UNIVERSIDADE DE SÃO PAULO

ALBERTO KOOPMAN OVANDO

Avaliação de métodos de ordenação inicial para o problema de programação da produção em ambiente *flowshop* não permutacional com critério de minimização do *makespan*

São Carlos
2015

ALBERTO KOOPMAN OVANDO

Avaliação de métodos de ordenação inicial para o problema de programação da produção em ambiente *flowshop* não permutacional com critério de minimização do *makespan*

Trabalho de conclusão de curso apresentado à Escola de Engenharia de São Carlos, Universidade de São Paulo

Curso de Engenharia de Produção Mecânica

Orientador: Dr. Marcelo Seido Nagano

São Carlos
2015

Resumo

Este trabalho aborda o problema de programação da produção de minimização do *Makespan* em ambiente não permutacional. Tem como objetivo a melhora das soluções do método heurístico HFC proposto por Koulamas(1998) a partir da alteração do método de ordenação inicial das tarefas utilizado pelo autor. Foram propostos 24 métodos alternativos ao original baseados em diferentes lógicas de ordenação: ordenação baseada nas médias e desvio padrão do tempo das tarefas; e uma ordenação baseada na alteração do índice proposto por Koulamas(1998). Foi realizada experimentação computacional de forma a se comparar o desempenho de cada um dos métodos.

Palavras-chave: programação da produção, *flowshop*, métodos heurísticos, *makespan*, ambiente não permutacional.

Abstract

This project addresses the scheduling problem of minimizing the maximum completion time when non-permutation schedules may be optimal. The objective is to improve the solutions of the HFC method suggested by Koulamas(1998) by changing how the first sequence is constructed. This work suggests 24 methods that use several rationale for scheduling: schedules based on the arithmetic means and standard deviations of the duration of the jobs; schedules based on the arithmetic means of the duration of the jobs prioritizing machines with larger processing times, and a schedule based on the improvement of the index suggested by Koulamas(1998). A computational experiment was performed in order to compare and evaluate each of the methods.

Keywords: scheduling, flowshop, heuristic methods, makespan, non-permutation schedules.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Este manuscrito consiste no trabalho de conclusão do curso em Engenharia de Produção mecânica pela Escola de Engenharia de São Carlos – Universidade de São Paulo, sendo um requisito obrigatório e parcial para obtenção de grau. Este trabalho foi realizado sob orientação do professor Marcelo Seido Nagano, com auxílio do doutorando Fernando Rossi.

O presente trabalho trata de um experimento computacional dentro da grande área de planejamento e controle da produção, mais especificamente sobre métodos de programação de operações (chamado de *Scheduling*). Busca-se testar alterações em parte do algoritmo do método heurístico HFC, apresentado por Koulamas (1998), com o intuito de melhorar seus resultados quanto à redução do *makespan* (C_{max}).

Este trabalho inicia-se com a com uma breve apresentação sobre planejamento e controle da produção e sobre como a programação da produção se situa dentro da Engenharia de Produção. Em seguida, apresentam-se mais informações sobre o conceito de programação da produção e a relevância do tema.

Nos capítulos seguintes, são apresentados os objetivos do trabalho e o método utilizado para se alcançar estes objetivos, incluindo a descrição dos algoritmos utilizados e os resultados do pseudocódigo de programação desenvolvido. Por fim, os resultados são analisados, e são discutidas as potenciais implicações deste projeto para a pesquisa em programação de operações.

1.1 CONTEXTUALIZAÇÃO DO TRABALHO

O conceito de PCP pode ser entendido como "[...] série de decisões com o objetivo de definir o que, quanto e quando produzir, comprar e entregar, além de quem e/ou onde e/ou como produzir." (SLACK, 2009). Nota-se que se trata de um conceito bastante amplo, que abrange de forma direta todo o funcionamento de um sistema produtivo, sendo o objetivo deste sistema um produto ou um serviço.

O PCP, por se tratar de um conceito amplo, abrange diversos temas que impactam a tomada de decisão. Parte das atividades do PCP é definida como pertencentes ao Planejamento da Produção (PP) e parte deles ao Controle da Produção (CP).

Fernandes (1991) e Burbridge (1990) diferenciam PP de CP em relação aos seus objetivos e abordagens. Enquanto o PP se refere a atividades de decisões de médio e longo prazo que possibilitem tornar eficiente de forma macro o sistema produtivo, o CP está mais ligado à regulação do sistema produtivo. Desta forma, as decisões detalhadas de curto prazo ficam a cargo do CP.

As atividades do PCP, como apresentado por Fernandes e Godinho (2010), são resumidos em nove itens principais. Inicialmente, as quatro atividades ligadas ao PP:

i) "Prever a demanda (Previsão);
ii) Desenvolver um plano de produção agregado (Planejamento Agregado da Produção);
iii) Realizar um planejamento da capacidade que suporte o planejamento agregado (Planejamento de Capacidade de médio prazo, também chamado RRP = *Resource Requirements Planning*);
iv) Desagregar o plano de produção (Desagregação)".

Já as atividades classificadas por Fernandes (2010) como CP são:

v) "Programar a produção no curto prazo em termos de itens finais (Programa Mestre de Produção – MPS) e analisar a capacidade no nível MPS;
vi) Controlar por meio de regras de controle (por exemplo, regras de controle de estoques) ou programar as necessidades em termos de componentes e materiais e avaliar/analisar a capacidade no nível SCO;
vii) Controlar a emissão/liberação das ordens de produção e compra, determinando se e quando liberar as ordens (atividade chamada na literatura de revisão e liberação de ordens – *Order Review and Release* – ORR);
viii) Controlar estoques;
ix) Programar/sequenciar as tarefas nas máquinas (*dispatching* ou *scheduling*)".

Neste trabalho o foco principal é a atividade de programação/sequênciamento das tarefas nas máquinas.

1.2 PROGRAMAÇÃO DA PRODUÇÃO

As definições de Morton e Pentico (1993) e Sipper e Bulfin (1997) nos permitem descrever a programação da produção, de forma geral, como o processo dinâmico de tomada de decisão relacionado à ordenação e alocação de tarefas nos recursos disponíveis, levando-se

em conta todas as restrições que possam existir (como tempo, relações entre atividades e recursos).

A importância da programação da produção deve-se a dois motivos:

i) As operações de *scheduling* têm forte impacto no desempenho dos sistemas produtivos. Conforme apresentado por MacCarthy e Liu (1993), a necessidade de responder às rápidas demandas do mercado e operar de maneira eficiente sistemas produtivos torna muito importante o estudo;

ii) A complexidade do problema. O avanço tecnológico não é o suficiente para tornar obsoleta a necessidade por bons algoritmos que determinem as melhores sequências. Sipper, e Bulfin (1997) dão como exemplo um problema de uma única máquina que precisa realizar 32 *jobs* (tarefas). O número de sequências possíveis para este problema aparentemente simples é $32! = 2{,}5 * 10^{35}$. Para testar todas as opções de sequências seria necessário que um computador rápido processasse o problema por centenas de anos.

Devido à complexidade dos problemas de programação da produção e às limitações computacionais para se realizar todas as operações necessárias para se obter uma solução ótima para este tipo de problema, foram desenvolvidos algoritmos heurísticos para resolvê-los. Estas soluções heurísticas realizam um número de operações menor do que a necessária para se obter uma solução ótima, de forma a ser uma operação viável com os recursos computacionais existentes e obtém uma solução aproximada. O objetivo de se desenvolver algoritmos eficientes é a busca por soluções mais próximas do ótimo, com um menor uso dos recursos computacionais (PINEDO, 2008).

Framinan, Gupta e Leisten (2004) classificam as heurísticas como a composição de duas heurísticas, construtivas e de melhoramento, com a criação de um índice:

i) Heurística construtiva - a solução constrói uma sequência de tarefas baseadas em determinados critérios.

ii) Heurística de melhoramento - parte de uma sequência já existente e tenta gerar resultados mais próximos aos objetivos.

De acordo com Graham et al.(1979) um problema de programação da produção pode ser representado por $\alpha \mid \beta \mid \gamma$. O primeiro campo refere-se ao ambiente da produção, o segundo refere-se às características de processamento e restrições, e o terceiro o objetivo a ser minimizado.

MacCarthy e Liu (1993) e Rossi (2014) apresentam alguns possíveis valores para cada um dos campos:

Ambiente da produção:

i) 1: Máquina única – Todas as tarefas são realizadas em uma única máquina.

ii) J: *Job shop* – As tarefas devem seguir regras de sequenciamento.

iii) F: *Flow shop* – Além de seguir regras de sequenciamento, toda tarefa é realizada uma única vez em cada uma das máquinas.

iv) O: *Open shop* - Não há regras de sequênciamento nas operações.

Características de processamento e restrições:

i) *Setup:* Sequência dependente dos tempos de *Setup*

ii) Prmu (permutacional): Requer que a ordem de processamento das tarefas em todas as máquinas seja a mesma.

Objetivo:

i) *Makespan* (C_{max}): objetivo é minimizar a utilização dos recursos. É o tempo de conclusão da última tarefa.

ii) *Flowtime* ($\sum C_j$): resposta rápida à demanda. É a soma dos tempos de conclusão de todas as tarefas.

2 OBJETIVOS

i) Estudar os conceitos teóricos de programação da produção (com foco para o método não permutacional HFC), além de alguns conceitos teóricos ligados a PCP, por ser o tema ao qual programação da produção pertence.

ii) Desenvolver métodos alternativos à primeira fase do HFC (Koulamas, 1998) de forma a se obter melhores soluções.

Portanto o presente trabalho foca em desenvolver métodos heurísticos para o problema de programação da produção em ambiente *flowshop* não permutacional com critério de minimização do *makespan*: F_m |no-prmu|C_{max}. No próximo capítulo serão apresentadas as principais heurísticas presentes na literatura.

3 REVISÃO DE LITEATURA

3.1 PROGRAMAÇÃO *FLOWSHOP*

Os problemas de programação *flowshop* são caracterizados em pesquisa operacional como problemas em que todas as tarefas são realizadas em todas as máquinas. Os tempos de processamento de cada uma destas tarefas em cada uma das máquinas são determinados e fixos. Os tempos de *setup* são desconsiderados ou incluídos nos tempos de processamento de cada tarefa. As tarefas devem ser concluídas após iniciadas, ou seja, não pode haver interrupção para posterior término da tarefa. Não há restrição quanto a estoques intermediários.

Estas características são consideradas nos métodos teóricos de pesquisa operacional. No entanto, não representam de maneira precisa os problemas reais em fábrica, uma vez que estes podem ser bastante complexos. Por isso, simplifica-se o problema real para estudá-lo.

3.2 DEFINIÇÃO DO PROBLEMA

Segundo Maccarthy e Liu (1993) pode-se definir o problema $F_m \mid \text{prmu} \mid C_{max}$ como uma sequência π de n tarefas que deve ser realizadas por m máquinas mantendo a sequência com que as tarefas são realizadas. As n tarefas são representadas pela variável j enquanto i representa o número de cada uma das m máquinas. Assim, os tempos de processamentos são representados por p_{ij}. C_{ij} É o tempo em que a tarefa j é concluída na máquina i. O C_{max} (*makespan*) é determinado como o tempo de conclusão da última tarefa realizada na última máquina. C_{ij} É calculado conforme a expressão (1):

$$\begin{gathered} C_{ij} = p_{ij}\,, para\ i = j = 1 \\ C_{ij} = C_{ij-1} + p_{ij}, para\ i = 1\ e\ (j = 1, \dots, n) \\ C_{ij} = C_{i-1j} + p_{ij}, para\ j = 1\ e\ (i = 1, \dots, m) \\ C_{ij} = max(C_{ij}, C_{ij-1}) + p_{ij}\,, para(i = 1, \dots, m) e\ (j = 1, \dots, n) \end{gathered} \tag{1}$$

Além do ambiente permutacional apresentado acima, Koulamas (1998) utiliza um método não permutacional. A diferença básica entre os dois ambientes é que o não permutacional fornece soluções em que as sequências de tarefas não são necessariamente as mesmas para todas as máquinas. As Figuras 1 e 2 ilustram a diferença entre soluções permutacionais e não permutacionais.

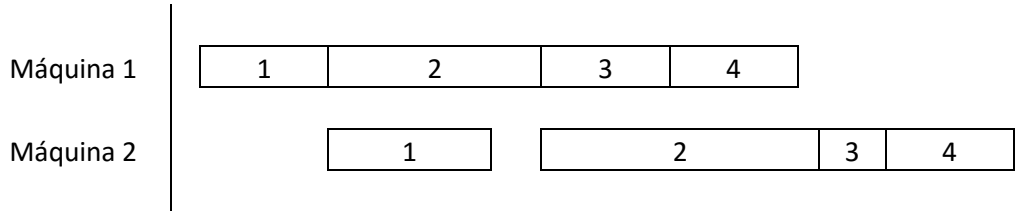

Figura 1 - Gráfico de Gantt de uma solução permutacional.

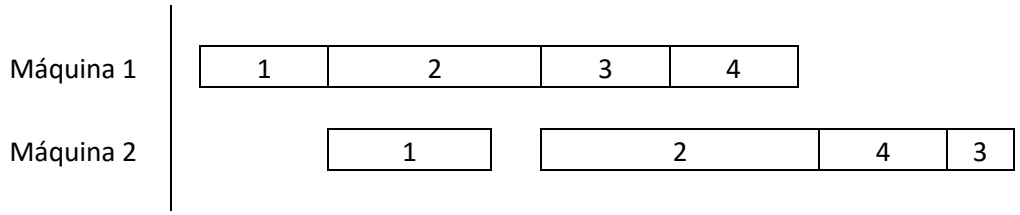

Figura 2 - Gráfico de Gantt de uma solução não permutacional.

3.3 MÉTODOS PARA A RESOLUÇÃO DE PROBLEMAS *FLOWSHOP* PERMUTACIONAL COM CRITÉRIO DE MINIMIZAÇÃO DO *MAKESPAN*

Vários métodos heurísticos foram criados para a solução de problemas de *flowshop* para a minimização do *makespan.* O precursor deles foi Johnson (1954) com seu método com solução ótima para problemas com n tarefas em duas máquinas. Parte dos métodos heurísticos que surgiram após o método de Johnson (1954) aplicam a lógica utilizada por ele em problemas de duas máquinas para problemas com m máquinas. Para isto, realizam o método para todos, ou alguns, pares de máquinas do problema e, a partir disso, chega-se a uma solução.

Palmer (1965) utiliza em sua heurística a criação de um índice chamado de *slope index*, e a partir deste índice, cria-se uma solução. Este índice baseia-se de forma a priorizar que tarefas que tendem a ter tempos de processamento cada vez maiores nas máquinas subsequentes sejam executadas primeiramente. O *slope index* é calculado conforme apresentado na expressão (2):

$$H_j = \sum_{i=1}^{m}(2i - m - 1)p_{ij}\ ,\ para\ j = 1,2, \dots n \quad (2)$$

Depois de calculado o índice, as tarefas são ordenadas de forma crescente de acordo com seus valores H_j.

Campbell, Dudek & Smith (1970) propuseram um método baseado no método de Johnson (1954). Nele são criados $(m - 1)$ problemas com 2 máquinas, e em cada um deles é aplicado o método de Johnson (1954). Os resultados obtidos são utilizados para calcular o *makespan* de cada um dos $(m - 1)$ problemas, e a sequência com menor *makespan* é escolhida como solução do problema.

Outro método heurístico foi o proposto por Nawaz, Enscore e Ham (1983). Este método é conhecido como NEH. Este método baseia-se na hipótese de que as maiores prioridades devem ser dadas às tarefas com maior tempo de processamento total nas m máquinas. Após esta ordenação de tarefas, cada umas das tarefas é testada em todas as posições possíveis. A sequência que produzir o menor *makespan* é escolhida como solução do problema. O método NEH é utilizado como referência por Koulamas (1998) para medir o desempenho de seu método: o HFC.

No tópico 3.4 é apresentado de forma mais detalhada o método HFC de Koulamas (1998), que é o método utilizado como base deste trabalho.

3.4 MÉTODO HFC

Diferentemente dos métodos anteriores, o HFC fornece solução não permutacional quando aplicável. Koulamas (1998) destaca a eficiência de seu método por se tratar, até o momento de sua publicação, do melhor método não permutacional existente.

O método HFC é uma heurística desenvolvida para a solução de problemas em ambiente *flowshop* com solução permutacional ou não permutacional, quando aplicável. Ela tem o objetivo de minimizar o *Makespan* (C_{max}). De acordo Potts et al. (1991), apesar de a maioria das heurísticas buscarem solução apenas permutacional, isso resulta numa perda da qualidade das soluções. O método HFC foi testado contra o método heurístico construtivo mais eficiente na data de sua publicação: o método NEH (NAWAZ et al., 1983). O método

HFC apresentou um desempenho próximo ao do método NEH e, na data de sua publicação, era o método heurístico não permutacional com melhor desempenho para o tipo de problema testado.

De acordo com a nomenclatura mostrada por Graham et al. (1979), este ambiente pode ser definido como F//C_{max}.

Seguindo a classificação de Framinan, Gupta e Leisten (2004), o método HFC apresenta as três fases:

i) Criação de índice – Baseado nas características das tarefas e das máquinas, é criado um índice com um valor para cada uma das tarefas.

ii) construção da solução – O índice criado em i) é utilizado para criar uma solução inicial para o problema.

iii) melhora da solução – São testadas soluções que alteram a solução obtida em ii), e quando obtém-se uma melhor solução esta substitui a anterior.

As três fases podem ser observadas no algoritmo HFC, apresentado nas Figuras 3 e 4. A nomenclatura utilizada por Koulamas (2006) em seu algoritmo, no entanto, apresenta as fases i e ii juntas na Fase I. Enquanto a fase iii de Framinan, Gupta e Leisten (2004) está na Fase II do algoritmo :

```
Algoritmo HFC (Fase I)
Passo 0.        I_i := 0; i = 1,...,n
Passo 1.        Para i = 1,...,n
                    Para j = j + 1,...,n
                        Para k = 1,...m
                            Para l = k + 1,...,m
                                Se mínimo(p_ik, p_jl)<mínimo(p_il, p_jk)
                                Então
                                I_i = I_i − 1
                                I_j = I_j + 1
                                H((i,j),(k,l)) = −1
                                Senão Se mínimo(p_ik, p_jl)>mínimo(p_il, p_jk)
                                Então
                                I_i = I_i + 1
                                I_j = I_j − 1
                                H((i,j),(k,l)) = 1
                                Fimse
                            Próximo l
                        Próximo k
                    Próximo j
                Próximo i
```

Figura 3 - Algoritmo HFC original – Primeira parte.
Fonte: KOULAMAS, 1998.

Passo 2. Ordenar I_i em ordem crescente e sequenciar as tarefas j na mesma ordem
Passo 3. Seja $\sigma(1), \ldots, \sigma(n)$ a sequência obtida, calcule o C_{max} e vá para fase II.

Algoritmo HFC (Fase II)
Para $i = 1, \ldots, n-1$
Seja k a linha em H correspondente ao par de tarefas $\sigma(i), \sigma(i+1)$.
Para $j = 2, \ldots, m-1$
$T_1 = 0$
Para $q = 1, \ldots, j-1$
Para $r = q+1, \ldots, j$
Seja l a coluna em H que corresponte ao par de máquinas q, r.
$T_1 = T_1 + H(k, l)$
Próximo r
Próximo q
Se $|T_1| < \big(j(j-1)\big)/2$ próximo j
$T_2 = 0$
Para $q = j+1, \ldots, m$
Para $r = q+1, \ldots, m$
Seja l a coluna em H que corresponde ao par de máquinas q, r.
$T_2 = T_2 + H(k, l)$
Próximo r.
Próximo q
Se $T_2 \neq T_1$
próximo j
Calcule C_{max} para a sequência $\sigma *$ onde $\sigma^*(k) = \sigma(k)$ para todo $k = 1, \ldots, i-1, i+2, \ldots n. \sigma^*(i) = \sigma(i), \sigma^*(i+1) = \sigma(i+1)$ para máquinas 1,...,j e $\sigma^*(i) = \sigma(i+1), \sigma^*(i+1) = \sigma(i)$ para máquinas $j+1, \ldots, m$.
Se $C_{max}(\sigma^*) < C_{max}(\sigma)$ então σ^* se torna a solução atual.
Próximo j
Próximo i

Figura 4 - Algoritmo HFC original - Segunda Parte.
Fonte: KOULAMAS, 1998, tradução do autor.

A partir do algoritmo HFC apresentado nas Figuras 3 e 4, foi programado o método HFC e métodos alternativos que utilizavam métodos distintos para as fases i e ii. Mais detalhes sobre os experimentos são apresentados nas seções seguintes.

4 MÉTODOS

4.1 EXPERIMENTO

Realizou-se um experimento computacional comparando os métodos propostos com o método HFC original para problemas de *flowshop* com minimização do *makespan*. Analisaram-se 24 alternativas à ordenação inicial realizada pelo HFC original. Utilizaram-se os problemas testes de Taillard (1993). Os métodos foram programados na linguagem C++, num compilador Dev-C++ versão 4.9.9.2 da Bloodshed. Foi executado em um processador Intel(R) Core(TM) i7-3600QM 2.30 GHz com 8 GB de memória RAM e HD de 1TB. Os problemas de Taillard são problemas variados com as diferentes complexidades n x m (número de tarefas x número de máquinas): 20x5, 20x10, 20x20, 50x5, 50x10, 50x20, 100x5, 100x10, 100x20, 200x10, 200x20, 500x20.

4.2 ALGORITMOS

Inicialmente, programou-se a versão original de HFC baseado no algoritmo apresentado por Koulamas (1998). A fase I foi modificada de forma a termos diferentes sequências iniciais para a fase II, de acordo com cada um dos 24 métodos propostos. Estes 24 métodos são classificados em quatro grupos de algoritmos, como descritos a seguir.

4.2.1 Métodos baseados nas médias e desvios padrão

Compreende os 11 primeiros métodos (métodos 1 a 11). Nestes algoritmos, a sequência inicial fornecida como entrada à fase II é baseada na ordenação de x_J de forma crescente. O cálculo de x_J é apresentado na expressão 3.

$$x_J = a * \sum_{i=1}^{m} p_{ij} + b * \sqrt{\frac{1}{m-1} + \sum_{i=1}^{m} (p_{ij} - \overline{p_j})^2} \qquad (3)$$

Sendo a e b a importância dado à média e ao desvio padrão na composição de x. Os valores de$a$ e $b$são apresentados no Quadro 1 a seguir:

Quadro 1 - Relação entre métodos e pesos a e b

Métodos	a	b
H1	1	0
H2	0,9	0,1
H3	0,8	0,2
H4	0,7	0,3
H5	0,6	0,4
H6	0,5	0,5
H7	0,4	0,6
H8	0,3	0,7
H9	0,2	0,8
H10	0,1	0,9
H11	0	1

Após o cálculo de x_J para cada uma das n tarefas, as tarefas são ordenadas em ordem crescente e decrescente de acordo com seus valores de x_J. Esta sequência é a entrada da fase II do algoritmo.

4.2.2 Ordenação baseada nas médias dos tempos das tarefas com priorização das máquinas com maior tempo de processamento

Neste algoritmo a ordenação inicial é baseada nas médias dos tempos de cada tarefa ponderadas pelo fator y_i. Esta ponderação nos fornece os valores x_J, e as tarefas são ordenadas de acordo com estes valores. Os cálculos de y_i e x_J são apresentados nas expressões 4 e 5.

$$y_i = \frac{1}{n} \times \sum_{j=1}^{n} p_{ij} \quad (4)$$

$$x_j = \sum_{i=1}^{m} y_i \times p_{ij} \quad (5)$$

As tarefas são colocadas em ordem crescente baseadas em seus valores de x_j. Esta sequência é a entrada da segunda fase do algoritmo.

4.2.3 Método HFC com I_j modificado

Ao estudar o funcionamento do algoritmo HFC, nota-se que sua ordenação inicial é baseada no índice I_j calculado para cada tarefa j. O método H24, apresentado neste tópico, se baseou na hipótese de que este índice pode ser calculado de uma forma mais precisa a partir de algumas mudanças simples. O índice I_j é calculado considerando-se as diferenças entre os tempos de uma grande quantidade de pares de tarefas. O cálculo deste índice não leva em conta a magnitude das diferenças. Por isso este método propõe que as diferenças dos tempos entre esses pares de tarefas sejam levados em conta.

Considerando-se p_{ij} o tempo de duração da tarefa j na máquina i, temos a matriz P exemplo abaixo:

$$P = \begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{pmatrix}$$

O método original HFC faz a ordenação inicial baseada no índice I_j. O algoritmo executa repetidas vezes a operação descrita na Figura 5, de forma que todos os subproblemas de p_{ij} sejam testados.

```
Se      (Mínimo (p11, p22)< Mínimo (p11, p22)) então
        Ii=Ii-1;
Ij=Ij+1;
Se      (Mínimo (p11, p22)> Mínimo (p11, p22)) então
        Ii=Ii+1;
        Ij=Ij-1;
```

Figura 5 - Cálculo do índice I - Trecho do HFC

Fonte: KOULAMAS, 1998, tradução do autor.

O algoritmo da Figura 5 calcula um valor de I para cada tarefa, aqui chamada de j. Então temos I_j. Vale ressaltar que os valores de i e $j$representados no algoritmo em "I_i" e "I_j" não seguem a mesma nomenclatura de i e j usado neste trabalho, mas sim uma variável interna do algoritmo.

Baseado na sequência crescente de I_j, são sequenciadas as tarefas de forma crescente.

Diferentemente do método original apresentado acima, o método H24 testou uma abordagem distinta quanto aos incrementos de I_j. Enquanto no método original os incrementos são discretos (1,-1) independentemente da diferença entre Mínimo (p_{11}, p_{22}) e Mínimo (p_{11}, p_{22}), aqui se propõe que esta magnitude seja levada em conta. Assim, ao invés de se incrementar 1 e -1, incrementa-se |Mínimo (p_{11}, p_{22}) - Mínimo (p_{11}, p_{22})| e -|Mínimo (p_{11}, p_{22}) - Mínimo (p_{11}, p_{22})|.

4.3 MÉTODO DE AVALIAÇÃO

Para avaliar os desempenhos dos métodos foi utilizado o desvio relativo (DR) conforme apresentado na expressão 6:

$$\mathrm{DR}\left(C_{max}(\pi^h)\right) = \frac{(C_{max}(\pi^h) - C_{max}{}^*)}{C_{max}{}^*} \times 100 \qquad (6)$$

Onde $C_{max}(\pi^h)$ é o makespan fornecido pela sequência π^h. Utilizaram-se como referência as melhores soluções de $C_{max}{}^*$ conhecidos para os problemas testes de Taillard.

5 RESULTADOS

A partir dos dados obtidos do programa, foram comparados os desempenhos entre cada um dos métodos propostos (H1 a H24) com o método HFC original. Os resultados são apresentados a seguir, divididos em quatro partes: métodos baseados nas médias e desvios padrão das tarefas com ordenação crescente; métodos baseados nas médias e desvios padrão das tarefas com ordenação decrescente; ordenação baseada nas médias dos tempos das tarefas com priorização das máquinas com maior tempo de processamento; e método HFC com índice I_j modificado.

5.1 MÉTODOS BASEADOS NAS MÉDIAS E DESVIOS PADRÃO COM ORDENAÇÃO CRESCENTE

As tabelas 1 a 9 a seguir ilustram a comparação entre o método HFC original e cada um dos métodos baseados em médias e desvios padrão. Nota-se que, de forma geral, os métodos propostos apresentam desempenho inferior ao HFC original, com exceção do método H24.

5.1.1 Método HFC

A tabela 1 apresenta os resultados do método HFC original, utilizado como referência neste projeto.

Tabela 1 – DRM em relação à melhor solução existente ao problema, e tempo de processamento do HFC original.

Problema	HFC	
	DRM	Tempo
20 x 5	8,37	0,00
20 x 10	14,98	0,00
20 x 20	15,09	0,00
50 x 5	4,54	0,00
50 x 10	14,49	0,00
50 x 20	20,57	0,01
100 x 5	3,18	0,00
100 x 10	8,80	0,01
100 x 20	17,48	0,03
200 x 10	5,82	0,04
200 x 20	13,51	0,12
500 x 20	7,49	0,73
Média	11,19	0,08

5.1.2 Métodos baseados em medias e desvios com ordenação crescente

Observa-se nas tabelas 2, 3 e 4 que o método HFC apresenta desempenho superior a todos os métodos H1 a H11 no critério DRM, com nenhuma diferença significativa no tempo de execução.

Tabela 2- DRM em relação à melhor solução existente ao problema, e tempo de processamento do HFC, H1, H2 e H3.

Problema	HFC		H1		H2		H3	
	DRM	Tempo	DRM	Tempo (s)	DRM	Tempo (s)	DRM	Tempo (s)
20 x 5	8,37	0,00	23,08	0,00	22,95	0,00	21,37	0,00
20 x 10	14,98	0,00	29,56	0,00	29,61	0,00	28,88	0,00
20 x 20	15,09	0,00	23,77	0,00	22,90	0,00	24,13	0,00
50 x 5	4,54	0,00	17,74	0,00	17,07	0,00	17,02	0,00
50 x 10	14,49	0,00	27,49	0,00	25,61	0,00	26,02	0,00
50 x 20	20,57	0,01	30,08	0,01	31,26	0,01	29,31	0,01
100 x 5	3,18	0,00	12,76	0,00	14,30	0,00	13,67	0,00
100 x 10	8,80	0,01	17,96	0,01	20,63	0,01	19,46	0,01
100 x 20	17,48	0,03	25,68	0,03	25,99	0,03	23,94	0,03
200 x 10	5,82	0,04	14,95	0,03	15,97	0,03	15,96	0,03
200 x 20	13,51	0,12	22,40	0,11	21,77	0,11	21,73	0,11
500 x 20	7,49	0,73	15,17	0,65	15,63	0,65	16,08	0,65
Média	11,19	0,08	21,72	0,07	21,98	0,07	21,46	0,07

Tabela 3 - DRM em relação à melhor solução existente ao problema, e tempo de processamento do H4, H5, H6 e H7.

Problema	H4		H5		H6		H7	
	DRM	Tempo	DRM	Tempo (s)	DRM	Tempo (s)	DRM	Tempo (s)
20 x 5	24,06	0,00	25,96	0,00	22,29	0,00	25,17	0,00
20 x 10	27,74	0,00	32,34	0,00	30,20	0,00	28,58	0,00
20 x 20	23,11	0,00	20,84	0,00	19,58	0,00	22,82	0,00
50 x 5	17,86	0,00	17,39	0,00	19,44	0,00	16,60	0,00
50 x 10	27,80	0,00	23,92	0,00	27,21	0,00	26,32	0,00
50 x 20	29,01	0,01	30,71	0,01	30,81	0,01	30,48	0,01
100 x 5	13,42	0,00	13,10	0,00	15,14	0,00	16,01	0,00
100 x 10	20,07	0,01	21,07	0,01	19,35	0,01	19,36	0,01
100 x 20	25,63	0,03	26,05	0,03	26,17	0,03	24,92	0,03
200 x 10	15,30	0,03	16,03	0,03	15,48	0,03	16,43	0,03
200 x 20	20,93	0,11	22,25	0,11	21,97	0,11	22,63	0,11
500 x 20	15,90	0,65	15,62	0,65	15,87	0,65	15,57	0,65
Média	21,73	0,07	22,11	0,07	21,96	0,07	22,07	0,07

Tabela 4 - DRM em relação à melhor solução existente ao problema, e tempo de processamento do H8, H9, H10 e H11.

Problema	H8		H9		H10		H11	
	DRM	Tempo	DRM	Tempo (s)	DRM	Tempo (s)	DRM	Tempo (s)
20 x 5	24,47	0,00	23,28	0,00	24,27	0,00	24,76	0,00
20 x 10	27,88	0,00	30,03	0,00	27,96	0,00	28,67	0,00
20 x 20	22,58	0,00	23,49	0,00	23,34	0,00	22,60	0,00
50 x 5	16,43	0,00	16,23	0,00	16,02	0,00	16,83	0,00
50 x 10	27,51	0,00	27,12	0,00	27,96	0,00	27,88	0,00
50 x 20	30,57	0,01	30,35	0,01	31,92	0,01	30,33	0,01
100 x 5	15,97	0,00	14,56	0,00	13,02	0,00	13,69	0,00
100 x 10	19,84	0,01	19,52	0,01	20,17	0,01	18,12	0,01
100 x 20	26,76	0,03	26,32	0,03	26,16	0,03	25,74	0,03
200 x 10	15,96	0,03	15,68	0,03	16,51	0,03	14,22	0,03
200 x 20	22,05	0,11	23,15	0,11	21,63	0,11	22,74	0,11
500 x 20	15,32	0,65	15,51	0,65	15,42	0,65	15,96	0,65
Média	22,11	0,07	22,10	0,07	22,03	0,07	21,80	0,07

A Figura 6 mostra o desempenho DRM dos métodos baseados em médias e desvios em função do número do método (que apresenta ligação direta às variáveis a e b).

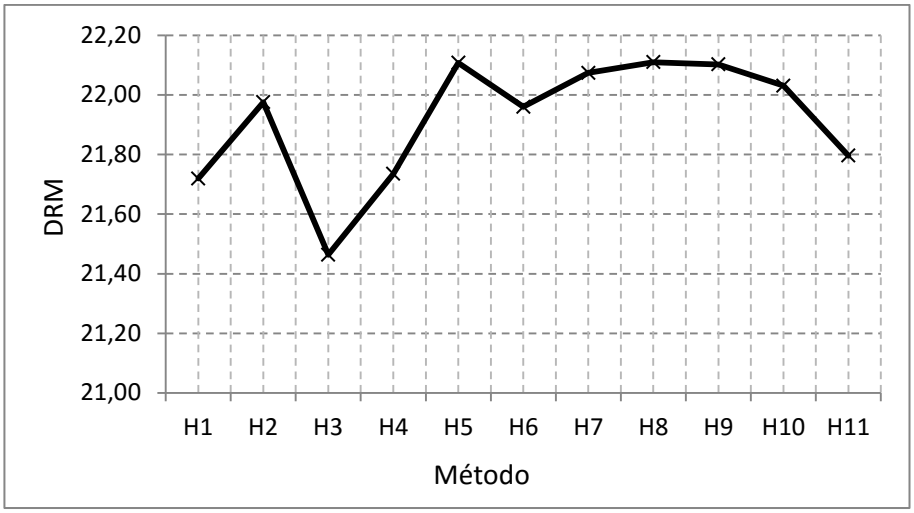

Figura 6 - Gráfico do DRM em relação às melhores soluções existentes para os métodos baseados em médias e desvios padrão com ordenação crescente.

5.1.3 Métodos baseados em médias e desvios com ordenação decrescente

Para os métodos H12 a H22 observa-se valores de DRM piores que os valores de referência para o HFC, com tempos de execução similares.

Tabela 5 - DRM em relação à melhor solução existente ao problema, e tempo de processamento do HFC, H12, H13 e H14.

Problema	HFC		H12		H13		H14	
	DRM	Tempo	DRM	Tempo (s)	DRM	Tempo (s)	DRM	Tempo (s)
20 x 5	8,37	0,00	23,10	0,00	26,18	0,00	25,40	0,00
20 x 10	14,98	0,00	27,52	0,00	26,28	0,00	28,37	0,00
20 x 20	15,09	0,00	20,49	0,00	20,97	0,00	20,09	0,00
50 x 5	4,54	0,00	15,48	0,00	16,98	0,00	17,95	0,00
50 x 10	14,49	0,00	28,44	0,00	26,39	0,00	26,99	0,00
50 x 20	20,57	0,01	30,47	0,01	30,43	0,01	31,92	0,01
100 x 5	3,18	0,00	11,83	0,00	13,58	0,00	12,10	0,00
100 x 10	8,80	0,01	19,13	0,01	20,86	0,01	21,31	0,01
100 x 20	17,48	0,03	26,48	0,03	26,78	0,03	25,59	0,03
200 x 10	5,82	0,04	14,23	0,03	16,06	0,03	15,56	0,03
200 x 20	13,51	0,12	22,16	0,11	22,61	0,11	22,30	0,11
500 x 20	7,49	0,73	15,81	0,65	15,72	0,65	15,30	0,64
Média	11,19	0,08	21,26	0,07	21,90	0,07	21,91	0,07

Tabela 6 - DRM em relação à melhor solução existente ao problema, e tempo de processamento do H15, H16, H17 e H18.

Problema	H15		H16		H17		H18	
	DRM	Tempo	DRM	Tempo (s)	DRM	Tempo (s)	DRM	Tempo (s)
20 x 5	25,99	0,00	24,81	0,00	25,43	0,00	23,88	0,00
20 x 10	27,18	0,00	28,64	0,00	27,37	0,00	28,16	0,00
20 x 20	19,32	0,00	19,03	0,00	19,92	0,00	21,91	0,00
50 x 5	17,09	0,00	17,67	0,00	16,52	0,00	17,79	0,00
50 x 10	26,54	0,00	28,27	0,00	27,97	0,00	26,22	0,00
50 x 20	30,04	0,01	28,66	0,01	31,17	0,01	29,88	0,01
100 x 5	13,17	0,00	12,78	0,00	12,09	0,00	13,75	0,00
100 x 10	20,68	0,01	19,93	0,01	19,96	0,01	22,00	0,01
100 x 20	26,79	0,03	26,43	0,03	27,62	0,03	26,48	0,03
200 x 10	15,35	0,03	15,58	0,03	15,23	0,03	14,65	0,03
200 x 20	22,22	0,10	22,11	0,10	23,17	0,11	21,61	0,11
500 x 20	15,48	0,64	15,89	0,64	15,71	0,65	15,78	0,65
Média	21,65	0,07	21,65	0,07	21,85	0,07	21,84	0,07

Tabela 7 - DRM em relação à melhor solução existente ao problema, e tempo de processamento do H19, H20, H21 e H22.

Problema	H19		H20		H21		H22	
	DRM	Tempo	DRM	Tempo (s)	DRM	Tempo (s)	DRM	Tempo (s)
20 x 5	22,22	0,00	22,94	0,00	22,21	0,00	25,36	0,00
20 x 10	29,70	0,00	28,92	0,00	29,95	0,00	29,73	0,00
20 x 20	22,35	0,00	20,68	0,00	22,29	0,00	21,14	0,00
50 x 5	17,32	0,00	17,47	0,00	19,16	0,00	18,56	0,00
50 x 10	27,65	0,00	27,85	0,00	27,65	0,00	26,50	0,00
50 x 20	28,76	0,01	32,77	0,01	30,17	0,01	29,88	0,01
100 x 5	13,09	0,00	13,17	0,00	12,50	0,00	13,36	0,00
100 x 10	20,62	0,01	20,53	0,01	19,92	0,01	19,90	0,01
100 x 20	27,16	0,03	25,83	0,03	25,08	0,03	25,73	0,03
200 x 10	15,66	0,03	15,56	0,03	15,50	0,03	15,33	0,03
200 x 20	21,51	0,10	22,19	0,11	21,67	0,11	21,96	0,11
500 x 20	16,43	0,65	16,30	0,64	16,08	0,64	16,22	0,65
Média	21,87	0,07	22,02	0,07	21,85	0,07	21,97	0,07

A Figura 7 apresenta a relação entre o desempenho e a numeração do método (relação entre a e b).

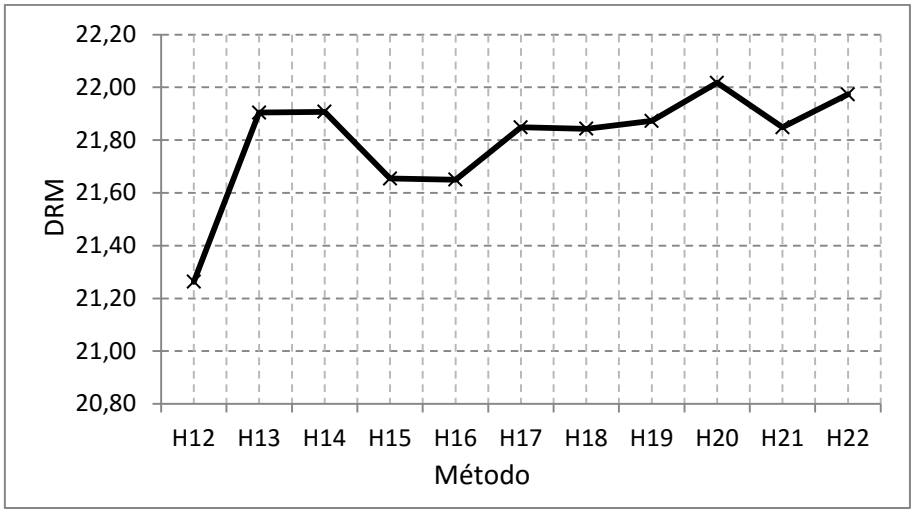

Figura 7 - Gráfico do DRM em relação às melhores soluções existentes para os métodos baseados em médias e desvios padrão com ordenação decrescente.

5.1.4 Métodos baseado em médias ponderadas

A tabela 8 mostra o desempenho da tarefa H23. Nota-se, assim como os métodos H1 a H22 uma solução inferior ao HFC original.

Tabela 8 - DRM em relação à melhor solução existente ao problema, e tempo de processamento do HFC e H23.

Problema	HFC		H23	
	DRM	Tempo	DRM	Tempo (s)
20 x 5	8,37	0,00	23,53	0,00
20 x 10	14,98	0,00	29,21	0,00
20 x 20	15,09	0,00	23,24	0,00
50 x 5	4,54	0,00	17,06	0,00
50 x 10	14,49	0,00	27,27	0,00
50 x 20	20,57	0,01	30,94	0,01
100 x 5	3,18	0,00	13,94	0,00
100 x 10	8,80	0,01	19,72	0,01
100 x 20	17,48	0,03	24,61	0,03
200 x 10	5,82	0,04	15,15	0,03
200 x 20	13,51	0,12	22,94	0,11
500 x 20	7,49	0,73	16,23	0,65
Média	11,19	0,08	21,99	0,07

5.1.5 Método baseado no índice I_j modificado

A tabela 9 mostra o desempenho de H24. Nota-se uma solução de qualidade superior à solução HFC, e com tempos de execução similares.

Tabela 9 - DRM em relação à melhor solução existente ao problema, e tempo de processamento do HFC e H24.

Problema	HFC		H24	
	DRM	Tempo	DRM	Tempo (s)
20 x 5	8,37	0,00	9,46	0,00
20 x 10	14,98	0,00	13,95	0,00
20 x 20	15,09	0,00	14,47	0,00
50 x 5	4,54	0,00	4,52	0,00
50 x 10	14,49	0,00	14,61	0,00
50 x 20	20,57	0,01	19,59	0,01
100 x 5	3,18	0,00	2,92	0,00
100 x 10	8,80	0,01	8,39	0,01
100 x 20	17,48	0,03	16,13	0,03
200 x 10	5,82	0,04	5,15	0,04
200 x 20	13,51	0,12	13,21	0,13
500 x 20	7,49	0,73	7,00	0,77
Média	11,19	0,08	10,78	0,08

Na sessão seguinte serão analisados os resultados obtidos neste capítulo.

6 DISCUSSÃO E CONCLUSÕES

Os resultados mostram uma grande diferença de desempenho quando se compara os *makespan*s obtidos para cada um dos métodos propostos em relação ao método HFC. Enquanto nenhum dos 23 primeiros métodos propostos apresentou desempenho favorável, o método H24 apresentou um desempenho superior ao método HFC original.

A comparação entre os resultados obtidos para cada um dos métodos propostos nos permite inferir algumas hipóteses a respeito do comportamento do HFC para diferentes soluções para a primeira fase. Além disso, os resultados nos permitem sugerir um caminho para as tentativas de se obter resultados cada vez mais próximos ao ótimo.

6.1 MÉTODOS HFC ORIGINAL

É importante destacar, inicialmente, o bom desempenho do HFC utilizando a primeira fase original, se comparado ao restante dos métodos em geral. O fato de este método apresentar desempenho superior aos métodos H1 a H23 nos permite observar que se trata de uma pré-ordenação com resultados sólidos e alinhados ao funcionamento do HFC como um todo. Assim sendo, nota-se a complexidade em se propor métodos com desempenho próximo ao original.

6.2 MÉTODOS BASEADOS NAS MÉDIAS E DESVIOS PADRÃO COM ORDENAÇÃO CRESCENTE

Dois fatos importantes a respeito dos métodos H1 a H11 podem ser destacados da análise. Inicialmente, percebe-se a grande inferioridade destes métodos se comparados ao HFC original. O que indica que uma abordagem baseada em médias e desvios padrão para a primeira fase pode não ser o caminho mais adequado ao se buscar melhorar o desempenho do método HFC.

O segundo fator importante a ser observado são os resultados semelhantes entre todos os métodos baseados em médias e desvios com ordenação crescente. Isso significa que, pelo menos para o funcionamento do algoritmo HFC, a proporção entre as médias e os desvios aplicados à primeira fase não impactam de maneira significativa o desempenho do método HFC como um todo. Assim sendo, não só os métodos baseados em médias e desvios não são o caminho para se melhorar o HFC, como as proporções não aparentam fazer diferença significativa para as soluções.

6.3 MÉTODOS BASEADOS NAS MÉDIAS E DESVIOS PADRÃO COM ORDENAÇÃO DECRESCENTE

Os resultados deste grupo de métodos, de forma geral, foram bastante semelhantes ao grupo anterior. As conclusões relacionadas à influência das médias e desvios no resultado destacadas, por se tratar de soluções semelhantes neste quesito, são aplicáveis para estes métodos também.

Quando comparados entre si, os métodos com ordenação crescente e decrescente apresentaram resultados significativamente próximos. O que significa que quando aplicado em conjunto com a terceira fase do HFC, a ordenação crescente ou decrescente de métodos baseados em médias e desvios tem pouca influência sobre os resultados.

6.4 MÉTODO BASEADO NA MÉDIA PONDERADA POR MÁQUINA DAS TAREFAS

A hipótese de que uma fase inicial desenvolvida a partir da ponderação dos tempos médios poderia gerar um melhor desempenho em comparação ao HFC original mostrou-se fraca quando se observaram os resultados obtidos para o método H23. Disso podemos concluir que a soma dos tempos das n tarefas para cada máquina não influencia de forma considerável a busca por um melhor resultado do algoritmo HFC. Logo, métodos semelhantes, baseados nestes mesmos critérios não aparentam ser as melhores opções ao se tentar obter resultados mais favoráveis.

6.5 MÉTODO HFC COM I MODIFICADO

O método H24 se destacou dos outros 23 testados anteriormente, pois não apenas foi o único a apresentar desempenho próximo ao método original, como ainda obteve resultado superior, sendo que forneceu o melhor resultado para 10 dos 12 problemas. Como dito anteriormente, a fase inicial do HFC original aparenta ser uma opção forte para se obter bons resultados. Devido a isto, não é de se estranhar que o único método proposto com uma solução superior ao original tenha justamente sido baseado numa ordenação inicial semelhante.

A hipótese de que é possível melhorar a ordenação original I_j por meio do aumento da variação de I_j para os casos em que as diferenças entre os tempos das tarefas são maiores se mostrou plausível. Dessa forma, é adequado sugerir que, baseado nos métodos testados neste projeto, tentativas que abordem a utilização mais específica de I_j, ao invés de utilização apenas de incrementos discretos, são a melhor opção para se seguir a pesquisa deste tema.

7 CONCLUSÃO

Neste trabalho foram determinados os seguintes objetivos: estudar métodos de *scheduling* em particular para problemas com ambiente *flowshop* e objetivo de minimização de *makespan*; e desenvolver alternativas à primeira fase do método HFC de Koulamas(1998) de forma a se obter uma solução com menor *makespan* que o método original.

Logo, inicialmente revisou-se alguns conceitos da teoria referente a planejamento e controle da produção, programação da produção e *scheduling*. Com os conhecimentos apresentados nesta revisão teórica, foram levantados métodos a serem implementados e testados de forma a atacar a possibilidade de se melhorar os resultados do HFC.

A maior parte dos métodos (H1 a H23) apresentam resultados longe de se tornarem viáveis devido ao fato de apresentarem soluções piores que o método original apenas para quase todos os problemas. Isto nos leva a recomendar que métodos de ordenação inicial que utilizem de maneira direta apenas critérios baseados em médias dos tempos das tarefas e desvios padrão, mesmo quando auxiliadas de uma ponderação baseada nos tempos totais das atividades em cada máquina, não sejam o foco de pesquisa ao se propor melhorias na primeira fase do HFC.

Já o método H24 nos permitiu observar algo importante sobre o desafio de se melhorar o desempenho do HFC. Ao mesmo tempo em que nos mostrou que há campo para melhoria dos resultados do método original, nos permitiu certo direcionamento a futuros estudos. Ou seja, espera-se que tentativas de otimização ligadas a um melhor ajuste da ordenação original já existente apresentem melhores resultados que outros métodos (ao menos se comparados a ordenações simples baseadas em médias e desvios padrão).

Sendo assim, uma sugestão de futura pesquisa seria testar diferentes ajustes “finos” aplicados à ordenação inicial original do HFC. Como por exemplo elevar os incrementos de I ao quadrado de forma a se aumentar a importância de grandes diferenças de tempos de execução de tarefas.

8 REFERÊNCIAS

BURBIDGE, J. L.; NEW, C. C. **Operations management, a systems approach through text and cases. London:** John Wilew, 1976.

CAMPBELL, H. G.; DUDEK, R. A.; SMITH, M. L. **Heuristic algorithm for *n* job, *m* machinesequencing problem.**Management Science Series B-Application, v. 16, n. 1, p. 630-637,1970.

FERNANDES, F. C. F. **Concepção de um sistema de controle da produção para manufatura celular.** 1991. Tese (Doutorado) – Escola de Engenharia de São Carlos. São Carlos.

FERNANDES, F.C.F; GODINHO FILHO, M. **Planejamento e Controle da Produção: Dos Fundamentos ao Essencial**. São Paulo, Editora Atlas, 2010.

FRAMINAN, J. M.; GUPTA, J. N. D.; LEISTEN, R. **A review and classification of heuristics for permutation flow shop scheduling with makespan objective.** Journal of the OperationalResearch Society, v. 55, n. 12, p. 1243-55, 2004.

GRAHAM, R. L.; LAWLER, E. L.; LENSTRA, J. K.; RINNOOY KAN, A. H. G. **Optimization and approximation in deterministic sequencing and scheduling: a survey.**Ann. Discrete Math, v. 4, p. 287-326, 1979.

JOHNSON, S. M. **Optimal two- and three-stage production schedules with setup times included.** Naval Research Logistics Quarterly, v. 1, p. 61-68, 1954.

KOULAMAS, C. **A new constructive heuristic for the flowshop scheduling problem.** European Journal of Operational Research, v. 105, p. 66-71, 1998.

MACCARTHY, B. L.; LIU, J. Y. **Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling.** International Journal of Production Research, London, v. 31, n. 1, p. 59-79, 1993.

MORTON, T.E.; PENTICO, D. **Heuristic Scheduling Systems**, New York: John Wiley, 1993.

NAWAZ, M.E.; ENSCORE JR.; HAM E. I.**A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem**, OMEGA 11 (1983) 91-95.

PALMER, D. **Sequencing jobs through a multi-stage process in the minimum total time – A quick method of obtaining a near optimum.** Operational Research Quarterly, v. 16, p .101-107, 1965.

PINEDO, M. **Theory, Algorithms, and Systems.** 3 ed. New Jersey, Prentice-Hall, Inc, 2008.

POTTS, C.N.; SHMOYS, D.B.; WILLIAMSON, D.P. **Permutation vs. non-permutation flow shop schedules**, Operations Research Letters 10 (1991) 281-284.

ROSSI, F, L. **Métodos Heurísticos para minimização da duração total da programação e do tempo total de fluxo em ambientes flow shop permutacional,** 2014.

SIPPER, D.; BULFIN JR., R. L. **Production planning, control and integration.** New York: McGraw-Hill, 1997.

SLACK, N. **Administração da produção.** 3 ed. São Paulo. Atlas, 2009.

TAILLARD, E. **Benchmarks for basic scheduling problems.** European Journal of Operational Research, v. 64, n. 2, p. 278-85, 1993.

9 ANEXOS

9.1 ANEXO I - MAIN.CPP

```
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <istream>
#include <string>
#include <cmath>
#include "procedimentos_auxiliares.h"
#include "ordenar.h"
#include "makespan.h"
#include "HFC.h"
#include <windows.h>

#include "time.h"

using namespace std;

int main(intargc, char *argv[])
{
int n, m;               // n = linha, m = coluna
int min1, min2;
intaux_Hlin, aux_Hcol;  //usado para alocação dinâmica
intHlin, Hcol;          //usado no principal
int aux;
int aux_lin1, aux_lin2;
intaux_ord;
int check=0;

LARGE_INTEGER tInicio, tFim, freq;

float tempo;
intms;
intopcao;

   n=500;
m=20;

int ****H2N = new int***[500];
for (inti=0; i<500; i++)
H2N[i]=new int**[n];

for (inti=0; i<500; i++)
for (int j=0; j<500; j++)
   H2N[i][j]=new int*[m];

for (inti=0; i<500; i++)
for (int j=0; j<500; j++)
for (int k=0; k<20; k++)
   H2N[i][j][k]=new int[m];

for (inti=0; i<500; i++)
for (int j=0; j<500; j++)
for (int k=0; k<20; k++)
for (int l=0; l<20; l++)
```

```
        H2N[i][j][k][l]=-10000;

int **ro = new int*[500];
int *Cmax = new int[n];
int *Cmax_2 = new int[n];
int **p = new int* [n];

double *meds= new double[500];
double *mmeds=new double[20];
double *dps= new double[500];
double **jobs= new double*[500];

for (inti=0; i<500; i++)
jobs[i]=new double[2];

for (inti=0; i<500; i++)
ro[i] = new int[20];

int **ro_2 = new int*[500];

for (inti=0;i<500;i++)
        ro_2[i] = new int [20];

for(inti = 0;i < n; i++)
p[i] = new int[m];

double **I = new double*[500];
for (inti=0; i<500; i++)
        I[i]= new double[2];

int aux2_Hlin=500*(500-1)/2;
int aux2_Hcol=20*(20-1)/2;


int **H = new int*[aux2_Hlin];
for(inti = 0;i < aux2_Hlin; i++)
        H[i] = new int[aux2_Hcol];

charBancoDados[13][50];
strcpy(BancoDados[0],"Taillard/Taillard 20 jobs 5 machines.txt");
strcpy(BancoDados[1],"Taillard/Taillard 20 jobs 10 machines.txt");
strcpy(BancoDados[2],"Taillard/Taillard 20 jobs 20 machines.txt");
strcpy(BancoDados[3],"Taillard/Taillard 50 jobs 5 machines.txt");
strcpy(BancoDados[4],"Taillard/Taillard 50 jobs 10 machines.txt");
strcpy(BancoDados[5],"Taillard/Taillard 50 jobs 20 machines.txt");
strcpy(BancoDados[6],"Taillard/Taillard 100 jobs 5 machines.txt");
strcpy(BancoDados[7],"Taillard/Taillard 100 jobs 10 machines.txt");
strcpy(BancoDados[8],"Taillard/Taillard 100 jobs 20 machines.txt");
strcpy(BancoDados[9],"Taillard/Taillard 200 jobs 10 machines.txt");
strcpy(BancoDados[10],"Taillard/Taillard 200 jobs 20 machines.txt");
strcpy(BancoDados[11],"Taillard/Taillard 500 jobs 20 machines.txt");
strcpy(BancoDados[12],"Taillard/Taillard 3 jobs 4 machines.txt");


ofstreamresultados;

resultados.open("RESULTADOS.txt");
```

```
        for(intopcao=0; opcao<=24; opcao++)
        {

        ifstreamarquivo;

        stringstr;


        for(intconj=0; conj<13;conj++)
        {
            //  system("PAUSE");
        str.clear();
        arquivo.open(BancoDados[conj]);
        for(intprob=0; prob<10; prob++)
              {

        getline(arquivo,str);
        arquivo>>n>>m;

        getline(arquivo,str);
        getline(arquivo,str);

        for (int i=0;i<m; i++)
        for (int j=0; j<n; j++)
        arquivo>>p[j][i];

        QueryPerformanceFrequency(&freq);
        QueryPerformanceCounter(&tInicio);
          //     cout<<" "<<tInicio.QuadPart*1000000/freq.QuadPart<<"\n";

        HFC(ro, ro_2, I, H, H2N, ms, p, n , m, meds, mmeds, dps, jobs, opcao);

        QueryPerformanceCounter(&tFim);

        tempo=(tFim.QuadPart - tInicio.QuadPart) * 1000000 / freq.QuadPart;   //tempo em
microsegundos

        getline(arquivo,str);

        resultados<<opcao<<"      "<<BancoDados[conj]<<"      "<<prob<<"      "<<tempo<<"
"<<ms<<"\n";
        /* for(inti=0; i<m; i++)
        {resultados<<"\n";
        for ( int j=0; j<n; j++)
        resultados<<ro[j][i]<<" ";}*/
              }

        arquivo.close();
        }

        }

        system("PAUSE");
        return EXIT_SUCCESS;
        }
```

9.2 ANEXO II - HFC.H

```
#include <cstdlib>
#include <iostream>
#include <cmath>
#include "procedimentos_auxiliares.h"
#include "ordenar.h"
#include "makespan.h"
#ifndef HFC_H_
#define HFC_H_

using namespace std;

void HFC(int** &ro, int** &ro_2, double** I, int** H, int**** H2N, int&ms, int** p,
int n, int m, double* meds, double* mmeds, double* dps, double** jobs, intopcao)
{
intHcol, Hlin, T1, T2, ms_2;
int a, b;
doublepeso_vmed, peso_dps, auxopcao;

for(inti=0;i<500;i++)
{
        I[i][1]=-1;
        I[i][0]=0;
   }

for (int i=0; i<n;i++)
{
for (int j=i+1; j<n; j++)
      {
for (int k=0; k<m; k++)
        {
for (int l=k+1; l<m; l++)
            {   a=min(p[i][k],p[j][l]);
                b=min(p[i][l],p[j][k]);
if (a<b)
                {
if(opcao==24)
{
I[i][0]=I[i][0]-(b-a);
                    I[j][0]=I[j][0]+(b-a);
                    }
else
                    {
                    I[i][0]=I[i][0]-1;
                    I[j][0]=I[j][0]+1;
                    }

                   H2N[i][j][k][l]=-1;

}
else
if (a>b)
                {
if(opcao==24)
{
I[i][0]=I[i][0]+(a-b);
                    I[j][0]=I[j][0]-(a-b);
```

```
                    }
else
                    {
                    I[i][0]=I[i][0]+1;
                    I[j][0]=I[j][0]-1;
                    }

                    H2N[i][j][k][l]=1;

}
            }
         }
      }
   }              //FINAL STEP 1

ordenar(I,n);

for (inti=0; i<n; i++)
{
for (int j=0; j<m; j++)
ro[i][j]=i;
}

if (opcao==0)
{
for (int i=0; i<n; i++)
jobs[i][1]=I[i][1];

ordenar (jobs, n);
   }
if (opcao>=1 &&opcao<=11)
   {
auxopcao=opcao;
peso_vmed=(11-auxopcao)/10;
peso_dps=(auxopcao-1)/10;
calcula_vmed(meds, p, n, m);
calcula_dps(dps, meds, p, n, m);

for(int i=0; i<n; i++)
jobs[i][0]=peso_vmed*meds[i]+peso_dps*dps[i];

ordenar (jobs, n);
   }

if (opcao>=12 &&opcao<=22)
   {
auxopcao=opcao;
peso_vmed=(22-auxopcao)/10;
peso_dps=(auxopcao-12)/10;

calcula_vmed(meds, p, n, m);
calcula_dps(dps, meds, p, n, m);

for(int i=0; i<n; i++)
jobs[i][0]=-peso_vmed*meds[i]-peso_dps*dps[i];

ordenar (jobs, n);
```

```
}
if (opcao==24)
   {
for (inti=0; i<n; i++)
jobs[i][1]=I[i][1];

ordenar (jobs, n);
}

if (opcao==23)
     {
calcula_mmed(mmeds, p, n, m);

calcula_vmedpond(meds, mmeds, p, n, m);

for(int i=0; i<n; i++)
jobs[i][0]=meds[i];

ordenar (jobs, n);

}
for (inti=0; i<n; i++)
{
for (int j=0; j<n; j++)
       {
if (jobs[j][1]==i)
ro[i][0]=j;
       }
    }

for (int i=0; i<n; i++)
for (int j=1; j<m; j++)
ro[i][j]=ro[i][0];

ms = makespan(p, ro, n, m);

for (int i=0; i<n-1; i++)
{
for (int j=1; j<m-1; j++)
       {
            T1=0;
for (int q=0; q<=j-1; q++)
{
for (int r=q+1; r<=j; r++)
               {

                        T1=T1+H2N[ro[i][r]][ro[i+1][r]][q][r];
} //next r
}  //next q


if (abs(T1)<j*(j+1)/2)
continue;  //next j

       T2=0;
for (int q=j+1; q<m; q++)
{
```

```
for (int r=q+1; r<m; r++)
          {

               T2=T2+H2N[ro[i][r]][ro[i+1][r]][q][r];
} //next r
       }     //next q

if (T2!=-T1)
continue;

for (int z=0; z<n; z++)
       {
for (int y=0; y<m; y++)
          {
            ro_2[z][y]=ro[z][y];
}
       }

for (int k = j+1; k<m; k++)
{
ro_2[i][k]=ro[i+1][k];
ro_2[i+1][k]=ro[i][k];
}
int soma=0, soma_2=0;
for (int d=0; d<m; d++)
for (int w=0; w<n; w++)
{
soma=soma+ro[w][d];
soma_2=soma_2+ro_2[w][d];
}

ms=makespan(p, ro, n, m);
ms_2=makespan(p, ro_2, n, m);

if(ms_2<ms)
{
ms=ms_2;
for (int d=0; d<m; d++)
for (int w=0; w<n; w++)
           {
ro[w][d]=ro_2[w][d];
           }
         }
       }
   }
}
#endif
```

FOLHA DE APROVAÇÃO

Autor: Alberto Koopman Ovando_____

Título: Avaliação de métodos de ordenação inicial para o problema de programação da produção em ambiente flowshop não permutacional com critérios de minimização do makespan_

_____

Trabalho de Conclusão de Curso defendido e aprovado

em_____/____/____,

com NOTA_____(___,___), pela comissão julgadora:

Assinatura_____

(Titulação/Nome/Instituição)

Assinatura_____

(Titulação/Nome/Instituição)

Assinatura_____

(Titulação/Nome/Instituição)

_____

Coordenador da Comissão de Coordenação do

Curso de Engenharia Produção