UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE ENGENHARIA MECÂNICA

ENGENHARIA MECÂNICA

ROBERTA CRISTINE YAMASHITA

SEQUENCIAMENTO DA PRODUÇÃO EM LINHAS FLOW SHOP PERMUTACIONAL COM BLOQUEIO E SEM ESTOQUE INTERMEDIÁRIO APLICANDO MÉTODOS HEURÍSTICOS DE III FASE

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO 2014

ROBERTA CRISTINE YAMASHITA

SEQUENCIAMENTO DA PRODUÇÃO EM LINHAS FLOW SHOP PERMUTACIONAL COM BLOQUEIO E SEM ESTOQUE INTERMEDIÁRIO APLICANDO MÉTODOS HEURÍSTICOS DE III FASE

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia Mecânica, do Departamento Acadêmico de Engenharia Mecânica da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Me. Maurício Iwama Takano

CORNÉLIO PROCÓPIO

2014

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Cornélio Procópio
Coordenação de Engenharia Mecânica – COEME

FOLHA DE APROVAÇÃO

SEQUENCIAMENTO DA PRODUÇÃO EM LINHAS FLOW SHOP PERMUTACIONAL COM BLOQUEIO E SEM ESTOQUE INTERMEDIÁRIO APLICANDO MÉTODOS HEURÍSTICOS DE III FASE

POR

ROBERTA CRISTINE YAMASHITA

Este trabalho de conclusão de curso foi apresentado às 08h20 do dia 01 de dezembro de 2014 como requisito parcial para obtenção do título de ENGENHEIRO MECÂNICO, linha de pesquisa – Física Nuclear, Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.
(Aprovado, aprovado com restrições, ou reprovado)

“A folha de aprovação assinada encontra-se na coordenação do curso.”

AGRADECIMENTOS

Agradeço a Deus, por iluminar sempre o meu caminho, me proteger, me amparar e me guiar nessa jornada. Agradeço a Ele também por iluminar minhas conquistas, e por dar a mim e a minha família o dom do amor. O que nos faz tão unidos e dedicados uns aos outros.

Agradeço aos meus amados pais, Roberto e Suely, por todo o apoio, pelos conselhos nas horas certas, pelos valores a mim ensinados, mas principalmente por todas às vezes ditas a mim “vá em frente, não desista”. Agradeço por terem feito do meu sonho, um sonho deles também.

As minhas irmãs, Karla e Karen, minhas amigas, companheiras, que sempre me incentivaram. Que sempre foram presentes na minha vida, e que me ajudaram tanto nessa trajetória. As quais são para mim, exemplo de força, perseverança e dedicação. Agradeço por toda preocupação e amizade, por todo abraço sincero e por toda a confiança.

Ao meu orientador, Professor Me. Maurício Iwama Takano, por ter acreditado no meu potencial e ter proporcionado a mim tantos ensinamentos. Por toda a paciência, disposição e dedicação ao me orientar. Por todo o seu tempo que foi dedicado a isto.

Ao meu fiel companheiro e amigo, Lucas, por ter me amparado e acompanhado com tanto amor e paciência durante essa batalha. Por me ouvir e por todo o carinho e cuidado dedicados.

A minha “mãedrinha”, Fátima, que me amparou no início dessa jornada e dedicou-se a mim com muito amor. Aos demais integrantes da casa do Senhor Toshiaki, meu amado avô, por toda a paciência em dividir o espaço comigo. A todos os meus tios e primos que sempre me deram todo o apoio e torceram pela minha vitória. Agradeço a cada um de vocês pelas orações, pelo carinho e pela torcida.

Aos meus amigos, cada um deles, mesmo sem citar os nomes, sabe a sua importância nesse longo caminho em busca deste sonho. Por todas as companhias

para estudos, noites mal dormidas, por dividirem comigo as minhas preocupações e as deles também. Por todas as risadas, estresses, mas principalmente pelo laço que criamos. Pelo amor que dedicamos um ao outro. Agradeço aos que ainda estão por perto, e também aos que estão longe, mas que se fazem presente da mesma forma. Cada um de vocês estará sempre no meu coração, onde quer que eu esteja.

A todos os excelentes professores e servidores, que fizeram parte da minha graduação. Agradeço pelos ensinamentos, pelas risadas e brincadeiras, e também pelas broncas. Cada palavra que ouvi de vocês me acrescentou alguma coisa boa, que levarei comigo sempre. E que me fizeram crescer de alguma forma.

A todos muito obrigada. Sem vocês eu jamais teria conquistado mais esse sonho.

RESUMO

YAMASHITA, Roberta C. Sequenciamento da produção em linhas flow shop permutacional com bloqueio e sem estoque intermediário aplicando métodos heurísticos de III Fase. (2014). Trabalho de Conclusão de Curso (Bacharelado em Engenharia Mecânica) – Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2014.

O problema de sequenciamento da produção pode ser resolvido através de diversas ferramentas, e a pesquisa operacional é um desses métodos. A pesquisa operacional tem como vantagem simplificar a visualização da realidade sem alterar a essência. O trabalho tem como objetivo analisar a eficiência de dois métodos heurísticos para a minimização do *makespan* e do tempo de fluxo, para um ambiente *flowshop* permutacional com bloqueio e sem estoque intermediário. Apresenta-se conceitos vindos de uma revisão feita na literatura sobre sequenciamento de produção, os métodos heurísticos apresentados e a classificação dos problemas. Dois métodos heurísticos de terceira fase são apresentados, tais métodos são avaliados em 120 problemas, divididos em 12 classes distintas, variando entre elas o número de tarefas e o número de máquinas. Os resultados são avaliados por meio dos desvios relativos de cada uma das funções objetivos. Também são analisados os tempos computacionais de cada heurística. Após as analises, percebe-se que as heurísticas PF_NEHls(x) e PW_NEHls(x) obtiveram bons resultados. Cada um desses métodos heurísticos apresenta melhoras em determinadas classes de problemas, para *makespan* e tempo de fluxo. Porém analisando a média dos desvios relativos médios, pode-se perceber que a heurística PW_NEHls(x) teve em média os melhores resultados para as funções objetivo estudadas no trabalho.

Palavras-chave: Bloqueio. *Flowshop* permutacional. *Makespan*. Tempo de fluxo

ABSTRACT

YAMASHITA, Roberta C. Scheduling of the production in permutation flow shop with blocking and no buffer by applying III phase heuristics methods. (2014). Course Conclusion Paper (Bachelors in Mechanical Engineering) – Federal University of Technology from Parana. Cornélio Procópio, 2014.

The production scheduling problem can be solved through various tools, and operational research is one of those methods. Operations research has the advantage of simplifying the reality display without changing the essence. The work aims to examine the efficiency of two heuristics methods to minimize the *makespan* and the flow time, for an environment permutation *flowshop* with blocking and no intermediate stock. It will be presented concepts coming from a revision made in the literature about production scheduling, presented heuristic methods and classification of problems. Two heuristic methods from the third phase are presented, such methods are evaluated in 120 issues, divided into 12 distinct classes, varying between them the number of jobs and the number of machines. The results are evaluated by means of the relative deviation of each of the objective functions. Computational times from each heuristic are also analyzed. After the analysis, it was found that the PF_NEHls (x) and PW_NEHls (x) have heuristics similar results. Each of these proposed methods offers improvements in certain classes of problems, for *makespan* and flow time. But analyzing the average of the mean relative deviations, one can see that the heuristic PW_NEHls (x) had on average the best results for the objective functions studied at work.

Keywords: Block. Permutation *flowshop*. *Makespan*. Flow Time

LISTA DE GRÁFICOS

LISTA DE SÍMBOLOS

α Indicador de restrição referente as configurações de máquinas

β Indicador de restrição adicional

λ Indicador do campo da função objetivo

Pm Máquinas em paralelo idênticas

Jm Job shop

FJc Job shop com máquinas múltiplas

Om Open shop

Fm Flow shop

$Fperm$ Flow shop permutacional

FFc Flow shop com máquinas múltiplas

m Número de máquinas

$prmu$ Permutação

$prmp$ Preempção

$Cmax$ Makespan

$\sum Cj$ Tempo de fluxo

n Número de tarefas

Cj,k Tempo de término da tarefa j na máquina k

Dj,k Tempo de saída da tarefa j na máquina k

Pj,k Tempo de processamento da tarefa j na máquina k

Dn,m Tempo de saída da última tarefa na última máquina

$\sigma j,k$ Função índice da heurística PF

wi Peso de uma tarefa em uma máquina i

$\chi j,k$ Equação utilizada para refletir o efeito da tarefa j em tarefas futuras

fj,k Função índice da heurística PW

v Tarefa artificial inserida em uma sequência parcial

βk Sequência parcial k

π Sequência criada através de uma solução inicial

λ Número de linhas da sequencia

DR Desvio relativo

Fx Solução encontrada pelo algoritmo

Fo Solução ótima para um determinado problema

TC Tempo computacional

Cnm *Makespan*

Mf Tempo de fluxo

SUMÁRIO

1. INTRODUÇÃO

Segundo GIGANTE(2010), a produção de produtos pode ser vista como um conjunto de processos que transformam insumos em produtos.

O departamento de Planejamento e Controle da Produção (PCP) é responsável pela execução controle e planejamento das atividades, tendo que garantir a produção de produtos certos, na quantidade certa e no tempo certo. Ou seja, garantir que os produtos saiam da forma planejada e com a qualidade esperada.

O departamento de PCP tem a necessidade de conciliar volume de produção e tempo, sendo assim, precisa conciliar mais três atividades, o carregamento, a sequência e a programação.

O carregamento é visto como a quantidade de trabalho alocada em um centro, a sequência é a decisão sobre em qual ordem os processos devem ser realizados, e a programação é a decisão de quantidade de produtos, início e término de produção, equipamentos utilizados (GIGANTE, 2010).

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo do trabalho é identificar qual o melhor método heurístico de terceira fase capaz de minimizar o tempo médio de fluxo e o *makespan* (tempo de término da última tarefa) em uma linha *flow shop* permutacional com bloqueio em problemas de pequeno, médio e grande porte.

1.1.2 Objetivo Específico

Para se alcançar o objetivo geral serão necessários:

- Adaptar os melhores métodos heurísticos de problemas em linhas *flow shop* permutacional com bloqueio para minimização do *makespan* para problemas em linhas *flow shop* permutacional com bloqueio para minimização do tempo de fluxo;
- Programar, utilizando *Matlab*, os métodos heurísticos com a função de minimizar o *makespan*;
- Programar, utilizando *Matlab*, os métodos heurísticos com a função de minimizar o tempo de fluxo;
- Analisar qual método é mais eficiente para cada função objetivo (minimização do tempo de fluxo e minimização do *makespan*);
- Avaliar o tempo de processamento dos métodos para cada função objetivo.

1.2 JUSTIFICATIVA

A importância de todo processo de melhoria da produção deve-se principalmente ao alto nível de competitividade nas indústrias, grande parte devido à globalização. A redução do *makespan* e também do tempo de fluxo aumentam a capacidade de produção da fábrica e melhora a utilização dos equipamentos. Desta forma os objetivos auxiliam na redução dos custos de fabricação, tornando-a mais competitiva no mercado global (PINEDO, 2008).

2. SEQUENCIAMENTO DE PRODUÇÃO

Segundo PINEDO (2008), qualquer problema de sequenciamento da produção pode ser definido por três identificadores (α | β | γ), sendo eles:

- α- representa a configuração das máquinas - pode conter apenas uma entrada;
- β- restrições adicionais - pode conter nenhuma, uma ou múltiplas entradas;
- γ- função objetivo - pode conter uma ou múltiplas entradas.

No campo α temos problemas de programação pelo fluxo das tarefas de cada máquina. As configurações das máquinas podem ter a seguinte classificação:

- Máquina única (1): Há apenas uma única máquina para o processamento das tarefas;
- Máquinas idênticas em paralelo (Pm): Há uma ou mais estações de trabalho, cada uma possui mais de uma máquina idêntica em paralelo. Cada máquina pode executar qualquer tarefa da estação de trabalho, porém apenas uma das máquinas é necessária por tarefa em cada estação de trabalho.
- *Flow shop* (Fm): Máquinas em série sendo que todas as tarefas seguem o mesmo fluxo;
- *Flow shop* flexível (FFc): Estações de trabalho com máquinas paralelas, a quantidade de máquinas pode variar em cada estação. Cada tarefa é processada em apenas uma máquina em cada estação de trabalho, podendo ser processadas em múltiplos estágios seguindo uma mesma ordem;

Job shop (Jm): Cada tarefa tem seu próprio fluxo de processamento nas máquinas;

- Job shop flexível (FJc): Um job shop onde existem idênticas máquinas por estação, e em cada estação de trabalho as tarefas são processadas por apenas uma máquina;
- Open shop (OM): Não há restrições de fluxos para o processamento das tarefas.

No campo β podem ser adicionadas, entre outras, as seguintes restrições adicionais:

- Quebra (brkdwn): Quando esta restrição é adicionada ao problema significa que uma máquina pode não estar disponível em tempo integral. O período não disponível da máquina pode ser, por exemplo, por parada para manutenção ou por troca de turno. Muitas vezes é referenciado como restrição de disponibilidades das máquinas;
- Permutação (prmu): Uma restrição de flow shop que funciona de acordo com a teoria do *First In First Out* (FIFO) - 'Primeiro que entra Primeiro que sai', ou seja, as tarefas são realizadas de acordo com a ordem de chegada. A ordem de processamento das tarefas se mantém a mesma da primeira até a última máquina;
- Bloqueio (block): O bloqueio ocorre se uma tarefa terminar de ser processada em uma máquina, e o buffer (ou estoque intermediário) da máquina seguinte estiver lotado. Desta forma tarefa fica impedida de sair da máquina onde ocorreu o processamento, bloqueando-a de iniciar o processamento das tarefas seguintes. Isso também pode ocorre quando o buffer é igual a zero, ou seja, não há estoques intermediários entre as máquinas.

No campo γ, pode-se citar como exemplos de função objetivo:

- *Makespan*: Consiste em minimizar o tempo de término do último trabalho, aumentando assim a produtividade;
- Tempo de fluxo: Minimização da soma dos tempos de término das tarefas;
- Tempo de fluxo ponderado: Minimização da soma ponderada dos tempos de término das tarefas. O peso atribuído a cada tarefa pode representar, por exemplo, o custo de manter uma tarefa em estoque intermediário.

Os problemas estudados serão **Fm | perm,bloq | Cmax** e **Fm | perm,bloq |** $\sum C_j$.

Segundo MOCCELLIN (2005) um problema de programação pode ser definido como um conjunto de *n* tarefas (J1, J2, J3,..., Jj,..., Jn) que devem ser processadas em *m* (número de máquinas no problema) máquinas (M1, M2, M3,..., Mk,..., Mm).

No *flow shop* permutacional os processamentos das tarefas seguem uma mesma ordem em todas as maquinas.

Neste trabalho consideramos que as tarefas já estejam prontas para serem executadas no instante zero, ou seja, inclui-se o setup no tempo de processamento, e as operações não podem ser interrompidas.

O makespan é o tempo em que a última tarefa sai da última máquina. Já o tempo de fluxo, é a somatória dos tempos em que as tarefas saem da última máquina. Esses conceitos são melhor explicados na figura 1.

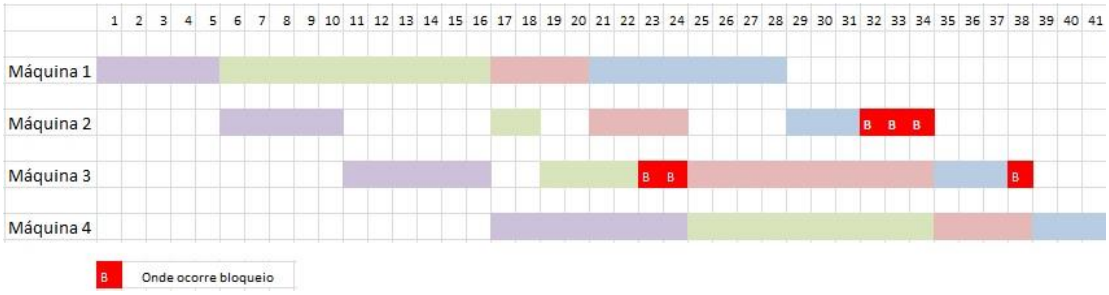

Figura 1- Funcionamento do bloqueio com buffer zero e sem estoque intermediário

Como exemplificado na figura 1, o *makespan* é igual a 41. Enquanto o tempo de fluxo é a soma de 24+34+38+41=137. Considera-se um processo sem entoque intermediário, e a sequência de processamento das tarefas segue uma mesma ordem em todas as máquinas.

2.1 CÁLCULO DO *MAKESPAN*

O cálculo do *makespan* avaliará o tempo que as tarefas demoram para ser processadas nas máquinas juntamente com o tempo que elas aguardam nas máquinas, isso porque o tempo em que a tarefa termina de ser processada na máquina não coincide necessariamente com o tempo em que ela passará para a próxima máquina.

O tempo de *setup,* que é o tempo utilizado para preparar a máquina ou um processo, neste caso está incluído no tempo de processamento.

Apresentamos um problema com *m* máquinas e *n* tarefas, no qual o estoque intermediário é igual a zero, onde P_{jk} é o tempo necessário de processamento da tarefa *j* na máquina *k*, e D_{jk} é o tempo em que a tarefa *j* sai da máquina *k*. A partir daí serão apresentadas as equações necessárias para o cálculo dos tempos de entrada e saída de cada operação.

Considera-se que o tempo de saída na primeira máquina é igual ao tempo de processamento da primeira tarefa nesta máquina:

$$D_{1,1} = P_{1,1} \quad (1)$$

Nas máquinas seguintes, o tempo de saída das primeiras tarefas será a soma do tempo de término na máquina anterior ao tempo de processamento da máquina atual, já que não existe a necessidade da primeira tarefa aguardar em uma máquina, pois não existem tarefas anteriores:

$$D_{1,K} = D_{1,K-1} + P_{1,K} \quad (2)$$

Para as tarefas seguintes temos dois casos. Se não existir bloqueio para a próxima tarefa, então o tempo de saída para cada próxima tarefa na primeira máquina será a soma do tempo de saída da tarefa anterior com o tempo de processamento da atual tarefa na primeira máquina. Porém se existir bloqueio, ou seja, se a tarefa que está sendo executada terminar, e a próxima máquina ainda estiver ocupada, o tempo de saída da tarefa que está em andamento será o mesmo tempo em que a tarefa anterior sair da segunda máquina:

$$D_{J,1} = máx\ (D_{J-1,1} + P_{J,1};\ D_{J-1,2}) \quad (3)$$

No caso das tarefas dois até *n* nas máquinas dois até *m-1*, se não houver bloqueio, o tempo de saída dessas tarefas serão a soma do tempo de saída da atual tarefa na máquina anterior com o tempo de processamento da atual tarefa na máquina atual. Porém quando existir bloqueio, o tempo de saída da tarefa atual será o tempo de saída da tarefa anterior na máquina posterior:

$$D_{j,k} = máx\ (D_{j,k-1} + P_{j,k};\ D_{J-1,k+1}) \quad (4)$$

Agora, considerando-se a última máquina, calcula-se o tempo de saída somando o tempo de saída da tarefa atual na máquina anterior com o tempo de

processamento da atual tarefa nessa última máquina, já que não existem bloqueios nessas posições por não existirem máquinas posteriores:

$$D_{j,m} = D_{j,k-1} + P_{j,k} \quad (5)$$

Tendo definidos todos os tempos de início e término das tarefas podemos calcular o *makespan*, que será o tempo que a última tarefa sair da última máquina:

$$D_{n,m} \quad (6)$$

2.2 CÁLCULO DO TEMPO DE FLUXO

Como foi demonstrado anteriormente calculou-se o tempo de início e término de todas as tarefas em todas as máquinas. Obteve-se uma matriz com os tempos de saída. Sendo assim, podemos calcular o tempo de fluxo, que é a soma do tempo de saída de todas as tarefas na última máquina.

$$\sum_{q=1}^{n} Dq, m \quad (7)$$

2.3 PESQUISA OPERACIONAL

Para resolver um problema de sequenciamento da produção podem ser utilizados diferentes métodos. A pesquisa operacional (PO) é uma ferramenta que pode ser utilizada para esse fim.

A pesquisa operacional é uma área de conhecimento da engenharia de produção que disponibiliza alguns métodos matemáticos, geralmente computacionais, para resolver problemas de tomada de decisão reais (BOIKO *et al.*, 2010). A pesquisa operacional tem como vantagem simplificar a visualização da realidade sem alterar a essência, possibilitar o entendimento de relações complexas, servir como base para aprimorar parâmetros, porém tem como desvantagens as limitações de determinação de todas as relevantes variáveis que influenciam na situação estudada e dificuldades de entendimento entre o provedor e o usuário (SILVA *et al.*, 2013)**.**

A modelagem matemática de um método exato apresenta soluções ótimas, porém é um método complexo e de tempo computacional elevado. Já os métodos heurísticos possuem um tempo computacional menor e também apresenta bons resultados.

2.3.1 MÉTODOS EXATOS

A modelagem matemática de um método exato pode ser classificado como, um problema de Programação Linear, Programação Inteira, Programação Linear-Inteira Mista, Programação Não-Linear e Programação Não-Linear Inteira Mista.

• Problemas envolvendo **Programação Linear** (LP) são problemas nos quais as expressões matemáticas (representando a função objetivo e as restrições) envolvidas são apenas lineares e onde todas as variáveis podem assumir valores não inteiros reais;

• Problemas envolvendo **Programação Inteira** (IP) são problemas onde todas as variáveis de decisão só podem assumir valores inteiros e as expressões matemáticas são lineares;

• Problemas envolvendo **Programação Linear-Inteira Mista** (MILP) são problemas similares aos de programação inteira, mas onde algumas variáveis de decisão podem assumir valores não inteiros;

• Problemas envolvendo **Programação Não-Linear** (NLP) são problemas que possuem ao menos uma expressão matemática (a função objetivo ou alguma das restrições) não linear (i.e., com produto de variáveis ou variáveis com expoente) e onde todas as variáveis podem assumir valores não inteiros reais;

• Problemas envolvendo **Programação Não-Linear Inteira Mista** (MINLP) são problemas similares à Programação Não-Linear, mas que possuem variáveis de decisão que devem assumir valores inteiros e outras variáveis de decisão que podem assumir valores não inteiros (HILLIER E LIEBERMAN, 1995).

2.3.2 MÉTODOS HEURÍSTICOS

Segundo FRAMINAN, GUPTA e LEISTEN (2004) é importante que haja uma classificação das soluções e uma busca constante para melhorar os resultados, já que existem diversas heurísticas disponíveis com diversos jeitos de abordagens do problema. Desta forma, faz-se uma divisão que consiste em três fases:

- Fase I: Desenvolvimento do índice;
- Fase II: Construção da solução;
- Fase III: Melhoria da solução.

Na primeira fase os trabalhos são organizados baseados no tempo de processamento de cada tarefa em cada máquina. De modo geral nesta fase não é necessário fazer qualquer suposição sobre a ordem dos trabalhos nas sequencias.

Na segunda fase, testam-se mudanças na solução inicial para conseguir melhorias, atingindo assim a construção da solução. Para isso é necessária uma solução inicial vinda da primeira fase.

Na terceira fase melhora-se uma solução já existente, obtida na primeira fase ou na segunda fase por meio de algum procedimento, fazendo com que a qualidade da solução seja igual ou melhor que a qualidade da solução inicial. A partir da solução de entrada, o procedimento procura uma alteração de sequência para melhorar a solução. Geralmente as abordagens de melhorias são classificadas como pesquisas locais, ou meta heurísticas.

3 O PROBLEMA DE SEQUENCIAMENTO EM AMBIENTE *FLOW SHOP* COM BLOQUEIO

Neste capítulo serão apresentados alguns trabalhos direcionados ao problema específico. Propostas de diversos algoritmos utilizados para solucionar o problema proposto.

WANG *et al*. (2012) propuseram um método para solução de problemas em ambientes *flowshop* sem capacidade de armazenamento intermédio dividido em três fases: 1. a utilização de uma nova regra de prioridade para gerar uma solução inicial para o problema; 2. um algoritmo Nawaz-Enscore-Ham (*NEH*) modificado; e 3. uma fase de melhoria da solução que utiliza um *simulated annealing* modificado que incorpora a busca local, reduzindo o esforço computacional. Em geral, esse novo algoritmo produz resultados melhores que os algoritmos existentes, por exigir menos esforços computacionais.

RONCONI (2003), propôs um método para o problema *flowshop* com bloqueio entre as maquinas que utiliza uma heurística chamada *MinMax* (*MM*). Propõe-se um limite inferior sobre o tempo de conclusão de uma determinada tarefa. Apesar dos bons resultados obtidos a heurística proposta foi superada pela heurística PF.

PAN e WANG (2011), propuseram um melhoramento nas heurísticas construtivas PFE, MME, wPFE e PWE. Criando as heurísticas PF-NEH(x), wPF-NEH(x), MMEls, PW-NEH(x), PW-NEHls(x) e PF-NEHls(x). As heurísticas PF_NEH(x)ls e PW_NEH(x)ls são consideravelmente melhores que as demais. Esse fato se justifica pelo fato de que essas heurísticas usam como solução inicial as heurísticas de segunda fase que também apresentaram melhores resultados.

WANG e TANG (2011) propuseram um método para o problema de programação clássica de *flowshop* permutacional, utilizando algoritmos com métodos exatos, heurísticos e metaheuristicos para esse tipo de problema. A proposta do algoritmo *DPSO* para o problema apresenta uma eficiência computacional.

WANG *et al*. (2011), propuseram um método baseado na busca hibrida harmônica, com o objetivo de encontrar uma programação na qual a ordem de processamento das tarefas são as mesmas em cada máquina e o *makespan* é mínimo.

WANG *et al*. (2009), propuseram um algoritmo proposto HDDE programado para equilibrar a exploração e aproveitamento. Um problema dependente de busca local é apresentado e aplicado a um processo individual U com uma probabilidade P para aumentar a capacidade de busca local.

HAN *et al*.(2011), propuseram o algoritmo IABC, colônia de abelhas artificial, que baseia-se em outros algoritmos (MM, NEH, NEH_WPT e MM_B) para gerar as tarefas iniciais. Então são executados os 4 programas para que estes gerem soluções iniciais, gerando uma busca nas soluções e armazenando as que forem diferentes.

RIBAS *et al*. (2010), propuseram em uma interação gulosa do algoritmo para *flow shop* com bloqueio, em que objetivo é fornecer um melhor resultado baseado na heurística NEH e um algoritmo IG muito competitivo para o BFSP.

TRABELSI *et al*. (2012), propuseram heusrísticas e metaheurísticas para bloqueios mistos com objetivo de maximizar a produtividade de uma empresa. Melhora-se o NEH a nivel local e faz-se comparação com a TSS, heuristica proposta.

LIN e YING (2012),propuseram o algoritmo *Revised Artificial Immune System* com objetivo de encontrar uma sequência de processamento de cada tarefa em todas as máquinas, para que o makespan seja minimizado. Utiliza-se o algoritmo RAIS para tal finalidade, o qual tem um resultado computacional favorável.

CARAFFA *et al*. (1999) propuseram um algorítimo genético para resolver o problema de *flow shop* permutacional. A partir de uma dada sequência de tarefas, utilizando o conceito de retardar certas operações, faz-se a busca da melhor sequência.

DAVENDRA e DAVENDRA (2012) propuseram o algoritmo de auto-organização baseado no comportamento competitivo usado para resolver um problema comum. Como o algoritmo usa a filosofia de competição e cooperação, pode-se dizer que ele traz resultados muito satisfatórios.

GRABOWSKI e PEMPERA (2005) propuseram o método de pesquisa tabu, que é um dos mais eficazes estudados. Consiste em, a partir de uma permutação inicial, fazer uma busca local gerando um conjunto de permutações pelos movimentos. E repetindo o processo, começando pela permutação com melhor makespan até o resultado final.

HAN&J *et al*. (2012), propuseram três algoritmos baseados em colônias de abelhas os quais são testados para minimização de *flowtime* total, hDABC1 , hDABC2, hDABC3. Os tres algoritmos são gerados através de busca local. O resultado final

mostra que essa proposta é muito eficaz para problemas de *flowshop* com bloqueio, cujo objetivo é minimizar o tempo de fluxo total.

WANG *et al*. (2010), propuseram o algoritmo de busca hibrida harmônica para ser utilizado com o objetivo de minimizar o tempo de fluxo total. Esse método é basicamente baseado em um algoritmo HS, o qual é inspirado em performance musical.

HAN *et al*. (2012), propuseram um algoritmo baseado em colônias de abelhas, que baseia-se em outros algoritmos (MM, NEH, NEH_WPT e MM_B) para gerar as tarefas iniciais, e resolver problemas de *flowshop* permutacional. Obtendo através dessa programação resultados satisfatórios.

Uma análise dos trabalhos apresentados permite verificar que as heurísticas construtivas PW-NEHls(x) e PF-NEHls(x) são, atualmente, as melhores heurísticas para solucionar problemas de minimização do *makespan* em ambientes *flow-shop* permutacional com bloqueio. O objetivo deste trabalho é verificar se essas mesmas heurísticas podem ser utilizadas para problemas de minimização do tempo de fluxo e analisar, entre elas, qual é a que obtem os melhores resultados.

4 MÉTODOS HEURÍSTICOS DE III FASE

4.1 Heurística PF_NEHls(x)

A heurística PF_NEHls(x), é baseada em quatro passos.

O primeiro passo é gerar uma sequência de tarefas $\alpha=(\alpha_1,\alpha_2,...,\alpha_n)$, a partir de uma ordem crescente dos tempos totais de processamento.

O segundo passo é feito para $L=1$ até $L=x$. Neste passo α_1 é a primeira tarefa e a partir disso gera-se uma sequência $\beta=(\beta_1,\beta_2,...,\beta_n)$ utilizando a heurística PF. Nessa etapa é construída a sequência $\pi_L=(\beta_1,\beta_2,...,\beta_n)$.

A heurística PF foi criada por McCormick et al., PF (*profile fitting*) é uma heurística construtiva simples. Baseia-se em sequenciar as tarefas para minimizar ao máximo a soma do tempo ocioso e o tempo de bloqueio da máquina. Utiliza-se duas regras, a partir de uma sequência parcial de tarefas $\beta=(\beta_1, \beta_2,..., \beta_k)$.

A primeira regra é selecionar a tarefa com menor de processamento para ser a primeira tarefa na sequência parcial β. A segunda é testar todas as tarefas não selecionadas, para a próxima posição da sequência, e optar pela tarefa que retornar o menor valor da expressão (8). Nestes testes levam em consideração os tempos em que a máquina fica parada sem nenhuma tarefa e os bloqueios.

$$\sigma_{j,k} = \sum_{i=1}^{m} (D_{k+1,i} - D_{k,i} - P_{j,i}) \quad (8)$$

O terceiro passo é para $k=n-\lambda+1$ até $k=n$, usa-se o procedimento da heurística NEH. Heurística NEH

NEH, foi uma heurística proposta por NAWAZ, ENSCORE JR. e HAM, que segundo PAN e WANG (2011) tem o reconhecimento de melhor método quando o critério é a minimização de *makespan* em problemas de *flow shop* permutacional. É uma heurística composta por duas fases.

Na primeira fase, forma-se uma sequência base onde as tarefas são ordenadas de acordo com o tempo de total processamento, que é o tempo de uma tarefa em todas as máquinas, em ordem decrescente. Essa regra é conhecida como LPT (*Longest Process Time*).

Na segunda fase a primeira tarefa da sequência gerada é selecionada e para $i=1,2,...,n$, gera-se sequências parciais alocando a i-ésima tarefa em todas as outras possíveis posições na sequência parcial. A partir deste teste, será selecionada a

sequência que obtiver o menor valor de *makespan* para ser a próxima sequência parcial.

A tarefa β_k será testada em todas as possíveis posições de π_L. Em seguida a tarefa β_k será alocada na posição de π_L que retornar os menores valores.

No quarto, e último passo, após gerada a sequência π_L, a heurística RLS, de busca local, realiza a pesquisa para π_L e retorna uma sequência final $\pi \in (\pi_1,\pi_2,...,\pi_x)$ com o mínimo valor de *makespan*.

Segundo PAN E WANG (2011), nas heurísticas compostas a busca local é utilizada para obtenção de uma melhoria, ou seja, otimizar uma solução gerada por uma heurística construtiva.

O RLS (*referenced local search*), é um exemplo de heurística de busca local. Considerando $\pi=(\pi_1,\pi_2,...,\pi_n)$ uma sequência a ser melhorada e $\pi=(\pi_{ref1},\pi_{ref2},...,\pi_{refn})$ a sequência referência, o procedimento adotado pela heurística segue os seguintes passo.

Primeiro passo calcula-se o tempo de início de todas as estações de trabalho, o tempo de início da última tarefa e o tempo de término das tarefas. Esse procedimento acontece do fim para o início da sequência.

Segundo passo, gera-se uma sequência β', inserindo tarefas para posição da tarefa *j k'=1,2,...,k+1*. Os tempos de início da tarefa *j* são calculados utilizando os tempos de início que são previamente calculados. E o *makespan* é calculado a partir da soma do tempo máximo de início da tarefa *j* e o tempo de término da tarefa k' da sequência β da mesma máquina.

4.2 Heurística PW_NEHls(x)

Da mesma forma que é feita na heurística PF_NEHls(x), a heurística PW_NEHls(x), é baseada em quatro passos.

O primeiro passo é gerar uma sequência de tarefas $\alpha=(\alpha_1,\alpha_2,...,\alpha_n)$, a partir de uma ordem crescente dos tempos totais de processamento.

O segundo passo é feito para *L=1* até *L=x.* Neste passo α_1 é a primeira tarefa e a partir disso gera-se uma sequência $\beta=(\beta_1,\beta_2,...,\beta_n)$ utilizando a heurística PW. Nessa etapa é construída a sequência $\pi_L=(\beta_1,\beta_2,...,\beta_n)$.

A heurística PW criada por LIU AND REEVES (2001), analisa o tempo ocioso e o bloqueio não somente da tarefa que está sendo executada, como no caso da heurística PF. Esta analisa também o bloqueio e o tempo ocioso das tarefas posteriores.

O primeiro passo é calcular o tempo de término e a soma dos tempos ociosos e de bloqueio da tarefa *v* que será inserida na sequência parcial β, utilizando um peso w_i. Essa tarefa *v* é uma tarefa artificial que é inserida na sequência β. Calcula-se então a partir das equações:

$$Pv,i = \sum_{i=1}^{m} \frac{Pq,i}{n-k-1} \qquad q \in U \qquad q \neq j \tag{9}$$

$$\chi_{j,k} = \sum_{i=1}^{m} wi\ (D_{[k+2],i} - D_{[k+1],1} - P_{v,i}) \tag{10}$$

A regra principal que rege a sequência de processamento é dada pela equação:

$$f_{j,k} = (n - k - 2) * \delta_{j,k} + \chi_{j,k} \tag{11}$$

Onde *(n – k – 2)* é usado para fazer o balanceamento do tempo ocioso e o tempo de bloqueio da tarefa *j* e o que essa tarefa causa das próximas.

Segundo PAN e WANG (2011), os três passos para realizar a heurística PW são:

1° passo: Selecionar como primeira tarefa da sequência a que tiver o menor valor para a expressão $f_{j,0}$. Caso exista valores coincidentes, seleciona-se a que apresentar menor valor para a expressão $\chi_{j,0}$.

2° passo: k=1 até k=n-2

Calcula-se o tempo de término da última tarefa da sequência β. Para cada tarefa j∈ U, calcula-se o tempo de processamento $p_{v,i}$ para as tarefas artificiais *v*. Além de fazer o cálculo do tempo de término tanto da tarefa *j* quanto da *v*. A partir daí seleciona-se a tarefa que apresentar o menor valor para a expressão .

3° passo: faz-se a última tarefa de U sendo a n*ésima* tarefa de β.

O terceiro passo é para $k=n-\lambda+1$ até $k=n$, usa-se o procedimento da heurística NEH.

A tarefa β_k será testada em todas as possíveis posições de π_L. Em seguida a tarefa β_k será alocada na posição de π_L que retornar os menores valores.

No quarto, e último passo, após gerada a sequência π_L, a heurística RLS, de busca local, realiza a pesquisa para π_L e retorna uma sequência final $\pi \in (\pi_1, \pi_2, ..., \pi_x)$ com o mínimo valor de *makespan*.

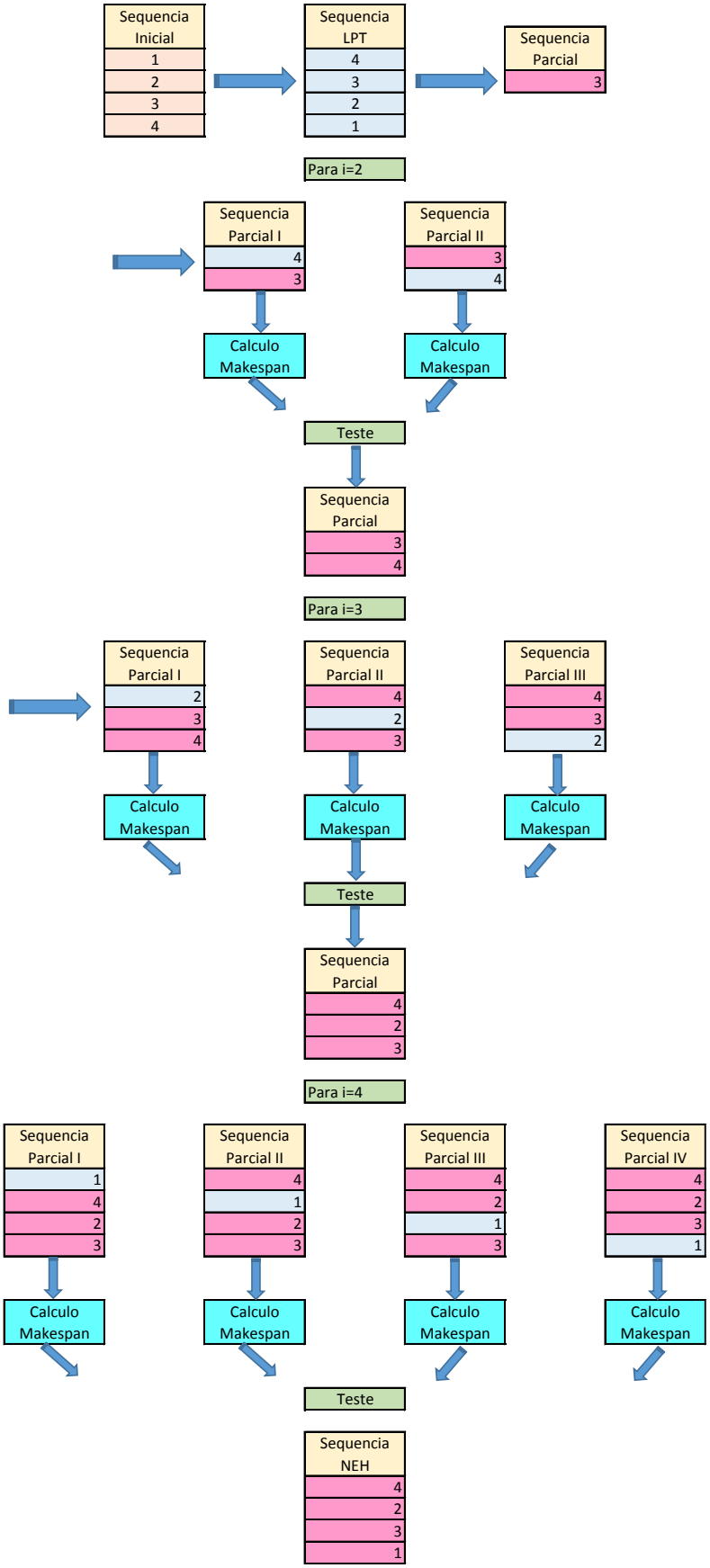

Figura 2- Heurística NEH para 4 tarefas

5. METODOLOGIA

Os resultados apresentados são obtidos através de experimentação computacional. As heurísticas foram implementadas utilizando o software *Matlab*. Os resultados foram avaliados através do cálculo do desvio médio relativo. Na implementação, todas as heurísticas recebem como único dado uma matriz com de tempos de processamento. E retornam ao usuário o resultado da minimização do *makespan* e do tempo de fluxo.

Nas heurísticas PF_NEHls(x) e PW_NEHls(x) foram assumidos valores de 5 e 20 para os coeficientes x e λ respectivamente.

O desvio relativo é o valor de variação correspondente a melhor solução encontrada pelos métodos. Quando o desvio relativo é igual a zero, quer dizer que o método forneceu o melhor valor para o problema. E quanto menor o valor do desvio relativo, mais próximo o resultado do método chegou da melhor solução. O desvio relativo foi calculado pela equação:

$$DR = \frac{Fx - Fo}{Fo} \quad (12)$$

Onde:

Fx: solução do método

Fo: melhor solução do método

As heurísticas foram implementadas utilizando o *software Matlab*. A avaliação das heurísticas foi submetida a uma base de dados conhecida de problemas de linhas *flow shop* com 120 problemas (TAILLARD,1992). Essa base de dados é composta por 120 matrizes que se dividem em 12 classificações de problemas, que variam de acordo com o número de tarefas e máquinas.

Neste trabalho os valores de tempo de processamento das tarefas nas máquinas serão as matrizes de cada classe de problema. Cada uma dessas classes possui 10 matrizes, as quais são compostas por valores aleatórios de zero a 99.

São 12 classes, que foram divididas variando o número de tarefas e máquinas respectivamente, sendo as classes: 20x5, 20x10, 20x20, 50x5, 50x10, 50x20, 100x5, 100x10, 100x20, 200x10, 200x20 e 500x20 (tarefas x máquinas)

Para fazer a análise dos melhores resultados, foi utilizado o Microsoft Excel.

6. RESULTADOS

Os gráficos a seguir, mostram a eficiência das heurísticas em cada classe.

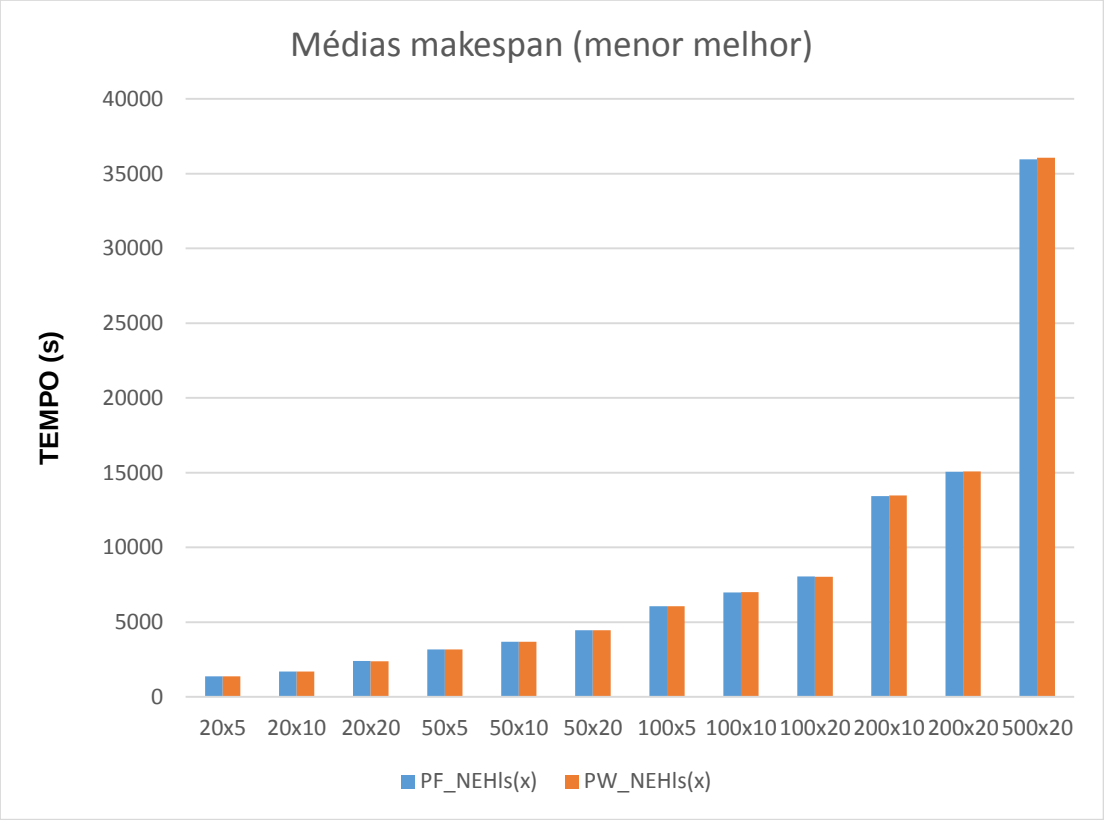

Gráfico1- Médias dos resultados do *makespan*

O gráfico 1, apresenta a média do *makespan*. Como pode ser observado, em média as heurísticas tiveram um resultado de eficiência parecidos, quando foram avaliados quanto à minimização do *makespan*.

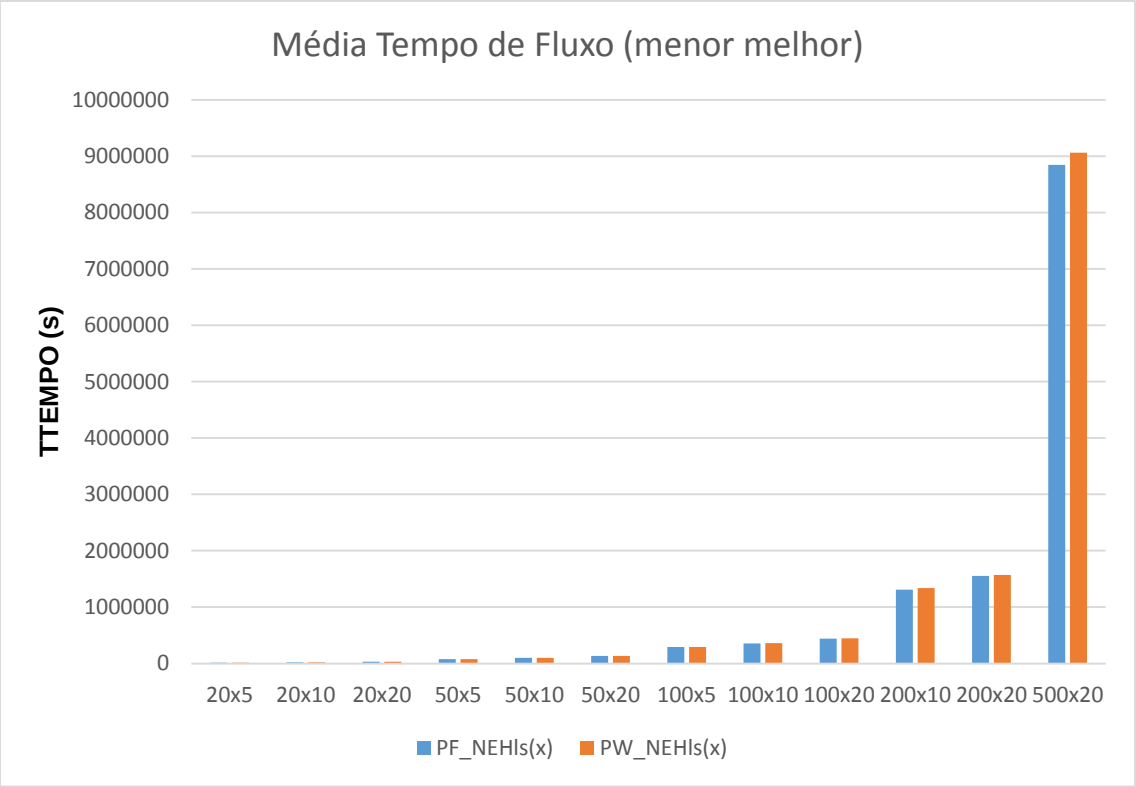

Gráfico 2- Médias dos resultados do tempo de fluxo

O gráfico 2, apresenta a média do tempo de fluxo. Também observa-se que, em média, as heurísticas apresentaram um valor muito próximo quando trata-se de minimização de tempo de fluxo. Na classe de 500 problemas e 20 máquinas, a heurística PF apresenta um resultado melhor mais considerável.

Também foi avaliado neste trabalho o tempo computacional das classes de problemas de cada heurística. Os gráficos 3 e 4, apresentam a média desses resultados para *makespan* e tempo de fluxo respectivamente.

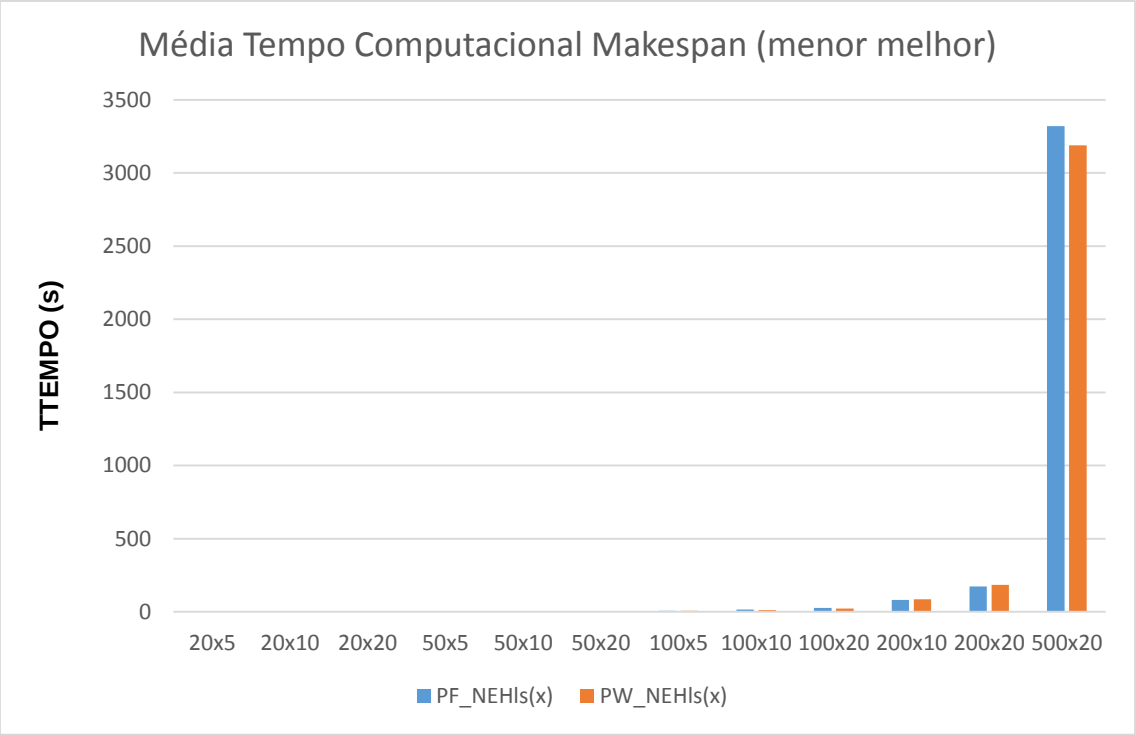

Gráfico 3- Média do tempo computacional para solucionar problemas com o objetivo de minimização do *Makespan*

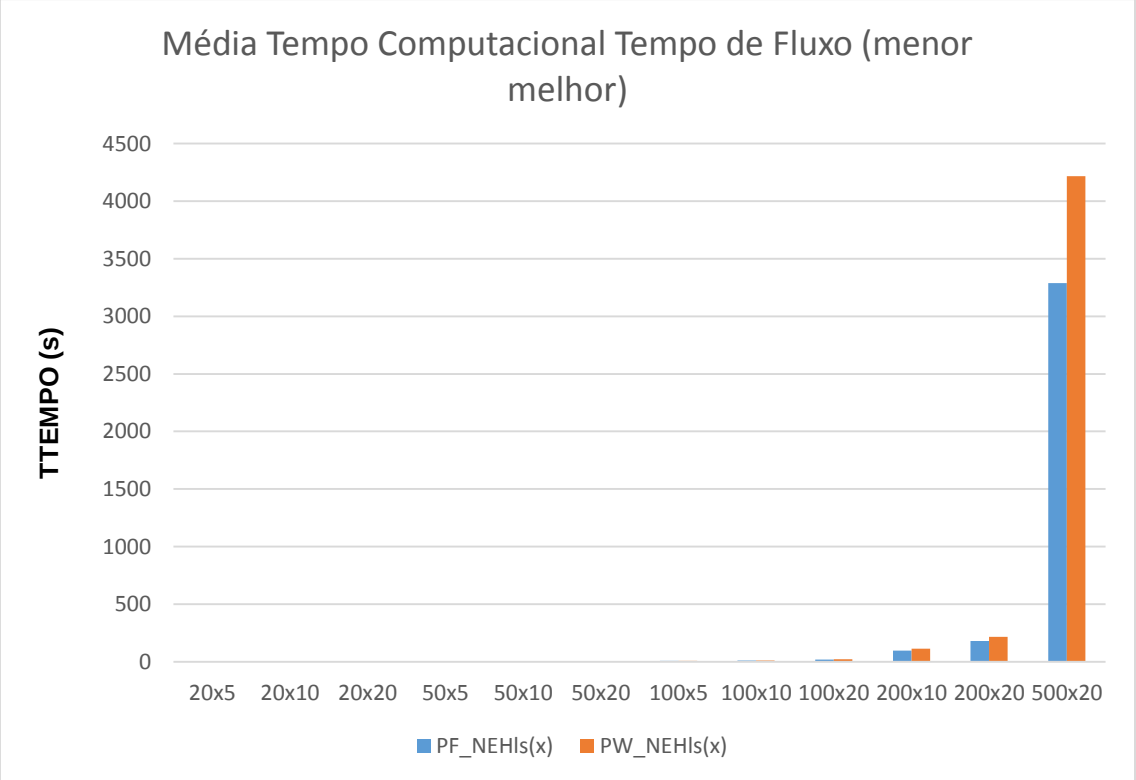

Gráfico 4- Média do tempo computacional para solucionar problemas com o objetivo de minimização do tempo de fluxo

Como mostram os gráficos, a heurística PW apresenta uma média de tempo computacional consideravelmente mais baixa para a classe de 500 problemas e 20 máquinas quando a minimização de *makespan* é avaliada. Já nas outras classes, as médias das heurísticas permanecem próximas.

Porém quando avalia-se o tempo de fluxo, os melhores tempos computacionais são da heurística PF, que apresentam tempos significativamente melhores para as três últimas classes de problemas (200x10, 200x20 e 500x20). Nas demais classes de problemas os resultados são semelhantes para as duas heurísticas.

DESVIO RELATIVO

Os próximos gráficos apresentam o desvio relativo médio dos resultados apresentados pelas heurísticas para a minimização do *makespan*, minimização do tempo de fluxo, e também dos tempos computacionais de cada um dos objetivos de cada uma das heurísticas.

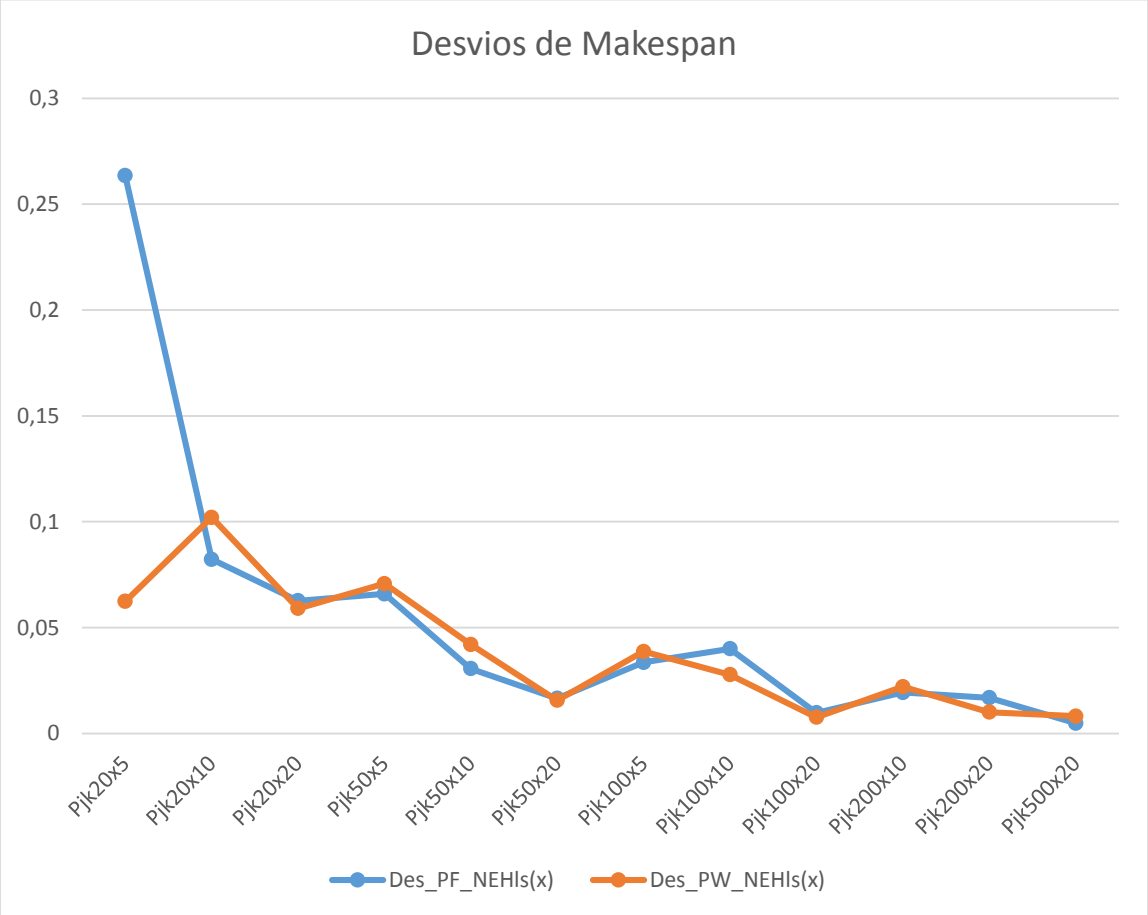

Gráfico 5 – Média do desvio relativo do *makespan*

Quando trata-se da média de desvio relativo do *makespan*, pode-se perceber no gráfico 5, mais claramente a eficiência de cada método para cada classe de problema. Para o objetivo minimização de *makespan* ambas heurísticas apresentaram bons resultados, para as classes 20x10, 50x5, 50x10, 100x5, 200x10 e 500x20 a heurística PF teve menores desvios, já para as demais, os melhores resultados foram da heurística PW.

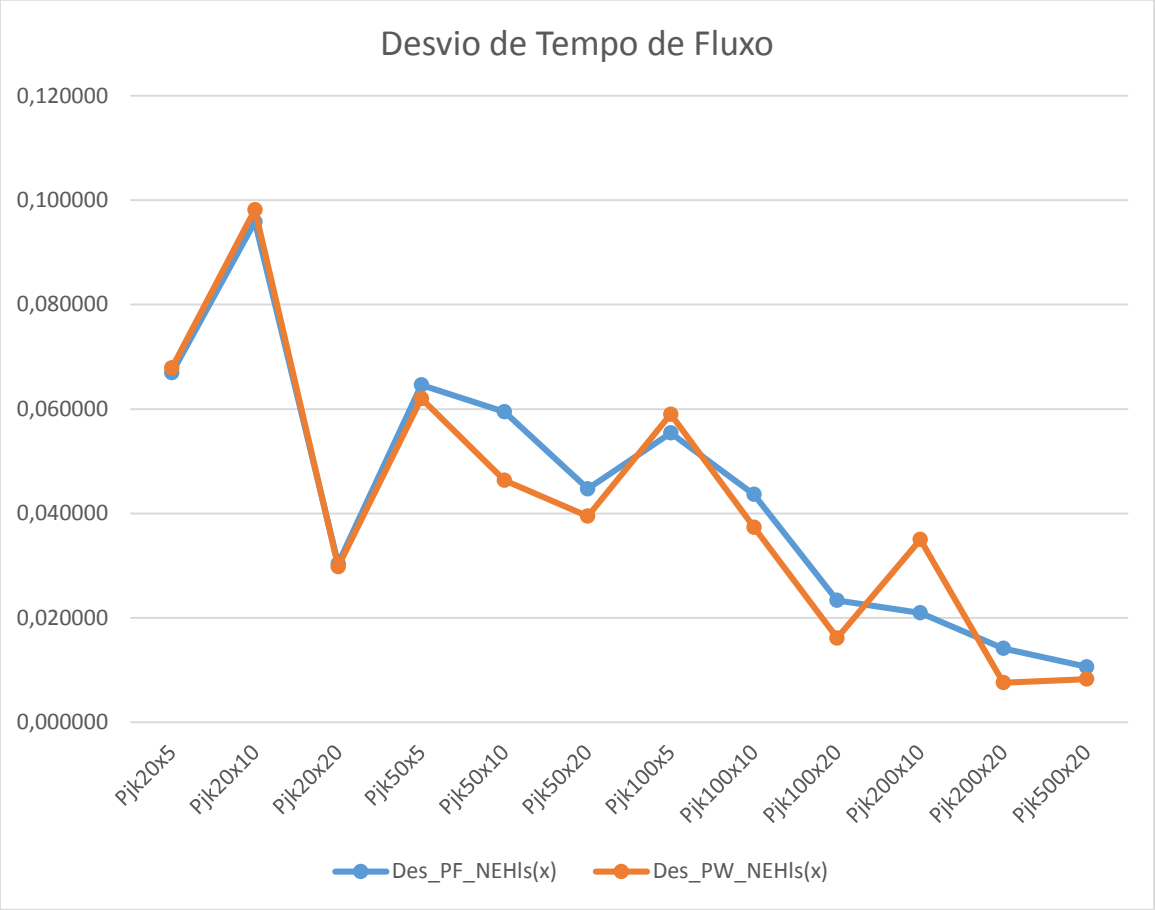

Gráfico 6 – Média do desvio relativo do tempo de fluxo

Para os desvios de tempo de fluxo, observando o gráfico 6 nota-se que a heurística PW foi a que obteve mais vezes o menor desvio relativo para os resultados. Esses resultados foram atingidos nas classes 20x20, 50x5, 50x10, 50x20, 100x10, 100x20, 200x20 e 500x20. E nas demais classes, a heurística PF teve os menores valores de desvio relativo.

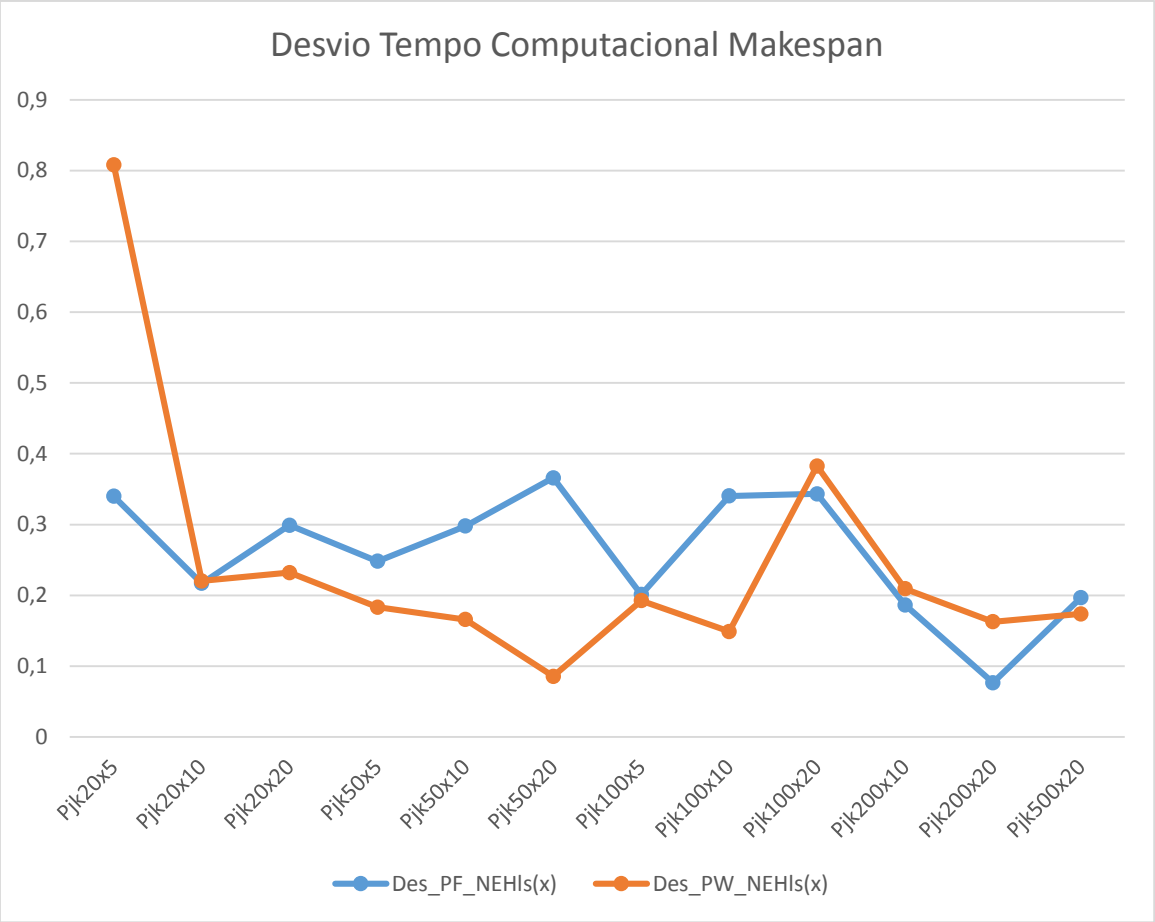

Gráfico 7- Médias do desvio relativo do tempo computacional do *makespan*

Para os desvios de tempo computacional, quando avaliou-se a eficiência das heurísticas para a minimização do *makespan*, a heurística PF foi eficiente em uma mesma quantidade de classes que a heurística PW. Como observa-se no gráfico 7.

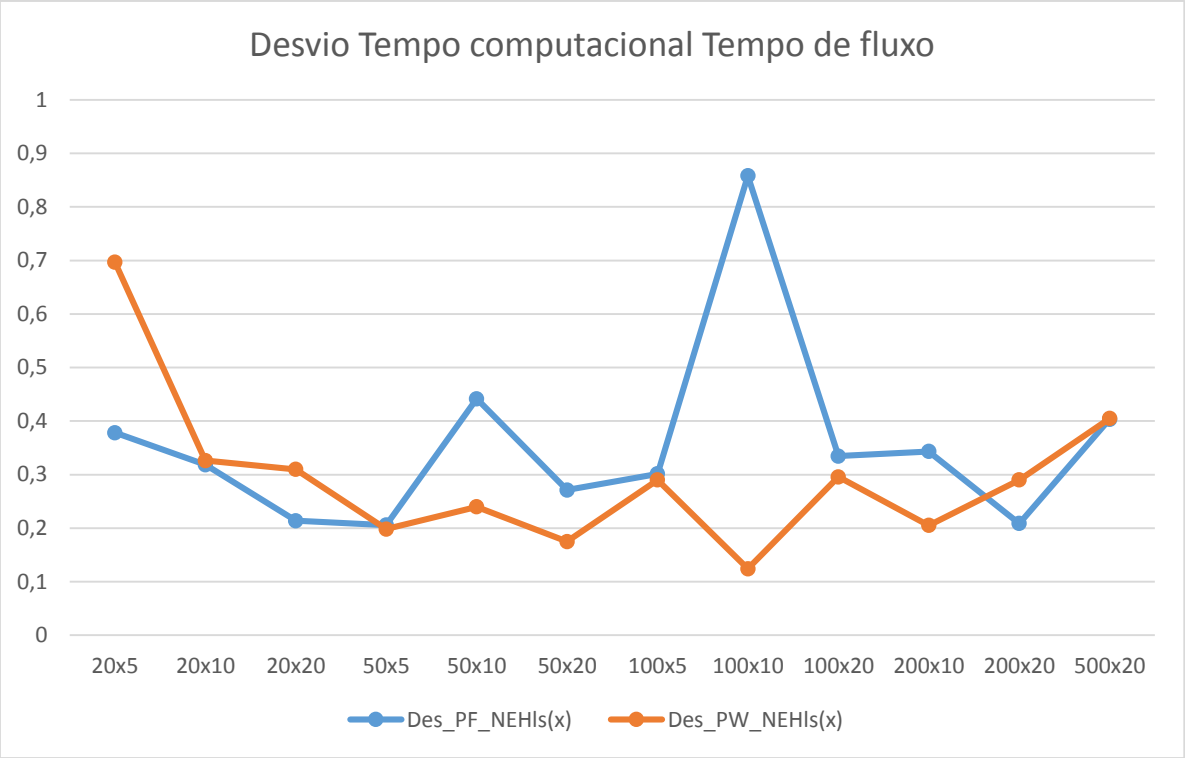

Gráfico 8- Médias do desvio relativo do tempo computacional do tempo de fluxo

Já quando analisou-se a eficiência das heurísticas para a minimização de tempo de fluxo, o tempo computacional da heurística PW, teve sucesso em um número maior de classes do que a heurística PF. Como mostra o gráfico 8.

A tabela abaixo apresenta a média dos desvios relativos médios.

MAKESPAN			TEMPO DE FLUXO	
	Des_PF_NEHls(x)	Des_PW_NEHls(x)	Des_PF_NEHls(x)	Des_PW_NEHls(x)
Pjk20x5	0,263427697	0,062308869	0,066947	0,067905
Pjk20x10	0,082234	0,102005	0,095860	0,098187
Pjk20x20	0,062660	0,059045	0,030360	0,029809
Pjk50x5	0,065905	0,070729	0,064608	0,062019
Pjk50x10	0,030559	0,042028	0,059469	0,046330
Pjk50x20	0,016534	0,015621	0,044730	0,039466
Pjk100x5	0,033515	0,038653	0,055431	0,059003
Pjk100x10	0,039961	0,027713	0,043659	0,037346
Pjk100x20	0,009690	0,007627	0,023348	0,016149
Pjk200x10	0,019337	0,022008	0,020944	0,035007
Pjk200x20	0,016805	0,010035	0,014145	0,007578
Pjk500x20	0,004784	0,008099	0,010627	0,008253
MÉDIA	0,053784302	0,038822755	0,044177	0,042254

TABELA 1- Média dos desvios relativos médios do makespan e tempo de fluxo

A partir da tabela 1, podemos perceber que quando a função objetivo foi minimizar o *makespan* a heurística PW_NEHls(x) foi mais eficiente, apresentando a média dos desvios relativos médios mais próximo de zero, o que significa que se aproximou mais da melhor solição. Quando a função objetivo foi minimizar o tempo de fluxo, apresentou-se uma diferença muito pequena entre as heurísticas, porém a heurística PW_NEHls(x), continuou apresentando o melhor resultado.

TEMPO COMPUTACIONAL MAKESPAN			TEMPO COMPUTACIONAL TEMPO DE FLUXO	
	Des_PF_NEHls(x)	Des_PW_NEHls(x)	Des_PF_NEHls(x)	Des_PW_NEHls(x)
Pjk20x5	0,339958444	0,808230671	0,378041628	0,696723769
Pjk20x10	0,216965	0,220446	0,31840608	0,326189527
Pjk20x20	0,298929	0,232023	0,213898396	0,309697245
Pjk50x5	0,248230	0,183233	0,205735375	0,198264027
Pjk50x10	0,297952	0,165870	0,441465353	0,239886476
Pjk50x20	0,365949	0,085478	0,271163055	0,174800133
Pjk100x5	0,201024	0,192498	0,301212736	0,289997644
Pjk100x10	0,340361	0,148800	0,857914977	0,124127927
Pjk100x20	0,343305	0,382785	0,334438622	0,295655842
Pjk200x10	0,186353	0,209219	0,343344444	0,204982211
Pjk200x20	0,076519	0,162665	0,208719852	0,289877704
Pjk500x20	0,196586	0,173770	0,403326581	0,404808614
MÉDIA	0,259344357	0,24708492	0,356472258	0,296250927

TABELA 2- Média dos desvios relativos médios do tempo computacional do makespan e tempo de fluxo

A tabela 2 apresenta as médias dos desvios relativos médios do tempo computacional para cada uma das funções objetivo. Para o primeiro objetivo, que é minimizar o makespan, a diferença do tempo computacional foi pequena, e o melhor resultado foi da heurística PW_NEHls(x). Quando analisou-se a minimização do tempo de fluxo, a heurística PW_NEHls(x) teve uma melhor eficiência, um sucesso no resultado mais considerável no tempo computacional.

A partir desses resultados, podemos concluir que a heurística PW_NEHls(x), é o método que atende com mais eficiência as funções objetivo estudas.

7. CONCLUSÃO

Este trabalho tem como função objetivo a minimização do *makespan* e a minimização do tempo de fluxo. Para isso, foram implementadas duas heurísticas de terceira fase para atuar nessa otimização. As analises foram feitas para um sistema flow shop permutacional com bloqueio e sem estoque intermediário.

Tais métodos foram submetidos a 120 problemas, com o objetivo de minimizar o *makespan*. Os mesmos métodos foram posteriormente adaptados com objetivo de minimizar também o tempo de fluxo. Avaliou-se os resultados a partir da média do desvio relativo de cada função objetivo, e da média dos tempos computacionais delas.

Os métodos implementados foram PF_NEHls(x) e PW_NEHls(x), para estes métodos pudemos perceber que cada um foi eficiente em uma determinada quantidade de classes de cada uma das funções objetivas.

Analisando a média dos desvios relativos médios pudemos perceber que a heurística PW_NEHls(x) foi mais eficiente para as funções objetivos do trabalho, teve melhores médias tanto para os objetivos como para o tempo computacional de cada um.

8. REFERÊNCIAS

BOIKO, T. J. P.; MORAIS, M. F.; VAROLO, F. W. R. **Programação da produção em sistemas de produção *flow shop* permutacional com restrições adicionais e critério de *flow time* médio.** Anais... Encontro de Produção Cientifica e Tecnológica, 5, Campo Mourão – PR, 2010.

CARAFFA, Vince; IANES, Stefano; BAGCHI, Tapan P.; SRISKANDARAJAH, Chelliah. **Minimizing makespan in a blocking flow shop using genetic algorithm**. International Journal of Prodiction Economics, v.70, p.101-115, 2001.

CASTRO, Lucas R. **Sequenciamento da produção em linhas flow shop Permutacional com bloqueio e sem estoque intermediário aplicando métodos heurísticos de II fase.** 76f. Trabalho de Conclusão de Curso (Graduação) – Curso Superior de Engenharia Industrial Mecânica, Universidade Tecnológica Federal do Paraná, Cornélio Procópio, 2014.

DEVANDRA, Donald; DEVANDRA, Magdalena Bialic. **Scheduling flow shops with blocking using a discrete self-organising migrating algorithm**. International Journal of Production Reasearch, p.1-19, 2012.

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 Operational Research Society, United Kingdom***.** v.12, n.55, p. 1243-1255, 2004.

GIGANTE, Rodrigo Luiz. **Heurística construtiva para a programação de operações *flow shop* permutacional.** São Carlos, 2010.

GRABOWSKI, Józef; PEMPERA, Jaroslaw. **The permutation flow shop problem with blocking. A tabu search approach**. International Journal of Management Science, v.35, p.302-311, 2007.

HAN&J, Yu-Yan; LIANG, J.J.; PAN, Quan-Ke; JUN-QUING, Li; SANG, Hong-Yan; CAO, N. N. **Effective hybrid discrete artificial bee colony algorithm for the total flowtime minimization in the blocking flowshop problem**. International Journal of Advenced Manufacturing Technology, 2012

HAN, Yu-Yan; DUAN, Jun-Hua; YANG, Yu-Jie; ZHANG, Min; YUN, Bao. **Minimizing the total flow time flowshop with blocking using a discrete artificial bee colony**. Lecture Notes in Compute Science, v.6839, p.91-97, 2012.

HAN, Yu-Yan; PAN, Quan-Ke; LI, Jun-Qing; SANG, Hong-yan. **An improved artificial bee colony algorithm for the blocking flow shop scheduling problem**. International Journal of Advenced Manufacturing Technology, v.60, p.1149-1159, 2012.

HILLIER, Frederick S.; LIEBERMAN, Gerald J., 1995. **Introduction to Operations Research.** Ed. McGraw-Hill, Estados Unidos, 998 p.

LIN, Shih-Wei; YING, Kuo-Ching. **Minimizing makespan in a blocking flow shop using a revised artificial immune system algorithm**. Journal Omega, p.383-384, 2013.

MOCCELLIN, J.V. **Técnicas de Seqüenciamento e Programação de operações em Máquinas**. 74p, Publicação Escola de Engenharia de São Carlos, Universidade de São Paulo, 2005.

PINEDO, Michael L. ***Scheduling*: *Theory, Algorithms, and Systems***. New York, 2008.

RIBAS, Imma; COMPANYS, Ramon; TORT-MARTORELL, Xavier. **An interated greedy algorithm for the flow shop scheduling problem with blocking**. Journal Omega, v.39, p.293-301, 2011.

RONCONI, Debora P. **A note on constructive heuristics for the flowshop problem with blocking**. Internacional Journal of Prodiction Economics, v.87, p39-48, 2004.

SANTOS, Fabio de S. **Sequenciamento da produção em linhas flow shop permutacional com bloqueio e com buffer zero**. 50 f. Trabalho de Conclusão de Curso (Graduação) - Curso Superior de Engenharia Mecânica, Universidade Tecnológica Federal do Paraná, Cornélio Procópio, 2013

SILVA, A. F.; MARINS, F. A. S.; SILVA, G. M.; LOPES, P. R. M. de A. **Desenvolvimento e otimização de modelos matemáticos por meio da linguagem GAMS.** Disponível em: < http://www.feg.unesp.br/~fmarins/GAMS/apostilagams.pdf>. Acesso em 7 jan.2014.

TAILLARD. E. **Benchmarks for basic scheduling problems**. European Journal of Operational Research, Amsterdam, v. 64, p.278-285, 1992.

TRABELSI, Wajdi; SAUVEY, Christophe; SAUER, Nathalie. **Heuristics and metaheuristics for mixed blocking constraints flow shop scheduling problems.** Journal Computers & Operations Research, v.39, p.2520-2527, 2012.

WANG, Cheng; SONG, Shiji; GUPTA, Jatinder N.D.; WU, Cheng. **A tree-phase algorithm for flow shop scheduling with blocking to minimize makespan**. Journal Computers & Operational Research, v.39, p.2880-2887, 2012.

WANG, Ling; PAN, Quan-Ke; SUGANTHAN, P.N.; WANG, Wen-Hong; WANG, Ya-Min. **A novel hybrid discret differential evolution algorithm for blocking flow shop scheduling problems**. Journal Computers & Operational, v.37, p.509-520, 2010.

WANG, Ling; PAN, Quan-Ke; TASGETIREN, M. Fatih. **A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem.** Journal Computers & Industrial Engineening, v.61, p.76-83, 2011.

WANG, Ling; PAN, Quan-Ke; TASGETIREN, M. Fatih. **Minimizing the total flow time in a flow shop if blocking by using hybrid harmony search algorithm.** Journal Expert Systems with Applications, v.37, p.7929-7936, 2010.

WANG, Quan-Ke Pan; WANG, Ling. **Effective heuristics for the blocking flowshop scheduling problem with makespan minimization**. Journal Omega, v.40, p.218-229, 2012.

WANG, Xianpeng; TANG, Lixin. **A discrete particle optimization algorithm with self-adaptive diversity control for permutation flowshop problem with blocking.** Journal Applied Soft Computing, v.12, p.652-662, 2012.

APÊNDICE A- Algoritmo *Makespan*

```
function [Makespan,D]=MakespanBlock(Ordem,n,m,Pjk)
%Programa para calcular o makespan de problema de bloqueio sem setup

D=zeros(n,m+1);
%Primeira tarefa na seqüência
D(1,1)=0;
for k=2:m+1
    D(1,k)=D(1,k-1)+Pjk(Ordem(1),k-1);
end

%Demais tarefas
for i=2:n
    D(i,1)=D(i-1,2);
    for k=2:m
        D(i,k)=D(i,k)+max(D(i,k-1)+Pjk(Ordem(i),k-1),D(i-1,k+1));
    end
    D(i,m+1)=D(i,m)+Pjk(Ordem(i),m);
end
Makespan=D(n,m+1);
```

APÊNDICE B- Algoritmo Tempo de Fluxo

```
function [Tempo_fluxo,D]=T_FluxoBlock(Ordem,n,m,Pjk)
%Programa para calcular o tempo de fluxo de problema de bloqueio sem setup

D=zeros(n,m+1);
%Primeira tarefa na seqüência
D(1,1)=0;
for k=2:m+1
    D(1,k)=D(1,k-1)+Pjk(Ordem(1),k-1);
end

%Demais tarefas
for i=2:n
    D(i,1)=D(i-1,2);
    for k=2:m
        D(i,k)=D(i,k)+max(D(i,k-1)+Pjk(Ordem(i),k-1),D(i-1,k+1));
    end
    D(i,m+1)=D(i,m)+Pjk(Ordem(i),m);
end
Tempo_fluxo=sum(D(:,m+1));
```

APÊNDICE C- Algoritmo RLS para *Makespan*

```
function [Ordem,Makespan,Djk]=RLS(pi,piref,n,m,Pjk)

melhoria=1;
%verificar enquanto houver melhora
while melhoria>0
    melhoria=0;
    [Makespan,~]=MakespanBlock(pi,n,m,Pjk);
    %criar uma matriz de zeros nxn pilinha
    for i=1:n
        pilinha=zeros(n,n);
        Makespanlinha=zeros(n,1);
        piaux=pi;
        piaux(piaux==piref(i))=[];
        %adicionar os valores da matriz
        for j=1:n
            if j>1
                pilinha(j,1:j-1)=piaux(1:j-1);
            end
            pilinha(j,j)=piref(i);
            if j<n
                pilinha(j,j+1:n)=piaux(j:end);
            end
            [Makespanlinha(j),~]=MakespanBlock(pilinha(j,:),n,m,Pjk);
        end
        %verificar se houve melhoria

        if min(Makespanlinha)<Makespan
            melhoria=1;
            Makespan=min(Makespanlinha);

pi=pilinha(find(Makespanlinha==min(Makespanlinha),1,'first'),:);
        end
    end
```

APÊNDICE D- Algoritmo RLS para Tempo de Fluxo

```
function [Ordem,TF,Djk]=RLS2(pi,piref,n,m,Pjk)

melhoria=1;
%verificar enquanto houver melhora
while melhoria>0
    melhoria=0;
    [TF,~]=T_FluxoBlock(pi,n,m,Pjk);
    %criar uma matriz de zeros nxn pilinha
    for i=1:n
        pilinha=zeros(n,n);
        TFlinha=zeros(n,1);
        piaux=pi;
        piaux(piaux==piref(i))=[];
        %adicionar os valores da matriz
        for j=1:n
            if j>1
                pilinha(j,1:j-1)=piaux(1:j-1);
            end
            pilinha(j,j)=piref(i);
            if j<n
                pilinha(j,j+1:n)=piaux(j:end);
            end
            [TFlinha(j),~]=T_FluxoBlock(pilinha(j,:),n,m,Pjk);
        end
        %verificar se houve melhoria

        if min(TFlinha)<TF
            melhoria=1;
            TF=min(TFlinha);
            pi=pilinha(find(TFlinha==min(TFlinha),1,'first'),:);
        end
    end
end
```

APÊNDICE E- Algoritmo PF_NEHls(x) para a função objetivo *Makespan*

```
function [Ordem,Makespan,Tempo_Computacional,D]=PF_NEHls(Pjk)
%Programa para heuristica construtiva PF-NEH para problema de
Sequenciamento
%em uma linha Flow Shop com bloqueio sem setup

tic
[n,m]=size(Pjk);
%Setando o valor de x (número de sequencias geradas)
x=5;
%Setando o valor de lambda (número de tarefas que passarão pela heurística
%NEH)
if n==20
    lambda=19;
else
    lambda=25;
end
%Iniciando a variável D (Departure time de cada tarefa)
Dt=zeros(n,m+1,x);
%Iniciando a sequencia Alpha, ordenada pela regra LPT
OrdemA=zeros(1,n);
SomaPjk=sum(Pjk,2);
LPT=sort(SomaPjk);
for pos=1:n
    OrdemA(pos)=find(SomaPjk==LPT(pos),1,'first');
    SomaPjk(OrdemA(pos))=NaN;
end

OrdemPi=zeros(x,n);
Makespans=zeros(1,x);
%Iniciando a heurística
for L=1:x
    %Primeira tarefa da sequencia Beta
    AL=OrdemA(L);
    %Iniciando a sequencia Beta, ordenada pela regra PF e setando a
    %primeira tarefa igual à AL
    [OrdemB]=PF5(Pjk,AL);

    %Iniciando a variável auxiliar
    clear OrdemBL;
    OrdemBL(1:n-lambda)=OrdemB(1:n-lambda);

    %Realizando a heurística NEH
    for k=n-lambda+1:n
        OrdemPiL=zeros(k,k);
        MakespanParc=zeros(1,k);
        for PosTeste=1:k
            if PosTeste>1
                OrdemPiL(PosTeste,1:PosTeste-1)=OrdemBL(1:PosTeste-1);
            end
            OrdemPiL(PosTeste,PosTeste)=OrdemB(k);
            if PosTeste<k
                OrdemPiL(PosTeste,PosTeste+1:end)=OrdemBL(PosTeste:end);
            end

[MakespanParc(PosTeste)]=MakespanBlock(OrdemPiL(PosTeste,:),k,m,Pjk);
        end
```

```
OrdemBL=OrdemPiL(find(MakespanParc==min(MakespanParc),1,'first'),:);
    end
       [OrdemPi(L,:),Makespans(L),Dt(:,:,L)]=RLS(OrdemBL,OrdemBL,n,m,Pjk);
end

Best=find(Makespans==min(Makespans),1,'first');
Makespan=Makespans(Best);
Ordem=OrdemPi(Best,:);
D=Dt(:,:,Best);
Tempo_Computacional=toc;
```

APÊNDICE F- Algoritmo PF_NEHls(x) para a função objetivo Tempo de Fluxo

```
function [Ordem,TF,Tempo_Computacional,D]=PF_NEHls2(Pjk)
%Programa para heuristica construtiva PF-NEH para problema de
Sequenciamento
%em uma linha Flow Shop com bloqueio sem setup

tic
[n,m]=size(Pjk);
%Setando o valor de x (número de sequencias geradas)
x=5;
%Setando o valor de lambda (número de tarefas que passarão pela heurística
%NEH)
if n==20
    lambda=19;
else
    lambda=25;
end
%Iniciando a variável D (Departure time de cada tarefa)
Dt=zeros(n,m+1,x);
%Iniciando a sequencia Alpha, ordenada pela regra LPT
OrdemA=zeros(1,n);
SomaPjk=sum(Pjk,2);
LPT=sort(SomaPjk);
for pos=1:n
    OrdemA(pos)=find(SomaPjk==LPT(pos),1,'first');
    SomaPjk(OrdemA(pos))=NaN;
end

OrdemPi=zeros(x,n);
TFs=zeros(1,x);
%Iniciando a heurística
for L=1:x
    %Primeira tarefa da sequencia Beta
    AL=OrdemA(L);
    %Iniciando a sequencia Beta, ordenada pela regra PF e setando a
    %primeira tarefa igual à AL
    [OrdemB]=PF5(Pjk,AL);

    %Iniciando a variável auxiliar
    clear OrdemBL;
    OrdemBL(1:n-lambda)=OrdemB(1:n-lambda);

    %Realizando a heurística NEH
    for k=n-lambda+1:n
        OrdemPiL=zeros(k,k);
        TFParc=zeros(1,k);
        for PosTeste=1:k
            if PosTeste>1
                OrdemPiL(PosTeste,1:PosTeste-1)=OrdemBL(1:PosTeste-1);
            end
            OrdemPiL(PosTeste,PosTeste)=OrdemB(k);
            if PosTeste<k
                OrdemPiL(PosTeste,PosTeste+1:end)=OrdemBL(PosTeste:end);
            end
            [TFParc(PosTeste)]=T_FluxoBlock(OrdemPiL(PosTeste,:),k,m,Pjk);
        end
```

```
         OrdemBL=OrdemPiL(find(TFParc==min(TFParc),1,'first'),:);
    end
       [OrdemPi(L,:),TFs(L),Dt(:,:,L)]=RLS2(OrdemBL,OrdemBL,n,m,Pjk);
end

Best=find(TFs==min(TFs),1,'first');
TF=TFs(Best);
Ordem=OrdemPi(Best,:);
D=Dt(:,:,Best);
Tempo_Computacional=toc;
```

APÊNDICE G- Algoritmo PW_NEHls(x) para a função objetivo *Makespan*

```
function [Ordem,Makespan,Tempo_Computacional,D]=PW_NEHls(Pjk)
%Programa para heuristica construtiva PF-NEH para problema de
Sequenciamento
%em uma linha Flow Shop com bloqueio sem setup

tic
[n,m]=size(Pjk);
%Setando o valor de x (número de sequencias geradas)
x=5;
%Setando o valor de lambda (número de tarefas que passarão pela heurística
%NEH)
if n==20
    lambda=19;
else
    lambda=25;
end
%Iniciando a variável D (Departure time de cada tarefa)
Dt=zeros(n,m+1,x);
%Iniciando a sequencia Alpha, ordenada pela regra LPT
OrdemA=zeros(1,n);
SomaPjk=sum(Pjk,2);
LPT=sort(SomaPjk);
for pos=1:n
    OrdemA(pos)=find(SomaPjk==LPT(pos),1,'first');
    SomaPjk(OrdemA(pos))=NaN;
end

OrdemPi=zeros(x,n);
Makespans=zeros(1,x);
%Iniciando a heurística
for L=1:x
    %Primeira tarefa da sequencia Beta
    AL=OrdemA(L);
    %Iniciando a sequencia Beta, ordenada pela regra PF e setando a
    %primeira tarefa igual à AL
    [OrdemB]=PW5(Pjk,AL);

    %Iniciando a variável auxiliar
    clear OrdemBL;
    OrdemBL(1:n-lambda)=OrdemB(1:n-lambda);

    %Realizando a heurística NEH
    for k=n-lambda+1:n
        OrdemPiL=zeros(k,k);
        MakespanParc=zeros(1,k);
        for PosTeste=1:k
            if PosTeste>1
                OrdemPiL(PosTeste,1:PosTeste-1)=OrdemBL(1:PosTeste-1);
            end
            OrdemPiL(PosTeste,PosTeste)=OrdemB(k);
            if PosTeste<k
                OrdemPiL(PosTeste,PosTeste+1:end)=OrdemBL(PosTeste:end);
            end

[MakespanParc(PosTeste)]=MakespanBlock(OrdemPiL(PosTeste,:),k,m,Pjk);
        end
```

```
OrdemBL=OrdemPiL(find(MakespanParc==min(MakespanParc),1,'first'),:);
    end
    [OrdemPi(L,:),Makespans(L),Dt(:,:,L)]=RLS(OrdemBL,OrdemBL,n,m,Pjk);
end

Best=find(Makespans==min(Makespans),1,'first');
Makespan=Makespans(Best);
Ordem=OrdemPi(Best,:);
D=Dt(:,:,Best);
Tempo_Computacional=toc;
```

APÊNDICE H- Algoritmo PW_NEHls(x) para a função objetivo Tempo de Fluxo

```
function [Ordem,TF,Tempo_Computacional,D]=PW_NEHls2(Pjk)
%Programa para heuristica construtiva PF-NEH para problema de
Sequenciamento
%em uma linha Flow Shop com bloqueio sem setup

tic
[n,m]=size(Pjk);
%Setando o valor de x (número de sequencias geradas)
x=5;
%Setando o valor de lambda (número de tarefas que passarão pela heurística
%NEH)
if n==20
    lambda=19;
else
    lambda=25;
end
%Iniciando a variável D (Departure time de cada tarefa)
Dt=zeros(n,m+1,x);
%Iniciando a sequencia Alpha, ordenada pela regra LPT
OrdemA=zeros(1,n);
SomaPjk=sum(Pjk,2);
LPT=sort(SomaPjk);
for pos=1:n
    OrdemA(pos)=find(SomaPjk==LPT(pos),1,'first');
    SomaPjk(OrdemA(pos))=NaN;
end

OrdemPi=zeros(x,n);
TFs=zeros(1,x);
%Iniciando a heurística
for L=1:x
    %Primeira tarefa da sequencia Beta
    AL=OrdemA(L);
    %Iniciando a sequencia Beta, ordenada pela regra PF e setando a
    %primeira tarefa igual à AL
    [OrdemB]=PW5(Pjk,AL);

    %Iniciando a variável auxiliar
    clear OrdemBL;
    OrdemBL(1:n-lambda)=OrdemB(1:n-lambda);

    %Realizando a heurística NEH
    for k=n-lambda+1:n
        OrdemPiL=zeros(k,k);
        TFParc=zeros(1,k);
        for PosTeste=1:k
            if PosTeste>1
                OrdemPiL(PosTeste,1:PosTeste-1)=OrdemBL(1:PosTeste-1);
            end
            OrdemPiL(PosTeste,PosTeste)=OrdemB(k);
            if PosTeste<k
                OrdemPiL(PosTeste,PosTeste+1:end)=OrdemBL(PosTeste:end);
            end
```

```
            [TFParc(PosTeste)]=T_FluxoBlock(OrdemPiL(PosTeste,:),k,m,Pjk);
        end
        OrdemBL=OrdemPiL(find(TFParc==min(TFParc),1,'first'),:);
    end
    [OrdemPi(L,:),TFs(L),Dt(:,:,L)]=RLS2(OrdemBL,OrdemBL,n,m,Pjk);
end

Best=find(TFs==min(TFs),1,'first');
TF=TFs(Best);
Ordem=OrdemPi(Best,:);
D=Dt(:,:,Best);
Tempo_Computacional=toc;
```

APÊNDICE I- Tabelas de resultados dos métodos para o *makespan* (menor melhor).

	CLASSE 20x5			
	PF_NEHls(x)		PW_NEHls(x)	
	Cnm	TC	Cnm	TC
Pjk1	1417	0,0955	1399	0,094745
Pjk2	1432	0,117027	1437	0,265266
Pjk3	1307	0,131231	1321	0,130799
Pjk4	1449	0,108142	1463	0,205832
Pjk5	1351	0,163546	1351	0,16483
Pjk6	1389	0,149885	1390	0,164942
Pjk7	1412	0,11107	1398	0,156266
Pjk8	1413	0,121239	1415	0,137653
Pjk9	1409	0,148244	1413	0,138027
Pjk10	1320	0,133781	1308	0,1601

	CLASSE 20x10			
	PF_NEHls(x)		PW_NEHls(x)	
	Cnm	TC	Cnm	TC
Pjk11	1740	0,198157	1731	0,157274
Pjk12	1843	0,166318	1839	0,186444
Pjk13	1705	0,145478	1701	0,152252
Pjk14	1576	0,172186	1546	0,226331
Pjk15	1617	0,168153	1617	0,203232
Pjk16	1624	0,14059	1630	0,176174
Pjk17	1632	0,185732	1633	0,165843
Pjk18	1753	0,138773	1754	0,193568
Pjk19	1767	0,133149	1777	0,20284
Pjk20	1799	0,171837	1809	0,194194

	CLASSE 20x20			
	PF_NEHls(x)		PW_NEHls(x)	
	Cnm	TC	Cnm	TC
Pjk21	2465	0,249059	2455	0,272534
Pjk22	2263	0,1858	2261	0,215909
Pjk23	2523	0,229799	2504	0,236519
Pjk24	2374	0,191659	2373	0,221099
Pjk25	2484	0,158032	2447	0,264969
Pjk26	2405	0,232664	2412	0,200735
Pjk27	2427	0,188516	2411	0,249803
Pjk28	2353	0,174225	2349	0,288362
Pjk29	2399	0,19206	2387	0,242822
Pjk30	2355	0,25091	2346	0,280352

	CLASSE 50x5			
	PF_NEHls(x)		PW_NEHls(x)	
	Cnm	TC	Cnm	TC
Pjk31	3118	0,982806	3104	1,128175
Pjk32	3284	1,042269	3310	0,929097
Pjk33	3106	0,929833	3129	0,977723
Pjk34	3257	1,148293	3187	1,009016
Pjk35	3267	1,098454	3281	0,898135
Pjk36	3273	1,010473	3260	0,964884
Pjk37	3097	1,152946	3135	1,183374
Pjk38	3144	0,965205	3152	1,125791
Pjk39	2977	0,832386	2962	1,281204
Pjk40	3209	1,227425	3195	1,129632

	CLASSE 50x10			
	PF_NEHls(x)		PW_NEHls(x)	
	Cnm	TC	Cnm	TC
Pjk41	3726	1,767366	3774	1,495528
Pjk42	3629	1,557345	3612	1,78981
Pjk43	3593	1,612679	3550	1,779894
Pjk44	3776	1,9146	3779	1,496688
Pjk45	3751	1,698263	3735	1,56944
Pjk46	3706	1,329383	3716	1,794663
Pjk47	3827	1,696613	3817	1,786144
Pjk48	3651	1,783722	3688	1,42698
Pjk49	3640	2,068815	3622	1,859052
Pjk50	3729	1,825968	3699	1,638534

	CLASSE 50x20			
	PF_NEHls(x)		PW_NEHls(x)	
	Cnm	TC	Cnm	TC
Pjk51	4584	2,628519	4617	3,007725
Pjk52	4404	2,852628	4407	2,888383
Pjk53	4385	2,734375	4385	3,216186
Pjk54	4498	2,614015	4468	3,229408
Pjk55	4421	2,358748	4387	2,874655
Pjk56	4439	2,486731	4427	2,752466
Pjk57	4451	1,962269	4390	2,991122
Pjk58	4428	3,435765	4432	2,895029
Pjk59	4405	2,982362	4449	3,121431
Pjk60	4560	2,748186	4573	2,901009

	CLASSE 100x5			
	PF_NEHls(x)		PW_NEHls(x)	
	Cnm	TC	Cnm	TC
Pjk61	6197	6,511008	6209	6,221947
Pjk62	6115	5,895635	6119	6,222539
Pjk63	6016	5,874165	5998	7,245999
Pjk64	5863	6,183776	5852	5,493071
Pjk65	6022	5,812573	6034	5,582936
Pjk66	5909	6,531955	5916	6,87956
Pjk67	6115	7,176176	6120	6,533637
Pjk68	5949	7,185707	6047	6,891362
Pjk69	6191	9,42454	6241	5,257597
Pjk70	6218	9,214847	6246	6,368101

	CLASSE 100x10			
	PF_NEHls(x)		PW_NEHls(x)	
	Cnm	TC	Cnm	TC
Pjk71	7106	13,39106	7140	11,57232
Pjk72	6875	14,41626	6848	11,02334
Pjk73	7041	16,92316	7035	11,95872
Pjk74	7227	15,77007	7287	11,0507
Pjk75	6904	14,70954	6925	12,49177
Pjk76	6719	18,87923	6827	10,54894
Pjk77	6946	16,681	6920	13,30731
Pjk78	6936	13,54353	6938	12,49394
Pjk79	7102	17,6945	7167	13,08352
Pjk80	7019	11,44895	7075	13,65565

	CLASSE 100x20			
	PF_NEHls(x)		PW_NEHls(x)	
	Cnm	TC	Cnm	TC
Pjk81	8007	25,96229	7989	21,88753
Pjk82	8051	22,15998	7996	27,93439
Pjk83	7989	23,29749	7986	23,59599
Pjk84	8017	28,15787	7985	19,88782
Pjk85	7988	29,66828	8059	17,38017
Pjk86	8061	33,91166	8096	15,61844
Pjk87	8162	30,9583	8110	18,31023
Pjk88	8144	32,8464	8099	19,93493
Pjk89	8061	20,54214	8037	23,662
Pjk90	8174	28,43933	8102	27,75792

	CLASSE 200x10			
	PF_NEHls(x)		PW_NEHls(x)	
	Cnm	TC	Cnm	TC
Pjk91	13458	75,95535	13508	78,9859
Pjk92	13344	85,02556	13468	77,93923
Pjk93	13461	69,10077	13472	78,64801
Pjk94	13410	95,45832	13472	103,7731
Pjk95	13416	90,17837	13446	86,09925
Pjk96	13182	86,2918	13195	70,69054
Pjk97	13658	74,27004	13764	93,10872
Pjk98	13592	88,84314	13651	74,39944
Pjk99	13397	77,02627	13407	104,6628
Pjk100	13451	77,62915	13471	86,49684

	CLASSE 200x20			
	PF_NEHls(x)		PW_NEHls(x)	
	Cnm	TC	Cnm	TC
Pjk101	14823	168,4945	14928	171,0545
Pjk102	15060	168,3869	15081	200,0215
Pjk103	15158	160,6827	15242	178,9505
Pjk104	15201	195,7385	15118	199,4004
Pjk105	14905	161,144	14959	186,303
Pjk106	15102	168	15118	166,2316
Pjk107	15126	175,2112	15097	206,3625
Pjk108	15183	181,6758	15118	193,1094
Pjk109	15056	174,8614	15059	178,3485
Pjk110	15107	175,5859	15058	158,0726

	CLASSE 500x20			
	PF_NEHls(x)		PW_NEHls(x)	
	Cnm	TC	Cnm	TC
Pjk111	35933	2975,046	35853	2716,211
Pjk112	36061	2983,868	36306	3155,098
Pjk113	35881	3491,111	35784	3283,721
Pjk114	36142	3713,419	36156	2742,204
Pjk115	35871	2885,214	36054	3156,695
Pjk116	36034	3533,506	36415	3411,577
Pjk117	35787	3706,167	36015	3045,889
Pjk118	35921	2775,275	36134	3574,693
Pjk119	35801	3696,898	35883	3509,578
Pjk120	36151	3448,053	36138	3286,406

APÊNDICE J- Tabelas de resultados dos métodos adaptados para o tempo de fluxo (menor melhor).

	CLASSE 20x5			
	PF_NEHls(x)		PW_NEHls(x)	
	Mf	TC	Mf	TC
Pjk1	15059	0,124142	15059	0,158218
Pjk2	16551	0,143044	16525	0,118071
Pjk3	14466	0,153122	14449	0,154475
Pjk4	16700	0,174317	16672	0,204048
Pjk5	14292	0,145381	14292	0,128359
Pjk6	14741	0,128293	14847	0,161846
Pjk7	15073	0,095472	15115	0,127819
Pjk8	15364	0,105364	15377	0,129249
Pjk9	15800	0,129614	15708	0,083314
Pjk10	14442	0,116899	14581	0,148214

	CLASSE 20x10			
	PF_NEHls(x)		PW_NEHls(x)	
	Mf	TC	Mf	TC
Pjk11	22537	0,147759	22566	0,203717
Pjk12	23969	0,20386	24137	0,143276
Pjk13	20958	0,165227	20988	0,163637
Pjk14	20110	0,133259	20084	0,160056
Pjk15	20551	0,164293	20387	0,16062
Pjk16	20239	0,196523	20239	0,151489
Pjk17	19495	0,16945	19471	0,174308
Pjk18	21447	0,119964	21459	0,167562
Pjk19	21697	0,130244	21588	0,123026
Pjk20	22635	0,15103	22909	0,183871

	CLASSE 20x20			
	PF_NEHls(x)		PW_NEHls(x)	
	Mf	TC	Mf	TC
Pjk21	34744	0,23667	34978	0,275017
Pjk22	33152	0,179715	33198	0,204252
Pjk23	35199	0,182837	35183	0,243627
Pjk24	33210	0,213356	33168	0,297193
Pjk25	35374	0,218259	35390	0,215073
Pjk26	33720	0,221623	33720	0,221612
Pjk27	34275	0,213998	34200	0,177114
Pjk28	33597	0,188583	33506	0,203461
Pjk29	35007	0,169596	35018	0,26379
Pjk30	33307	0,234086	33206	0,218514

	CLASSE 50x5			
	PF_NEHls(x)		PW_NEHls(x)	
	Mf	TC	Mf	TC
Pjk31	74960	1,009804	76292	1,282842
Pjk32	81581	0,924328	80389	1,381435
Pjk33	76607	0,92645	76390	1,151631
Pjk34	79297	0,930766	79916	1,090369
Pjk35	81886	1,071684	82919	1,080387
Pjk36	77476	1,333369	78903	1,433912
Pjk37	75988	1,114941	77568	1,402643
Pjk38	75880	1,284828	75977	1,333191
Pjk39	73130	1,039832	73734	1,060315
Pjk40	81743	0,871456	80981	1,488643

	CLASSE 50x10			
	PF_NEHls(x)		PW_NEHls(x)	
	Mf	TC	Mf	TC
Pjk41	102022	1,45861	102835	1,501699
Pjk42	98154	1,400625	99181	1,410106
Pjk43	93974	1,08007	95915	1,745177
Pjk44	100659	1,969218	101514	1,608881
Pjk45	100315	1,237942	100142	2,031435
Pjk46	98951	1,547131	100115	1,489119
Pjk47	102531	1,956758	103081	2,150488
Pjk48	99386	1,729127	100197	1,362875
Pjk49	99894	1,542132	99236	1,754385
Pjk50	99739	1,64722	101371	1,843942

	CLASSE 50x20			
	PF_NEHls(x)		PW_NEHls(x)	
	Mf	TC	Mf	TC
Pjk51	139426	2,487054	139392	2,261265
Pjk52	131859	2,875953	132484	3,269683
Pjk53	129036	2,254357	129809	2,762899
Pjk54	134724	2,865299	133689	2,383634
Pjk55	133709	2,85442	133519	2,405756
Pjk56	134334	3,179843	134349	2,28322
Pjk57	137296	2,127896	137076	3,401255
Pjk58	134557	3,326801	135935	2,39253
Pjk59	134798	2,601666	134661	2,247322
Pjk60	138339	2,475736	138407	2,993981

	CLASSE 100x5			
	PF_NEHls(x)		PW_NEHls(x)	
	Mf	TC	Mf	TC
Pjk61	298123	7,600021	301003	6,67736
Pjk62	294956	6,520775	297472	8,44504
Pjk63	289015	11,04724	291625	9,767196
Pjk64	274981	7,237464	274929	7,059313
Pjk65	288944	7,616897	289815	8,015863
Pjk66	289319	7,898238	283793	10,14005
Pjk67	287340	7,215618	289243	8,057123
Pjk68	285658	7,34738	288306	11,86636
Pjk69	297154	7,612037	301210	8,762353
Pjk70	296744	5,835409	294109	7,347138

	CLASSE 100x10			
	PF_NEHls(x)		PW_NEHls(x)	
	Mf	TC	Mf	TC
Pjk71	359919	8,9161	367453	10,52922
Pjk72	345412	8,925823	346271	12,01018
Pjk73	354993	10,43345	355320	11,07992
Pjk74	368163	7,424438	372955	10,54423
Pjk75	347416	9,965102	348478	11,03897
Pjk76	338652	10,43732	346562	11,52967
Pjk77	344632	10,04738	348495	10,0198
Pjk78	350102	9,874826	362170	12,15845
Pjk79	365050	4,861793	374629	12,64884
Pjk80	360034	9,441756	369696	11,07605

	CLASSE 100x20			
	PF_NEHls(x)		PW_NEHls(x)	
	Mf	TC	Mf	TC
Pjk81	432910	17,80911	436442	20,63038
Pjk82	444589	21,85513	444543	22,41601
Pjk83	440483	14,70228	440226	26,13048
Pjk84	445010	22,68733	441030	24,98693
Pjk85	436331	18,32963	438262	25,16489
Pjk86	439860	17,31088	443947	22,43386
Pjk87	447633	20,95741	445742	17,79923
Pjk88	450696	23,64492	450084	26,03618
Pjk89	444513	14,58543	444413	18,49533
Pjk90	448150	22,75147	450213	17,06957

	CLASSE 200x10			
	PF_NEHls(x)		PW_NEHls(x)	
	Mf	TC	Mf	TC
Pjk91	1310117	83,22542	1327805	100,0917
Pjk92	1296847	71,41056	1354699	126,3935
Pjk93	1311758	98,52047	1350286	93,07683
Pjk94	1308838	91,33621	1321672	103,9151
Pjk95	1308131	113,6661	1334633	100,0081
Pjk96	1284064	144,5739	1295107	123,4744
Pjk97	1344042	102,9751	1383781	122,8621
Pjk98	1328758	90,97021	1355873	110,7554
Pjk99	1303928	71,49161	1337219	126,3563
Pjk100	1313097	91,1202	1343375	114,6257

	CLASSE 200x20			
	PF_NEHls(x)		PW_NEHls(x)	
	Mf	TC	Mf	TC
Pjk101	1529689	185,0487	1560606	204,7709
Pjk102	1561103	148,3173	1583806	196,0246
Pjk103	1571652	148,4764	1586436	167,8535
Pjk104	1560719	192,356	1579647	238,594
Pjk105	1532716	207,7374	1562593	269,2108
Pjk106	1545078	171,5522	1571756	239,7525
Pjk107	1554544	192,119	1567778	211,1543
Pjk108	1556175	206,5017	1580505	216,4253
Pjk109	1540164	169,9245	1566191	246,7624
Pjk110	1561426	170,7072	1564997	174,5563

	CLASSE 500x20			
	PF_NEHls(x)		PW_NEHls(x)	
	Mf	TC	Mf	TC
Pjk111	8751207	3237,893	9016387	3001,961
Pjk112	8863777	2920,477	9141691	3864,991
Pjk113	8808876	2344,102	9067546	3988,482
Pjk114	8875459	3948,026	9040738	4047,143
Pjk115	8846803	2805,566	9075426	5612,749
Pjk116	8886303	3210,743	9151225	3644,867
Pjk117	8812681	2877,522	8987574	4320,357
Pjk118	8858905	2915,71	9083773	4493,496
Pjk119	8856031	3495,393	9010792	4955,569
Pjk120	8882039	5139,977	9042354	4242,19