PRODUÇÃO ONLINE
REVISTA CIENTÍFICA ELETRÔNICA DE ENGENHARIA DE PRODUÇÃO
ISSN 1676-1901

# UMA HEURÍSTICA ROBUSTA PARA PROGRAMAÇÃO DE MÁQUINAS PARALELAS COM TEMPOS DE *SETUP* DEPENDENTES DA SEQUÊNCIA

# A ROBUST HEURISTIC FOR PARALLEL MACHINES SCHEDULING PROBLEM WITH SEQUENCE-DEPENDENT SETUP TIMES

Leandro Resende Mundim* E-mail: leandroresendemundim@gmail.com
Helio Yochihiro Fuchigami** E-mail: heliofuchigami@yahoo.com.br
* Universidade de São Paulo/ICMC (USP/ICMC), São Paulo, SP
**Universidade Federal de Goiás / Faculdade de Ciências e Tecnologia (UFGO/FCT), Goiás, GO

**Resumo:** O propósito deste trabalho foi elaborar uma heurística robusta e eficiente para o problema de programação de um conjunto de *n* tarefas com tempos de *setup* explícitos e dependentes da sequência em um conjunto de *m* máquinas de forma a minimizar o *makespan* (duração total da programação). Os resultados foram testados com instâncias artificiais baseadas em trabalhos publicados na literatura específica e comparados com um limitante inferior (*lower bound*) proposto a partir de um modelo de programação linear inteira mista. A qualidade da solução heurística foi medida com base no desvio percentual em relação ao limitante inferior. A experimentação demonstrou tanto a eficácia da solução, com desvio relativo médio de 5,51%, como a eficiência computacional da heurística, com tempos de execução desprezível.

**Palavras-chave**: Programação da Produção. Máquinas paralelas. *Setup* dependente da sequência. Heurística.

**Abstract:** This paper addresses the problem of minimizing makespan on *m*-parallel machine with setup times separated of processing times of jobs and sequence-dependent. A robust and efficient heuristic was proposed and computationally implemented. Results were tested by instances generated based on works of the specific literature and compared to a model of mixed integer linear programming proposed as a lower bound for the makespan. The quality of the heuristic solution was measured by percentage deviation of lower bound. Experiments demonstrated both the solution's efficacy, with average relative deviation of 5.51%, and computational efficiency of the heuristic, with negligible execution times.

**Keywords**: Scheduling. Parallel machines. Sequence-dependent setup times. Heuristic.

## 1 INTRODUÇÃO

De acordo com Baker (1995, p.2), a programação da produção ou *scheduling* é a alocação de recursos escassos para a execução de tarefas em uma base de tempo. Os recursos podem ser exemplificados por máquinas em fábricas, pistas de um aeroporto, trabalhadores em construções, unidades de processamento em ambiente computacional; e as tarefas podem ser pedidos de produção (ordens de serviço ou de produção), decolagens e aterrissagens de aviões, estágios em um projeto de construção ou execuções de programas computacionais.

O *scheduling* pode ser definido como o processo de tomada de decisão presente tanto em sistemas de produção como em ambientes de processamento de informações, além das empresas de transporte, distribuição e outros tipos de serviços industriais (PINEDO, 2016).

O termo “máquinas paralelas” é utilizado quando existe um conjunto de recursos ou máquinas que podem processar quaisquer atividades ou tarefas. Nesta configuração, mais de uma máquina estão disponíveis para executar as tarefas que deverão ser processadas. Existem três variações deste problema: máquinas paralelas idênticas, proporcionais e não relacionadas.

As máquinas paralelas são denominadas idênticas quando uma mesma tarefa teria tempos de processamentos iguais em quaisquer das máquinas em que fosse processada. Um exemplo prático disto acontece quando se decide melhorar o fluxo em uma linha de produção e, após os estudos, identificam-se os gargalos da linha, onde é possível multiplicar a quantidade de máquinas operando em paralelo naquela etapa. Isto acelera o processo, fazendo com que os gargalos desapareçam ou sejam minimizados.

Denominam-se máquinas paralelas uniformes ou proporcionais quando o tempo para processar determinada tarefa depende da máquina utilizada. O tempo de processamento das tarefas nas máquinas é proporcional, ou seja, não são os mesmos mas existe um coeficiente de proporcionalidade entre os tempos das tarefas a serem processadas e as máquinas. Já nas máquinas não relacionadas o tempo de processamento depende tanto da tarefa como da máquina.

Em geral, problemas reais exigem um tempo de preparação ou *setup* entre a execução das tarefas. Muitas pesquisas em programação da produção desconsideram estes tempos ou então os incluem no tempo de processamento de cada tarefa. Isto simplifica a análise das aplicações, porém afeta diretamente a qualidade da solução para muitas situações que requerem o tratamento explícito do *setup* (ALLAHVERDI *et al.*, 1999).

Os trabalhos que consideram os tempos de *setup* explícitos, ou seja, separados do tempo de processamento das tarefas são divididos em dois grupos: com *setup* independente da sequência e com *setup* dependente da sequência. Nos primeiros, o tempo de *setup* depende apenas da tarefa a ser processada. Já no segundo grupo de problemas, a duração do *setup* depende tanto da tarefa a ser

processada quanto daquela que foi processada imediatamente antes na mesma máquina.

Alguns exemplos de aplicações com presença de tempos de *setup* dependentes da sequência são as indústrias de tinta e farmacêutica, em que os processos de limpeza e esterilização devem ser diferenciados dependendo da tarefa que foi feita e daquela que será processada em seguida. Processos que requerem ajuste de temperatura também requerem tempos de preparação diferentes dependendo da sequência de tarefas.

Neste trabalho foi estudado o problema de programação em máquinas paralelas com tempos de *setup* explícitos e dependentes da sequência de tarefas. No problema abordado, considera-se que existem *n* tarefas a serem processadas em *m* máquinas paralelas idênticas. Cada tarefa deve ser processada exatamente uma vez e, cada máquina pode processar uma tarefa por vez. O objetivo é minimizar a duração total da programação (*makespan*).

Este problema é representado pela conhecida notação de três campos da seguinte forma: $Pm|s_{ij}|C_{\max}$, onde "*Pm*" se refere ao ambiente com *m* máquinas paralelas idênticas, "$s_{ij}$" representa a restrição de *setup* dependente da sequência entre cada par de tarefas $J_i$ e $J_j$ e $C_{\max}$ é a maior dentre as datas de término das tarefas ou a duração total da programação.

Optou-se por este problema devido a sua importância, tanto teórica como prática. Do ponto de vista teórico, é uma generalização de máquina única, um estágio do *flexible flow shop* ou *flow shop* híbrido e é um problema de difícil solução por pertencer à classe *NP-hard*. A demonstração para o caso mais simples de minimização do *makespan* em máquinas paralelas pode ser encontrada em Pinedo (2016). Sveltana, Kravchenko e Werner (2001) e Nait *et al.* (2003) também provaram que o problema abordado é *NP-hard*.

Do ponto de vista prático, o estudo deste problema é importante devido a frequente ocorrência de recursos em paralelo no mundo real (PINEDO, 2016), sendo encontrada em indústrias de manufatura, produção, automobilística, entre outras. Behnamian, Zandieh e Fatemi Ghomi (2009) citam várias outros exemplos, como indústria do vidro, metalúrgica, química, têxtil, madeireira e aeroespacial.

Este artigo está estruturado da seguinte forma: após a contextualização da seção 1, é apresentada na seção 2 a revisão da literatura relacionada ao problema tratado; a descrição do funcionamento da heurística proposta é feita na seção 3 e a

definição do limitante inferior desenvolvido é apresentada na seção 4; os testes computacionais e a análise dos resultados é feita na seção 5 e as principais contribuições da pesquisa são salientadas nas conclusões (na seção 6).

## 2 REVISÃO BIBLIOGRÁFICA

O artigo pioneiro de máquinas paralelas idênticas com objetivo de minimizar o *makespan* foi publicado por McNaughton (1959), que considerou o problema que não permite interrupção de tarefas (*preemption*) e propôs limitantes inferiores usados em inúmeros outros trabalhos posteriores, como por exemplo Hu (1961) e Muntz e Coffman (1969). Estes últimos, por sua vez, propuseram o algoritmo MULTI-FIT, que serviu também como base para diversas outras pesquisas, como Friesen (1984), Lee e Massey (1988), Tang (1990), Rajgopal e Bidanda (1991) e Kurz e Askin (2001).

Cheng e Sin (1990) publicaram uma revisão do estado da arte para aquele momento dos principais resultados de pesquisas com máquinas paralelas. Mokotoff (2001) apresentou uma revisão com ênfase em métodos que fornecem o *makespan* ótimo para máquinas paralelas idênticas.

Nos últimos anos, vários trabalhos foram publicados com os diferentes tipos de máquinas paralelas. A maioria considerou máquinas não relacionadas, como no caso de Lin, Lu, Ying (2011), Ruiz e Andrés-Romano (2011), Vallada e Ruiz (2011), Kuo, Hsu e Yang (2011), Chang e Chen (2011), Bozorgirad e Logendran (2012) e Yang e Yang (2013).

Muitos estudos abordaram o problema de máquinas paralelas idênticas, a exemplo de Monma e Potts (1993), Ovacik e Uzhoy (1993), Damodaran e Chang (2008), Montoya-Torres, Soto-Ferrari e González-Solano (2010), Behnamian, Zandieh e Fatemi Ghomi (2011), Turker e Sel (2011) e Behnamian e Fatemi Ghomi (2013). E o problema de máquinas paralelas uniformes foi analisado por Li, Yang e Ma (2011), Mora e Mosheiov (2012), Lee, Chuang e Yeh (2012) e Lin (2013).

Estudos de casos abordando máquinas paralelas foram feitos por Van Hop e Nagarur (2004), que propuseram métodos de programação da produção de placas de circuito impresso, e Santos e Vilarinho (2010), para o planejamento operacional de uma indústria têxtil. O trabalho de Weng, Lu e Ren (2001) proveio do estudo de uma empresa de serviços – o problema de reparos de falhas em praças de pedágio.

Behnamian, Zandieh e Fatemi Ghomi (2009) propuseram uma meta-heurística híbrida para minimização do *makespan* em problemas com *setup* dependente, baseada em otimização por colônia de formiga, *simulated annealing* e busca em vizinhança variável, comprovando sua eficácia em relação a outros métodos publicados na literatura. Este trabalho foi utilizado como base para o delineamento da experimentação computacional da presente pesquisa.

Vários trabalhos usaram meta-heurísticas como técnicas de solução para o problema de máquinas paralelas, tais como Hou, Ansari e Ren (1994), Ming e Cheng (1998), Correa, Ferreira e Rebreyend (1999), Radhakrishnan e Ventura (2000), Kurz e Askin (2001), Gendreau, Laporte e Guimarães (2001), Kim *et al.* (2002), Mendes *et al*. (2002), Fowler, Horng e Cochran (2003), Kim e Shin (2003), Bilge *et al*. (2004), Chang, Chou e Lee (2004), Lee, Wu e Chen (2006), Kashan, Karimi e Jenabi (2008) e Laha (2012).

Especificamente em relação aos trabalhos considerando explicitamente os tempos de *setup*, Liaee e Emmons (1997) apresentaram uma classificação por critério de desempenho dos problemas de processamento de famílias de tarefas com tempos de *setup*. Allahverdi, Gupta e Aldowaisan (1999) fizeram uma revisão da literatura de problemas de programação da produção envolvendo tempos de *setup*. Os problemas foram classificados em *batch* e *non-batch*, e *setup* dependente e independente da sequência.

Zhu e Wilhelm (2006) publicaram uma revisão da literatura de diversas configurações de ambientes com tempos e custos de *setup* dependentes da sequência de execução das tarefas. E Allahverdi *et al.* (2008) atualizaram a revisão da literatura de problemas com tempos e custos de *setup*, classificando mais de 300 trabalhos publicados após o levantamento de Allahverdi, Gupta e Aldowaisan (1999).

Merecem destaque os trabalhos de Kurz e Askin (2001), Montoya-Torres *et al*. (2009) e Montoya-Torres, Soto-Ferrari e González-Solano (2010), que propuseram métodos heurísticos para minimização do *makespan* para o problema de máquinas paralelas idênticas com tempos de *setup* dependentes da sequência. Estes pesquisadores abordaram ambientes bastante similares ao tratado nesta pesquisa, porém com a diferença de que nas publicações citadas, o problema é dinâmico, ou seja, as tarefas não estão todas disponíveis na data zero da programação, mas possuem diferentes datas de liberação.

## 3 HEURÍSTICA PROPOSTA

Em geral problemas de programação da produção (*scheduling*) possuem um conjunto de *n* tarefas, denotado por $N = \{1,..., n\}$, e um conjunto de *m* máquinas, representado por $M = \{1,..., m\}$. O problema de máquinas paralelas idênticas abordado neste trabalho, além dos conjuntos de tarefas e máquinas deve assumir como verdade as hipóteses a seguir: (*i*) É conhecida a quantidade de máquinas; (*ii*) É conhecido o tempo de processamento e de *setup* das tarefas; (*iii*) Todas as tarefas possuem data de liberação igual a zero e estão disponíveis durante todo o processo; (*iv*) Todas as tarefas devem ser executadas, e por apenas uma máquina; (*v*) Uma máquina não pode executar mais que uma operação simultaneamente; (*vi*) Não são consideradas indisponibilidades temporárias devido a quebras ou manutenção das máquinas.

O objetivo do método de solução heurística encontrar uma programação que minimize o *makespan*. Para isto, é necessário estabelecer a alocação das tarefas nas máquinas e a sequência em que serão executadas. O *makespan* será definindo como maior dentre as datas de término $C_j$ das tarefas ($max\{C_1, C_2, C_3, ..., C_n\}$).

A heurística proposta neste trabalho é baseada no balanceamento de carga das máquinas, considerada como a quantidade de trabalho que elas têm para processar, ou seja, a soma dos tempos de processamento das tarefas designadas a cada máquina.

Será utilizada aqui uma matriz de custo, quadrada de ordem *n*, composta por “pesos” que são iguais ao tempo de processamento somado à média dos tempos de *setup* das tarefas e subtraído o tempo de *setup* atual (*setup* da célula correspondente na matriz dos tempos de *setup*). A ideia é priorizar as tarefas com maior peso, alocando-a na máquina de menor carga, penalizando as posições com os maiores tempos de *setup* por meio da subtração do *setup* atual.

Por meio desta lógica, as tarefas críticas, ou seja, de maiores pesos, são alocadas nas primeiras posições de cada máquina, deixando as tarefas de menor peso para o final da programação. Isto proporciona um melhor equilíbrio das cargas das máquinas e consequentemente minimizando o *makespan*.

O pseudocódigo da heurística proposta é apresentado a seguir.

**Algoritmo 1** Heurística

**Entrada:** *n*, *m*, *tarefas*, *setup*
*n* é o número de tarefas
*m* é o número de máquinas
*tarefas* é o vetor com o tempo de processamento das *n* tarefas
*setup* é uma matriz com o tempo de *setup* das *n* tarefas
**Saída:** *S*, *makespan*
*S* é a solução com a posição das *n* tarefas nas *m* máquinas
*makespan* é a duração total da programação

```
 1: begin
 2: media_setup é um vetor que recebe a média do setup das n tarefas
 3: for (i de 1 até n) do
 4:     aux = 0;
 5:     for (j de 1 até n) do
 6:         aux = aux + setup[i][j];
 7:     end for
 8:     media_setup[i] = aux/n;
 9: end for
10: custo é uma matriz n x n
11: for (i de 1 até n) do
12:     for (j de 1 até n) do
13:         custo[j][i] = tarefas[i] + media_setup[j] – setup[j][i];
14:     end for
15: end for
16: for (i de 1 até n) do
17: busque_maior é uma função que busca a tarefa de maior custo, retira da lista de tarefas e salva em job
18: aloca_tarefa é uma função que aloca a tarefa salva em job na máquina de menor carga e atualiza S
19: end for
20: Chama a função calcula_makespan
21: Retorna a solução S e o makespan
22: end
```

Como pode ser visto no Algoritmo 1, a heurística recebe como parâmetros de entrada o número de tarefas (*n*), o número de máquinas (*m*), um vetor com os tempos de processamento das tarefas (denominado *tarefas*) e a matriz dos tempos de *setup* dependentes da sequência (definida como *setup*), que é quadrada de ordem *n*. Os elementos da diagonal principal (com índices $i = j$) representam os tempos de *setup* quando a tarefa *j* é alocada na primeira posição da máquina. Os demais elementos ($i \neq j$) se referem aos tempos de *setup* requeridos após a execução da tarefa *i* e antes da tarefa *j*.

Na linha 2 do Algoritmo 1, declara-se o vetor *media_setup*, que receberá a média dos tempos de *setup* da tarefa *j* considerando todas as *n*–1 possibilidades de tarefas *i* antecessoras a *j* e também o caso em que a tarefa *j* é a primeira da máquina (totalizando *n* possibilidades). A ordem das tarefas neste vetor é a mesma do vetor *tarefas*. Entre as linhas 3:9 o vetor *media_setup* recebe o seu conteúdo.

Na linha 10, é declarada a matriz *custo* quadrada de ordem *n*, que constitui o ponto principal deste algoritmo, pois contém os custos ou pesos utilizados na tomada de decisão da alocação das tarefas nas máquinas. Ela segue o padrão da matriz *setup*, onde a posição *ij* da matriz representa o custo de se alocar a tarefa *j* depois da *i*, quando $i \neq j$, e o custo de se alocar a tarefa *j* na primeira posição, quando $i = j$.

Entre as linhas 11:15, a matriz recebe os custos calculados na linha 13. A ideia deste custo é dar um peso elevado às tarefas com maior tempo de processamento e com menor tempo de *setup*. No laço das linhas 16:19, a heurística aloca sequencialmente as tarefas com maior custo sempre na máquina com menor carga. Por fim, na linha 21 o algoritmo retorna a programação das tarefas nas máquinas e o *makespan* calculado na linha 20.

## 4 LIMITANTE INFERIOR PROPOSTO

Nesta seção será apresentado o método desenvolvido para estimar a qualidade da heurística proposta. Para isso, foi desenvolvido um limitante inferior (*lower bound*) por meio de uma relaxação do problema estudado.

Problemas de otimização geralmente podem ser modelados matematicamente através de programação linear (PL), programação linear inteira (PLI) ou programação linear inteira mista (PLIM). Apesar dos modelos sempre encontrarem a solução ótima, problemas mais elaborados como o abordado neste trabalho possuem um custo computacional inviável. Isto acontece porque instâncias de médio e grande porte podem levar até anos para serem resolvidas.

O modelo matemático proposto é o mesmo do problema clássico de minimização do *makespan* em máquinas paralelas. A diferença deste problema com o abordado neste trabalho está na consideração dos tempos de *setup*, pois no problema clássico estão incluídos no tempo de processamento das tarefas.

Como o tempo de processamento das tarefas é o mesmo em qualquer máquina ou sequencia de tarefas, o ideal para o problema aqui apresentado seria que o tempo de *setup* das tarefas fossem sempre os menores tempos de *setup* para cada tarefa. Com base nesta ideia, a relaxação do problema original considera a soma do menor tempo de *setup* e o tempo de processamento das tarefas.

O modelo a seguir apresenta uma formulação de PLIM que combina uma variável contínua (*makespan*) com variáveis binárias ($x_{ik}$). A variável $C_{max}$ representa

o *makespan* e as variáveis de decisão $x_{ik}$ recebem valor 1 se a tarefa *i* for alocada na maquina *k* e valor zero, caso contrário. A variável auxiliar $P_i$ contém a soma do menor tempo de *setup* da tarefa *i* o seu tempo de processamento.

Minimizar $C_{\max}$ (1)

Sujeito a

$$\sum_{k=1}^{m} x_{ik} = 1, \qquad i = 1, ..., n, \tag{2}$$

$$C_{\max} - \sum_{i=1}^{n} P_i x_{ik} \geq 0, \qquad k = 1, ..., m, \tag{3}$$

$$C_{\max} \geq 0 \tag{4}$$

$$x_{ik} \in \{0,1\}. \tag{5}$$

A função objetivo do modelo (1) visa minimizar o *makespan*. Já o conjunto de restrições (2) impõe que cada tarefa deve ser alocada apenas uma vez em exatamente uma máquina. O segundo conjunto de restrições (3) define o valor do *makespan*, garantindo que seja o maior valor dos somatórios dos tempos de processamento e do menor *setup* de cada tarefa, dentre todas as máquinas. As restrições (4) e (5) definem o domínio das variáveis, sendo de não negatividade para o $C_{\max}$ e de valores binários (zero ou um) para $x_{ik}$.

O valor ótimo da função objetivo deste modelo será usado como limitante inferior (*LB*) do *makespan* para o problema original e comparado com a solução da heurística proposta por meio do desvio percentual:

$$Desvio(\%) = \frac{C_{\max}^{heur} - LB}{LB} \tag{6}$$

## 5 TESTES COMPUTACIONAIS E RESULTADOS

Na classificação metodológica da pesquisa, esta investigação pode ser definida como *abordagem quantitativa*, pois há preocupação com mensurabilidade, causalidade, generalização e replicação. Além disso, pode ser classificada como *pesquisa aplicada* quanto à natureza, por gerar conhecimento com finalidades de aplicação prática, *pesquisa exploratória* quanto aos objetivos, pois visa melhoria teórico-prática de sistemas, processos e produtos, e inovação pela proposição de novos modelos, além de ser feita a partir de impulsos criativos, simulações e

experimentações, podendo originar novos modelos destinados a invenções, inovações e a otimização, e *pesquisa experimental* quanto aos procedimentos, para a obtenção de novos conhecimentos e produtos tecnológicos, requerendo uma manipulação de variáveis detalhada e sistemática, e originando inovações a partir de ensaios e estudos dinâmicos em laboratório.

A heurística proposta neste trabalho foi codificada na linguagem C/C++. Os testes computacionais ocorreram em um computador com processador Intel® Core™ 2 Quad 2,4 GHz, 4 GB de memória RAM e sistema operacional *Linux*.

Para resolver o modelo utilizou-se o software ILOG® CPLEX® 12 Callable Library. Este resolvedor é um software robusto da IBM e possui pacotes que gerenciam o processo de otimização automaticamente utilizando algoritmos como *branch-and-cut* e *branch-and-bound*, o que justifica a sua escolha.

Como não foram encontradas instâncias públicas para o problema de máquinas paralelas idênticas com *setup* dependente da sequência, geraram-se instâncias aleatórias utilizando a ideia do trabalho Behnamian, Zandieh e Fatemi Ghomi (2009), que formalizou os testes com um conjunto de níveis distribuídos em grupos, onde o número de tarefas foi de {6, 30, 100} e o número de máquinas foi de {1, 2, 10}. Já os tempos de *setup* foram obtidos de forma aleatória variando de 20 a 40 por cento do tempo máximo de processamento das tarefas.

Para fazer uma análise mais completa, os parâmetros deste trabalho são definidos de forma a considerar todos os parâmetros de Behnamian, Zandieh e Fatemi Ghomi (2009) e também outros pertinentes. Isto é feito a partir dos conjuntos definidos na Tabela 1. Os parâmetros em negrito são os originais de Behnamian, Zandieh e Fatemi Ghomi (2009).

Nesta tabela, pode-se verificar o número de tarefas e de máquinas, o intervalo de tempos de processamento considerado e também os três intervalos de tempos de *setup* utilizados. Estes intervalos são os limites para os tempos gerados de forma aleatória com distribuição uniforme.

**Tabela 1** – Parâmetros dos problemas resolvidos

| **Parâmetros** | **Valores** |
|---|---|
| Número de tarefas | 3, 6, 30, 60, 100 |
| Número de máquinas | 1, 2, 4, 10 |
| Tempos de processamento | 1-100 |
| Intervalos de tempos de *setup* | 0-10, 20-40, 60-80 |

**Fonte:** Autores

Definidos os parâmetros utilizados, os resultados dos testes computacionais são apresentados nas Tabelas 2, 3 e 4 a seguir, separados por opções de intervalos de tempos de *setup*. As tabelas apresentam para cada problema-teste resolvido o porte do problema (número de tarefas e de máquinas), os *makepans* da heurística e do modelo, os tempos de execução requeridos na resolução e o desvio relativo percentual.

Como pode ser visto na Tabela 2, que contém os resultados para problemas com tempos de *setup* no intervalo de 0-10, o desvio percentual médio da solução heurística em relação ao limitante inferior foi de 4,72.

Como esperado, o modelo PLIM demandou maior tempo computacional na solução dos problemas maiores (com 10 máquinas), enquanto o método heurístico levou um tempo de execução de praticamente zero (segundo).

**Tabela 2** – Resultados com tempos de *setup* entre 0 e 10

| Tarefas | Máquinas | Heurística | | Modelo PLIM | | Desvio (%) |
|---|---|---|---|---|---|---|
| | | ***Makespan*** | **Tempo (s)** | ***Makespan*** | **Tempo (s)** | |
| 3 | 1 | 179 | 0,00 | 171 | 0,00 | 4,68 |
| | 2 | 89 | 0,00 | 87 | 0,00 | 2,30 |
| 6 | 1 | 342 | 0,00 | 308 | 0,00 | 11,04 |
| | 2 | 176 | 0,00 | 159 | 0,00 | 10,69 |
| | 4 | 92 | 0,00 | 88 | 0,00 | 4,55 |
| 30 | 1 | 1599 | 0,00 | 1470 | 0,00 | 8,78 |
| | 2 | 1001 | 0,00 | 956 | 0,00 | 4,71 |
| | 4 | 354 | 0,00 | 321 | 0,00 | 10,28 |
| | 10 | 177 | 0,00 | 168 | 30,38 | 5,36 |
| 60 | 1 | 3232 | 0,00 | 3045 | 0,00 | 6,14 |
| | 2 | 1652 | 0,00 | 1568 | 0,01 | 5,36 |
| | 4 | 844 | 0,00 | 804 | 0,12 | 4,98 |
| | 10 | 427 | 0,00 | 401 | 110,35 | 6,48 |
| 100 | 1 | 5407 | 0,00 | 5199 | 0,00 | 4,00 |
| | 2 | 2697 | 0,00 | 3600 | 0,00 | –25,08 |
| | 4 | 1455 | 0,00 | 1302 | 10,15 | 11,75 |
| | 10 | 751 | 0,00 | 720 | 203,81 | 4,31 |
| | | **Média** | | | | **4,72** |

**Fonte:** Autores

**Tabela 3** – Resultados com tempos de *setup* entre 20 e 40

| Tarefas | Máquinas | Heurística | | Modelo PLIM | | Desvio (%) |
|---|---|---|---|---|---|---|
| | | ***Makespan*** | **Tempo (s)** | ***Makespan*** | **Tempo (s)** | |
| 3 | 1 | 180 | 0,00 | 172 | 0,00 | 4,65 |
| | 2 | 89 | 0,00 | 87 | 0,00 | 2,30 |
| 6 | 1 | 550 | 0,00 | 499 | 0,00 | 10,22 |
| | 2 | 267 | 0,00 | 253 | 0,00 | 5,53 |
| | 4 | 161 | 0,00 | 146 | 0,00 | 10,27 |
| 30 | 1 | 2069 | 0,00 | 1910 | 0,00 | 8,32 |
| | 2 | 1051 | 0,00 | 955 | 0,00 | 10,05 |
| | 4 | 530 | 0,00 | 478 | 0,00 | 10,88 |
| | 10 | 222 | 0,00 | 192 | 52,03 | 15,63 |
| 60 | 1 | 4514 | 0,00 | 4294 | 0,00 | 5,12 |
| | 2 | 2282 | 0,00 | 2147 | 0,01 | 6,29 |
| | 4 | 1148 | 0,00 | 1074 | 0,12 | 6,89 |
| | 10 | 466 | 0,00 | 430 | 60,86 | 8,37 |
| 100 | 1 | 7813 | 0,00 | 7457 | 0,00 | 4,77 |
| | 2 | 3895 | 0,00 | 3729 | 0,02 | 4,45 |
| | 4 | 1964 | 0,00 | 1865 | 0,13 | 5,31 |
| | 10 | 796 | 0,00 | 746 | 610,62 | 6,70 |
| **Média** | | | | | | **7,40** |

**Fonte:** Autores

Para os problemas com intervalo de tempos de *setup* de 20-40, conforme a Tabela 3, os resultados são consistentes com os anteriores, ficando o desvio médio da solução heurística em relação ao limitante inferior em 7,40%. Os tempos de execução da heurística se mantiveram em zero.

**Tabela 4** – Resultados com tempos de *setup* entre 60 e 80

| Tarefas | Máquinas | Heurística | | Modelo PLIM | | Desvio (%) |
|---|---|---|---|---|---|---|
| | | ***Makespan*** | **Tempo (s)** | ***Makespan*** | **Tempo (s)** | |
| 3 | 1 | 343 | 0,00 | 342 | 0,00 | 0,29 |
| | 2 | 219 | 0,00 | 212 | 0,00 | 3,30 |
| 6 | 1 | 767 | 0,00 | 758 | 0,00 | 1,19 |
| | 2 | 409 | 0,00 | 380 | 0,00 | 7,63 |
| | 4 | 233 | 0,00 | 226 | 0,00 | 3,10 |
| 30 | 1 | 3410 | 0,00 | 3275 | 0,00 | 4,12 |
| | 2 | 1707 | 0,00 | 1638 | 0,00 | 4,21 |
| | 4 | 918 | 0,00 | 819 | 0,26 | 12,09 |
| | 10 | 369 | 0,00 | 328 | 50,13 | 12,50 |
| 60 | 1 | 7267 | 0,00 | 7034 | 0,00 | 3,31 |
| | 2 | 3618 | 0,00 | 3517 | 0,00 | 2,87 |
| | 4 | 1833 | 0,00 | 1759 | 0,17 | 4,21 |
| | 10 | 746 | 0,00 | 705 | 304,04 | 5,82 |
| 100 | 1 | 11087 | 0,00 | 10938 | 0,00 | 1,36 |
| | 2 | 5545 | 0,00 | 5419 | 0,01 | 2,33 |
| | 4 | 2792 | 0,00 | 2710 | 0,34 | 3,03 |
| | 10 | 1123 | 0,00 | 1084 | 941,00 | 3,60 |
| **Média** | | | | | | **4,41** |

**Fonte:** Autores

Por fim, quando os tempos de *setup* ficaram na faixa de 60-80, como consta na Tabela 4, os resultados foram levemente melhores, com desvio médio da solução heurística em relação ao limitante inferior de 4,41% e também tempos de execução da heurística em zero segundo.

Numa análise por porte do problema, as Tabelas 5 e 6 apresentam os desvios relativos médios para cada opção do número de tarefas e de máquinas, respectivamente.

**Tabela 5** – Desvio relativo médio por número de tarefas

| Tarefas | Desvio Médio (%) |
|---|---|
| 3 | 2,92 |
| 6 | 7,14 |
| 30 | 8,91 |
| 60 | 5,49 |
| 100 | 2,21 |

**Fonte:** Autores

Como pode ser observado na Tabela 5, os menores desvios em relação ao limitante inferior ocorreram em problemas com 100 tarefas, com 2,21%, e os maiores, em problemas com 30 tarefas, atingindo 8,91%. Esta amplitude relativamente modesta revela certa estabilidade dos resultados da heurística com a variação do tamanho do problema, especificamente em relação ao número de tarefas.

**Tabela 5** – Desvio relativo médio por número de máquinas

| Máquinas | Desvio Médio (%) |
|---|---|
| 1 | 5,20 |
| 2 | 3,13 |
| 4 | 7,28 |
| 10 | 7,64 |

**Fonte:** Autores

Complementando a análise, a Tabela 6 mostra também a relativa estabilidade dos desvios quando considerados separadamente por opção do número de máquinas. Conforme se nota, os menores desvios em relação ao limitante inferior ocorreu em problemas com duas máquinas, com 3,13%, e os maiores desvios se deram em problemas com 10 máquinas, com 7,64%.

Conjuntamente em relação às Tabelas 5 e 6, pode-se inferir que o porte do

problema, tanto em relação ao número de tarefas como em relação à configuração do ambiente de produção (número de máquinas) não afetam o desempenho da heurística proposta.

Estes resultados, portanto, mostraram-se bastante promissores, indicando a aplicabilidade da heurística, dada sua eficácia na qualidade da solução e eficiência computacional.

## 6 CONCLUSÃO

Apresentou-se neste trabalho uma heurística robusta para o problema de minimização do *makespan* em ambientes com máquinas em paralelo e tempos de preparação dependentes da sequência, uma configuração bastante frequente em diversos sistemas industriais, tanto isoladamente como em versões híbridas em meio a processos de fluxo, ou ainda em empresas de serviços. Ocorrem em geral quando se incrementa a capacidade produtiva do posto de trabalho adicionando-se novos recursos, como máquinas, equipamentos ou funcionários.

O método de solução proposto se baseia no balanceamento de carga das máquinas, ou seja, na quantidade de trabalho que vai sendo alocada a cada recurso, considerando conjuntamente os tempos de processamento das tarefas e de preparação das máquinas. Foi também desenvolvida uma estratégia de comparação dos resultados da heurística, utilizando um limitante inferior que ateste a qualidade da solução obtida com base em uma relaxação de um modelo de programação linear inteira mista.

Os experimentos computacionais mostraram o pequeno desvio geral da solução heurística em relação ao limitante inferior de apenas 5,51% em média. Analisando o comportamento dos resultados gerados pelo software CPLEX, observa-se que para as maiores instâncias o tempo de solução do modelo é significativo, enquanto o da heurística permanece desprezível em todos os casos.

Deve-se destacar ainda que a experimentação computacional realizada, por abranger a opção de uma máquina, verifica também o desempenho da heurística proposta no problema de máquina única com *setup* dependente. O desvio médio para este caso em particular é de 5,20%.

De forma geral, a heurística apresentou resultados bem expressivos diante dos testes realizados e demonstra grande potencial para estudos futuros, pois pode facilmente ser adaptada para outros problemas em vários ambientes de produção. A

heurística pode ainda ser utilizada em meta-heurísticas para gerar uma solução inicial ou a partir de uma aleatoriedade gerar uma população inicial.

## AGRADECIMENTOS

A pesquisa relatada neste artigo teve o apoio do CNPq, da CAPES e da FAPEG.

## REFERÊNCIAS

ALLAHVERDI, A.; GUPTA; J.N.D.; ALDOWAISAN, T. A review of scheduling research involving setup considerations. **Omega - The International Journal of Management Science**, v. 27, p. 219-239, 1999.

ALLAHVERDI, A.; NG, C.T.; CHENG, T.C.E.; KOVALYOV, M.Y. A survey of scheduling problems with setup times or costs. **European Journal of Operational Research**, v. 187, p. 985-1032, 2008. https://doi.org/10.1016/j.ejor.2006.06.060

BAKER, K.R. **Elements of Sequencing and Scheduling**. Hanover: NH, 1995.

BEHNAMIAN, J.; FATEMI GHOMI, S.M.T. The heterogeneous multi-factory production network scheduling with adaptive communication policy and parallel machine. **Information Science**, v. 209, p. 181-196, 2013. https://doi.org/10.1016/j.ins.2012.07.020

BEHNAMIAN, J.; ZANDIEH, M.; FATEMI GHOMI, S.M.T. Bi-objective parallel machines scheduling with sequence-dependent setup times using hybrid metaheuristics and weighted min-max technique. **Soft Computing**, v. 15, n. 7, p. 1313-1331, 2011. https://doi.org/10.1007/s00500-010-0673-0

BEHNAMIAN, J.; ZANDIEH, M.; FATEMI GHOMI, S.M.T. Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm. **Expert Systems with Applications**, v. 36, p. 9637-9644, 2009. https://doi.org/10.1016/j.eswa.2008.10.007

BILGE, U.; KIRAÇ, F.; KURTULAN, M.; PEKGUN, P. A tabu search algorithm for parallel machine total tardiness problem. **Computers and Operations Research**, v. 31, p. 397-414, 2004. https://doi.org/10.1016/S0305-0548(02)00198-3

BOZORGIRAD, M.A.; LOGENDRAN, R. Sequence-dependent group scheduling problem on unrelated-parallel machines. **Expert Systems with Applications**, v. 39, p. 9021-9030, 2012. https://doi.org/10.1016/j.eswa.2012.02.032

CHANG, P.; CHEN, S. Integrating dominance properties with genetic algorithms for parallel machine scheduling problems with setup times. **Applied Soft Computing**, v. 11, n. 1, p. 1263-1274, 2011. https://doi.org/10.1016/j.asoc.2010.03.003

CHANG, T.-Y.; CHOU, F.-D.; LEE, C.-E.; A heuristic algorithm to minimize total weighted tardiness on a single machine with release dates and sequence-dependent setup times. **Journal of the Chinese Institute of Industrial Engineers**, v. 21, n. 3, p. 289-300, 2004. https://doi.org/10.1080/10170660409509410

CHENG, T.C.E.; SIN, C.C.S. A state-of-the-art review of parallel-machine scheduling research. **European Journal of Operational Research**, v. 47, n. 3, p. 271-292, 1990. https://doi.org/10.1016/0377-2217(90)90215-W

CORREA, R.C.; FERREIRA, F.; REBREYEND, P. Scheduling multiprocessor tasks with genetic algorithms. **IEEE Transactions on Parallel and Distributed Systems**, v. 10, n. 8, p. 825-837, 1999. https://doi.org/10.1109/71.790600

DAMODARAN, P.; CHANG, P.-Y. Heuristics to minimize makespan of parallel batch processing machines. **The International Journal of Advanced Manufacturing Technology**, v. 37, n. 9, p. 1005-1013, 2008. https://doi.org/10.1007/s00170-007-1042-8

FOWLER, J.W.; HORNG, N.G.; COCHRAN, J.K. A hybridized genetic algorithm to solve parallel machine scheduling problems with sequence dependent setups. **International Journal of Industrial Engineering – Theory Applications and Practice**, v. 10, n. 3, p. 232-243, 2003.

FRIESEN, D.K. Tighter bounds for the MULTIFIT processor scheduling algorithm. **SIAM Journal on Computing**, v. 12, n. 1, p. 170-181, 1984. https://doi.org/10.1137/0213013

GENDREAU, M.; LAPORTE, G.; GUIMARÃES, E.M. A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. **European Journal of Operational Research**, v. 133, p. 183-189, 2001. https://doi.org/10.1016/S0377-2217(00)00197-1

HOU, E.S.H.; ANSARI, N.; REN, H. A genetic algorithm for multiprocessor scheduling. **IEEE Transactions on Parallel and Distributed Systems**, v. 5, n. 2, p. 113-120, 1994. https://doi.org/10.1109/71.265940

HU, T.C. Parallel sequencing and assembly line problems. **Operations Research**, v. 9, n. 6, p. 941-848, 1961. https://doi.org/10.1287/opre.9.6.841

KASHAN, A.H.; KARIMI, B.; JENABI, M. A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes. **Computers & Operations Research**, v. 35, n. 4, p. 1084-1098, 2008. https://doi.org/10.1016/j.cor.2006.07.005

KIM, D.-W.; KIM, K.-H.; JANG, W.; CHEN, F.F. Unrelated parallel machine scheduling with setup times using simulated annealing. **Robotics and Computer Integrated Manufacturing**, v. 18, n. 3-4, p. 223-231, 2002. https://doi.org/10.1016/S0736-5845(02)00013-3

KIM, S.S.; SHIN, H.J. Scheduling jobs on parallel machines: a restricted tabu search approach. **International Journal of Advanced Manufacturing Technology**, v. 22, p. 278-287, 2003. https://doi.org/10.1007/s00170-002-1472-2

KUO, W.-H.; HSU, C.; YANG, D.-L. Some unrelated parallel machine scheduling problems with past-sequence-dependent setup time and learning effects. **Computers & Industrial Engineering**, v. 61, n. 1, p. 179-183, 2011. https://doi.org/10.1016/j.cie.2011.03.008

KURZ, M.E.; ASKIN, R.G. Heuristic scheduling of parallel machines with sequence-dependent set-up times. **International Journal of Production Research**, v. 39, n. 16, p. 3747-3769, 2001. https://doi.org/10.1080/00207540110064938

LAHA, D. A simulated annealing heuristic for minimizing makespan in parallel machine scheduling. **Swarm, Evolutionary, and Memetic Computing**, v. 7677, p. 198-205, 2012. https://doi.org/10.1007/978-3-642-35380-2_24

LEE, C.-Y.; MASSEY, J.D. Multiprocessor scheduling: combining LPT and MULTIFIT. **Discrete Applied Mathematics**, v. 20, n. 3, p. 233-242, 1988. https://doi.org/10.1016/0166-218X(88)90079-0

LEE, W.-C.; CHUANG, M.-C.; YEH, W.-C. Uniform parallel-machine scheduling to minimize makespan with position-based learning curves. **Computers & Industrial Engineering**, v. 63, p. 813-818, 2012. https://doi.org/10.1016/j.cie.2012.05.003

LEE, W.-C.; WU, C.-C.; CHEN, P. A simulated annealing approach to makespan minimization on identical parallel machines. **The International Journal of Advanced Manufacturing Technology**, v. 31, n. 3, p. 328-334, 2006. https://doi.org/10.1007/s00170-005-0188-5

LI, K.; YANG, S.-L.; MA, H.-W. A simulated annealing approach to minimize the maximum lateness on uniform parallel machines. **Mathematical and Computer Modelling**, v. 53, p. 854-860, 2011. https://doi.org/10.1016/j.mcm.2010.10.022

LIAEE, M.M.; EMMONS, H. Scheduling families of jobs with setup times. **International Journal of Production Economics**, Amsterdam, v. 51, n. 3, p. 165-176, 1997. https://doi.org/10.1016/S0925-5273(96)00105-3

LIN, S.-W.; LU, C.-C.; YING, K.-C. Minimization of total tardiness on unrelated parallel machines with sequence- and machine-dependent setup times under due date constraints. **The International Journal of Advanced Manufacturing Technology**, v. 53, p. 353-361, 2011. https://doi.org/10.1007/s00170-010-2824-y

LIN, Y.-K. Fast LP models and algorithms for identical jobs on uniform parallel machines. **Applied Mathematical Modelling**, v. 37, p. 3436-3448, 2013. https://doi.org/10.1016/j.apm.2012.07.023

LIN, Y.-K.; FOWLER, J.W.; PFUND, M.E. Multiple-objective heuristics for scheduling unrelated parallel machines. **European Journal of Operational Research**, v. 227, n. 2, p. 239-253, 2013. https://doi.org/10.1016/j.ejor.2012.10.008

McNAUGHTON, R. Scheduling with deadlines and loss functions. **Management Science**, v. 6, n. 1, p. 1-12, 1959. https://doi.org/10.1287/mnsc.6.1.1

MENDES, A.S.; MULLER, F.M.; FRANÇA, P.M.; MOSCATO, P. Comparing meta-heuristic approaches for parallel machine scheduling problems. **Production Planning and Control**, v. 13, n. 2, p. 143-154, 2002. https://doi.org/10.1080/09537280110069649

MIN, L.; CHENG, W. Identical parallel machine scheduling problem for minimizing the makespan using genetic algorithm combined by simulated annealing. **Chinese Journal of Electronics**, v. 7, n. 4, p. 317-321, 1998.

MOKOTOFF, E. Parallel machine scheduling problems: a survey. **Asia-Pacific Journal of Operational Research**, v. 18, n. 1, p. 193-242, 2001.

MONMA, C.L.; POTTS, C.N. On the complexity of scheduling with batch setup times. **Operations Research**, v. 37, n. 5, p. 798-804, 1989. https://doi.org/10.1287/opre.37.5.798

MONTOYA-TORRES, J. R.; SOTO-FERRARI, M.; GONZÁLEZ-SOLANO, F. Production scheduling with sequence-dependent setups and job release times. **Dyna**, v. 77, n. 163, p. 260-269, 2010.

MONTOYA-TORRES, J. R.; SOTO-FERRARI, M.; GONZÁLEZ-SOLANO, F.; ALFONSO-LIZARAZO, E.H. Machine scheduling with sequence-dependent setup times using a randomized search heuristic. **IEEE – International Conference on Computers & Industrial Engineering**, p. 28-33, 2009. https://doi.org/10.1109/iccie.2009.5223742

MORA, B.; MOSHEIOV, G. Batch scheduling on uniform machines to minimize total flow-time. **Computers & Operations Research**, v. 39, n. 3, p. 571-575, 2012. https://doi.org/10.1016/j.cor.2011.05.014

MUNTZ, R.R.; COFFMAN, E.G. Optimal preemptive scheduling on two-processor systems. **IEEE Transactions on Computers**, v. 18, n. 11, p. 1014-1020, 1969. https://doi.org/10.1109/T-C.1969.222573

NAIT, T. D.; CHU, C.; YALAOUI, F.; AMODEO, L. A new approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times based on linear programming. **International Conference on Industrial Engineering and Production Management** (IEPM-03), v. 3, p. 266-274, 2003.

OVACIK, I.M.; UZHOY, R. Worst-case error bounds for parallel machine scheduling problems with bounded sequence-dependent setup times. **Operations Research Letters**, v. 14, n. 5, p. 251-256, 1993. https://doi.org/10.1016/0167-6377(93)90089-Y

PINEDO, M. **Scheduling:** Theory, Algorithms, and Systems. New Jersey: Prentice-Hall, 2016.

RADHAKRISHNAN, S.; VENTURA, J.A. Simulated annealing for parallel machine scheduling with earliness-tardiness penalties and sequence-dependent set-up times. **International Journal of Production Research**, v. 38, n. 10, p. 2233-2252, 2000. https://doi.org/10.1080/00207540050028070

RAJGOPAL, J.; BIDANDA, B. On scheduling parallel machines with two setup classes. **International Journal of Production Research**, v. 29, n. 12, p. 2443-2458, 1991. https://doi.org/10.1080/00207549108948095

RUIZ, R.; ANDRÉS-ROMANO, C. Scheduling unrelated parallel machines with resource-assignable sequence-dependent setup times. **The International Journal of Advanced Manufacturing Technology**, v. 57, p. 777-794, 2011. https://doi.org/10.1007/s00170-011-3318-2

SANTOS, F.C.; VILARINHO, P.M. The problem of scheduling in parallel machines: a case study. **Proceedings of the World Congress on Engineering**, v. 3, 2010.

SVELTANA, A.; KRAVCHENKO, S.; WERNER, F. A heuristic algorithm for minimizing mean flow time with unit setups. **Information Processing Letters**, v. 79, p. 291-296, 2001. https://doi.org/10.1016/S0020-0190(01)00136-3

TURKER, A. K.; SEL, C. Scheduling two parallel machines with sequence dependent setups and a single server. **Gazy University Journal of Science**, v. 24, n. 1, p. 113-123, 2011.

VALLADA, E.; RUIZ, R. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. **European Journal of Operational Research**, v. 211, n. 3, p. 612-622, 2011. https://doi.org/10.1016/j.ejor.2011.01.011

VAN HOP, N.; NAGARUR, N.N. The scheduling problem of PCBs for multiple non-identical parallel machines. **European Journal of Operational Research**, v. 158, p. 577-594, 2004. https://doi.org/10.1016/S0377-2217(03)00376-X

WENG, M.X.; LU, J.; REN, H. Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. **International Journal of Production Economics**, v. 70, n. 3, p. 215-226, 2001. https://doi.org/10.1016/S0925-5273(00)00066-9

YANG, D.-L.; YANG, S.-J. Unrelated parallel-machine scheduling problems with multiple rate-modifying activities. **Information Sciences**, v. 235, p. 280-286, 2013. https://doi.org/10.1016/j.ins.2013.02.013

ZHU, X.; WILHELM, W.E. Scheduling and lot sizing with sequence-dependent setup: a literature review. **IIE Transactions**, v. 38, p. 987-1007, 2006. https://doi.org/10.1080/07408170600559706

Artigo recebido em 03/03/2016 e aceito para publicação em 06/04/2017
DOI: http://dx.doi.org/10.14488/1676-1901.v17i2.2375