	**0,00**	**0,19**	**128,71**	**0,00**	57,23
10	2	**308,09**	**0,00**	**15,38**	**308,09**	**0,00**	839,49
10	3	**225,44**	**0,00**	**0,88**	**225,44**	**0,00**	1101,99
10	4	**192,56**	**0,00**	**0,61**	192,67	2,25	1907,50
10	5	**168,13**	**0,00**	**0,49**	169,13	12,27	2206,41
12	2	**349,67**	**0,16**	**247,76**	383,31	70,13	3600,00
12	3	**254,82**	**0,00**	**6,12**	300,93	73,86	3600,00
12	4	**213,49**	**0,00**	**3,57**	248,87	67,85	3568,58
12	5	**181,87**	**0,00**	**2,46**	220,82	73,08	3600,00
	Médias:	**202,92**	**0,01**	**17,42**	212,62	18,71	1285,83

4.3. Experimento 2: Comparação entre as heurísticas propostas

Neste experimento é analisado o desempenho das heurísticas propostas de $HC1$ a $HC7$. O experimento utilizou o conjunto de instâncias grandes e o RPD é calculado considerando a melhor solução obtida dentre todas as sete heurísticas. Para validar os resultados obtidos e verificar se há diferenças estatisticamente significativas, foi realizada uma Análise de Variância (ANOVA) paramétrica. Também foram verificadas as três pressuposições da ANOVA: normalidade, igualdade de variância e independência dos resíduos. O resultado da ANOVA apontou diferença significativa entre as sete heurísticas. Na Figura 4 pode ser observado o gráfico de intervalos HSD de *Tukey* com nível de confiança de 95%, onde a heurística $HC7$ obteve a menor média de RPD.

4.4. Experimento 3: Comparação entre a heurística $HC7$ e o $Modelo^1$

Neste experimento é comparado o desempenho do $Modelo^1$, da heurística $HC7$ e das soluções geradas aleatoriamente. Para isso, é utilizado o conjunto de instâncias de tamanho médio e o modelo matemático é resolvido utilizando CPLEX com limite máximo de 3600 segundos de tempo de CPU. Além disso, para cada instância, foram geradas 50 soluções aleatórias e foi selecionada a melhor solução aleatória para ser utilizada no experimento.

A Tabela 2 é dividida em 3 partes. Na primeira parte, a média da função objetivo (C_{max}) do modelo, da heurística e das soluções aleatórias para os 12 grupos de instâncias são comparadas. Foi possível verificar que $HC7$ não encontrou soluções iguais ao modelo, mas obteve soluções mais próximas ao modelo do que as soluções aleatórias. Na segunda parte, o GAP_{LB} do modelo foi em

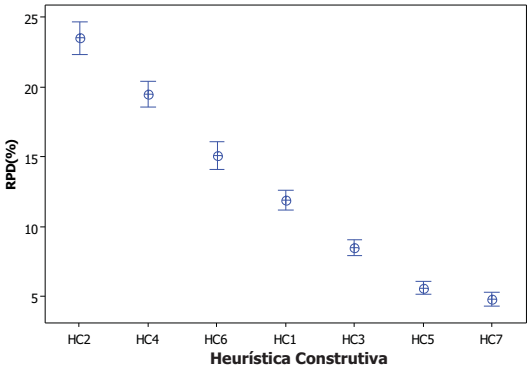


Figura 4: Gráfico de Intervalos HSD de *Tukey* para Comparação entre as Heurísticas.

média 18,70% e para três grupos de instâncias obteve média de 0%, que significa que todas as soluções encontradas para as instâncias desses grupos são ótimas. O $Modelo^1$ conseguiu encontrar 291 soluções ótimas do total de 540 instâncias. Além disso, o tempo de CPU gasto pelo modelo foi em média de 1863,43 segundos e também é possível observar que o modelo gastou mais tempo para grupos de instâncias com menos máquinas. Na última parte, o RPD da heurística e das soluções aleatórias são comparadas, onde pode ser observado que $HC7$ obteve RPD médio total de 28,10% contra 172,51% das soluções aleatórias. Vale destacar também a diferença do tempo de execução: a heurística $HC7$ executa cada instância em média $0,001s$ contra $1.863,43s$ gastos pelo $Modelo^1$.

Tabela 2: Comparação entre o $Modelo^1$, $HC7$ e Soluções Aleatórias.

		Média C_{max}			$Modelo^1$		Média RPD (%)	
n	m	$Modelo^1$	$HC7$	Aleatório	GAP_{LB}	Tempo	$HC7$	Aleatório
15	2	**426,47**	556,13	670,42	10,92	1533,19	**30,79**	58,68
15	5	**200,93**	267,67	468,89	**0,00**	14,61	**30,88**	145,61
15	10	**142,42**	186,60	412,80	**0,00**	17,41	**29,09**	214,67
20	2	**578,02**	713,71	952,82	28,82	2803,95	**22,95**	69,05
20	5	**241,78**	352,20	647,76	8,00	1038,40	**33,28**	182,46
20	10	**174,51**	229,49	556,36	**0,00**	125,21	**30,04**	245,42
25	2	**729,69**	897,73	1.256,62	36,84	3220,53	**22,68**	76,48
25	5	**297,16**	399,00	865,16	18,64	2727,93	**33,18**	209,26
25	10	**177,87**	254,80	687,38	3,94	1135,93	**41,28**	314,58
30	2	**827,07**	961,09	1.468,07	49,40	3600,00	**15,99**	78,68
30	5	**380,09**	455,89	1.028,47	41,50	3462,62	**18,22**	183,55
30	10	**239,60**	313,00	877,33	26,35	2681,18	**28,83**	291,54
Médias		**367,97**	463,36	824,34	18,70	1863,43	**28,10**	172,51

5. Conclusões

Neste trabalho foi abordado o problema de sequenciamento de tarefas em máquinas paralelas não-relacionadas considerando restrição de precedência entre as tarefas e tempos de preparação dependentes da sequência com o objetivo de minimizar o *makespan*. Para isso, primeiramente foram propostos dois diferentes modelos matemáticos para o problema. Depois, foram propostas sete heurísticas construtivas baseadas em diferentes regras de prioridade. Experimentos computacionais foram conduzidos para comparar o desempenho dos modelos matemáticos e das heurísticas propostas. No primeiro experimento, os testes mostraram que o $Modelo^1$ encontrou melhores resultados e com tempo computacional inferior ao $Modelo^2$. No segundo experimento, foi comparado o comportamento das heurísticas e os resultados apontaram que a heurística $HC7$ obteve melhor desempenho que as demais heurísticas. Por fim, foi comparada a heurística $HC7$ em relação ao $Modelo^1$, onde foi observado que a heurística obteve resultados significativamente melhores que soluções geradas aleatoriamente e possui um tempo computacional próximo de zero. Como trabalhos futuros, sugere-se a aplicação de meta-heurísticas para o problema abordado, onde as soluções

de partida sejam geradas utilizando as heurísticas construtivas propostas neste trabalho.

Agradecimentos
Os autores agradecem o financiamento do CNPq, FAPEMIG e CAPES.

Referências

Aho, I., e Mäkinen, E. (2006). On a parallel machine scheduling problem with precedence constraints. *Journal of Scheduling*, 9(5), 493-495.

Arroyo, J. E. C., e Armentano, V. A. (2004). A partial enumeration heuristic for multi-objective flowshop scheduling problems. *Journal of the Operational Research Society*, 55(9), 1000-1007.

Brucker, P., Hurink, J., e Kubiak, W. (1999). Scheduling identical jobs with chain precedence constraints on two uniform machines. *Mathematical Methods of Operations Research*, 49(2), 211-219.

Driessel, R., e Mönch, L. (2011). Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times. *Computers & Industrial Engineering*, 61(2), 336-345.

Gacias, B., Artigues, C., e Lopez, P. (2010). Parallel machine scheduling with precedence constraints and setup times. *Computers & Operations Research*, 37(12), 2141-2151.

Graham, R. L., Lawler, E. L., Lenstra, J. K., e Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5, 287-326.

Hassan Abdel-Jabbar, M. A., Kacem, I., e Martin, S. (2014, October). Unrelated parallel machines with precedence constraints: application to cloud computing. In Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on (pp. 438-442).

Herrmann, J., Proth, J. M., e Sauer, N. (1997). Heuristics for unrelated machine scheduling with precedence constraints. *European Journal of Operational Research*, 102(3), 528-537.

Johnson, D. S., e Garey, M. R. (1979). Computers and intractability: A guide to the theory of np-completeness. *Freeman&Co, San Francisco*, 31.

Kim, E. S. (2011). Scheduling of uniform parallel machines with s-precedence constraints. *Mathematical and Computer Modelling*, 54(1), 576-583.

Kumar, V. A., Marathe, M. V., Parthasarathy, S., e Srinivasan, A. (2009). Scheduling on unrelated machines under tree-like precedence constraints. *Algorithmica*, 55(1), 205-226.

Lin, Y. K., Pfund, M. E., e Fowler, J. W. (2011). Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. *Computers & Operations Research*, 38(6), 901-916.

Liu, C. (2013). A hybrid genetic algorithm to minimize total tardiness for unrelated parallel machine scheduling with precedence constraints. *Mathematical Problems in Engineering*, 2013.

Liu, C., e Yang, S. (2011). A heuristic serial schedule algorithm for unrelated parallel machine scheduling with precedence constraints. *Journal of Software*, 6(6), 1146-1153.

Rabadi, G., Moraga, R. J., e Al-Salem, A. (2006). Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing*, 17(1), 85-97.

Ramachandra, G., e Elmaghraby, S. E. (2006). Sequencing precedence-related jobs on two machines to minimize the weighted completion time. *International Journal of Production Economics*, 100(1), 44-58.

Vallada, E., e Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 211(3), 612-622.

Van Zuylen, A. (2011). An improved monotone algorithm for scheduling related machines with precedence constraints. *Operations Research Letters*, 39(6), 423-427.

Weng, M. X., Lu, J., e Ren, H. (2001). Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *International journal of production economics*, 70(3), 215-226.