UNIVERSIDADE FEDERAL DE GOIÁS
REGIONAL CATALÃO
MESTRADO EM GESTÃO ORGANIZACIONAL

MÉTODOS HEURÍSTICOS PARA PROGRAMAÇÃO DE *FLOW SHOP* BICRITÉRIO COM DATAS DE LIBERAÇÃO E *SETUP* INDEPENDENTE

Caio Soares de Araújo

CATALÃO-GO
2014

Caio Soares de Araújo

MÉTODOS HEURÍSTICOS PARA PROGRAMAÇÃO DE *FLOW SHOP* BICRITÉRIO COM DATAS DE LIBERAÇÃO E *SETUP* INDEPENDENTE

Dissertação apresentada ao Programa de Mestrado Profissional em Gestão Organizacional da Universidade Federal de Goiás, Regional Catalão, como requisito parcial para obtenção do título de Mestre em Gestão Organizacional.

Orientador: Prof. Dr. Hélio Yochihiro Fuchigami

CATALÃO-GO
2014

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Neste estudo são apresentadas e avaliadas regras de sequenciamento e métodos heurísticos para programação da produção em sistemas *flow shop* permutacional com tempos de *setup* independentes da sequência de execução das tarefas. Foram consideradas também as datas de liberação das tarefas, para simular uma realidade mais próxima de um ambiente de produção numa organização em que as tarefas geralmente chegam de forma dinâmica e não simultaneamente. Na primeira parte deste trabalho, foram definidas e implementadas computacionalmente oito regras (R_1, R_2, R_3, R_4, R_5, R_6, R_7 e R_8), fundamentadas nas reconhecidas regras SPT (*Shortest Processing Time*) e LPT (*Longest Processing Time*), adaptadas a este problema. A segunda etapa deste estudo constituiu-se da criação de quatro heurísticas (H_1, H_2, H_3 e H_4), baseadas nas melhores regras, e análise comparativa do desempenho de todos os métodos de solução propostos, nos moldes da avaliação feita para as Regras de Prioridade, incluindo os resultados da solução ótima obtida por meio do método de enumeração completa. Após a implementação computacional, as heurísticas H_1, H_2 e H_3 apresentaram um desempenho eficiente, com resultados satisfatórios próximos a solução ótima.

Palavras-chave: Programação da produção; *Flow Shop* Permutacional; *Setup* independente.

ABSTRACT

In this work are presented and evaluated sequencing rules and heuristic methods for production scheduling in permutation flow shop systems with setup times independent of the sequence of execution of tasks. The release dates of the tasks were also considered, seeking simulate a reality closer to a production environment in an organization where tasks usually arrive dynamically and not simultaneously. In the first part of this study, were defined and implemented computationally eight rules (R_1, R_2, R_3, R_4, R_5, R_6, R_7 and R_8), based on recognized rules SPT (Shortest Processing Time) and LPT (Longest Processing Time), adapted to this problem. The second stage of this study consisted of the creation of four heuristics (H_1, H_2, H_3 and H_4), based on the best rules, and comparative analysis of the performance of all the methods proposed solution, similar to the assessment rules priority, including the results of the optimal solution obtained by the method of complete enumeration. After computational implementation, the H_1, H_2 and H_3 heuristics performed well, with satisfactory results near optimal solution.

KEYWORDS: Production scheduling; Permutation Flow Shop; Setup independent.

1. INTRODUÇÃO

A Administração da Produção desempenha um papel fundamental para que sejam alcançadas as metas e os objetivos estratégicos de uma empresa. A grande concorrência, em todos os níveis, tem levado as empresas a utilizarem técnicas de otimização dos seus processos cada vez mais eficientes, para gerir da melhor forma as atividades de produção. Diante dessa crescente concorrência e competitividade da maioria das organizações de diferentes setores da economia, faz-se necessária a busca pelo melhor desempenho dos seus recursos.

A gestão da produção, entre outros, consiste no atendimento das necessidades de seus consumidores, que primam por produtos e serviços com qualidade, rapidez, confiabilidade, flexibilidade e custo adequado, além de dar suporte à empresa na obtenção de vantagem competitiva, que pode ser obtida por meio do gerenciamento dos recursos de todo o processo produtivo, para favorecer a organização com um conjunto de características de desempenho adequado as suas necessidades estratégicas.

Um eficiente processo produtivo requer um gerenciamento dinâmico por meio do Planejamento e Controle da Produção (PCP). O PCP é responsável pelo planejamento, execução e controle das atividades realizadas no sistema de produção, ou seja, desenvolver os planos que irão guiar a produção, além de realizar o controle. Deste modo, Zaccarelli (1987) salienta que o PCP seria a atividade de determinar “o que” vai ser produzido, “em qual quantidade” vai ser produzido, “como” e “onde” vai ser produzido, “quem” vai produzir e “quando” vai ser produzido.

As atividades do PCP são exercidas em três níveis hierárquicos: estratégico, tático e operacional. No nível estratégico, o PCP corrobora na formação do Planejamento Estratégico, para gerar um plano de produção. No nível tático, o PCP desenvolve o Planejamento-mestre da Produção, obtendo o Plano-mestre de Produção (PMP). No nível operacional, o PCP organiza a Programação da Produção, faz a gestão de estoques, sequencia, além de executar o acompanhamento e controle da produção (TUBINO, 2007).

Dentro das diversas funções do PCP encontra-se a atividade de programação da produção. Nada mais é do que a determinação das alocações e quantidades das operações para que seja fabricado um produto. É uma complexa atividade na administração dos sistemas de produção por envolver vários tipos de tarefas e recursos ao mesmo tempo,

distribui essas tarefas nas máquinas e determina o começo e o fim de cada uma delas (MOCCELLIN, 2005).

De acordo com Baker (1974), o problema de programação da produção em geral, requer decisões de sequenciamento e alocação de recursos. Tais recursos podem ser ferramentas, materiais, máquinas, etc. O sequenciamento da produção (*scheduling*) é um problema de priorização de atividades. Pinedo (2012) diz que problemas de agendamento incluem a alocação de recursos limitados, ao longo do tempo, para realizar tarefas, de modo que satisfaçam os objetivos estabelecidos.

Diante desse contexto, incorpora-se o ambiente *flow shop*. O *flow shop* trata-se de uma linha de produção com fluxo numa única direção. Segundo Fuchigami (2014), cada estágio de produção é composto por uma única máquina e as tarefas possuem um fluxo linear de processamento que passa por várias máquinas. O problema de sequenciamento no *flow shop* clássico consiste em um conjunto de n tarefas (J_1, J_2, ..., J_j, ..., J_n), que tem que ser sequenciado em m máquinas (M_1, M_2, ..., M_k, ... , M_m) , dispostas em série. Cada tarefa é composta por m operações, cuja primeira é executada na máquina M_1, a segunda na M_2 e assim por diante (FARBER e COVES, 2005).

A seguir, a Figura 1 ilustra um ambiente de *flow shop* num processo produtivo.

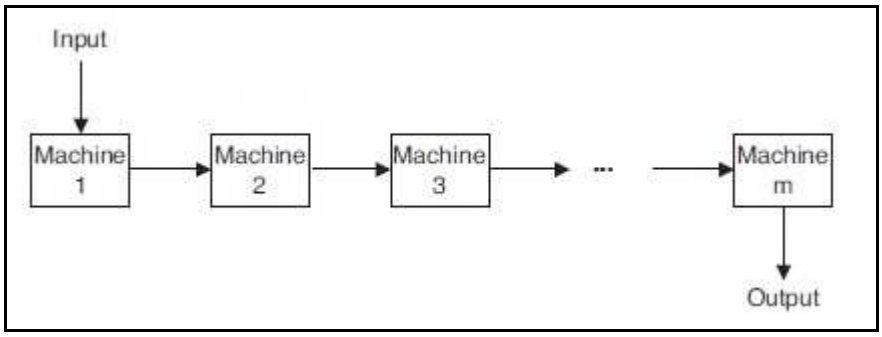

Figura 1 – Fluxo de trabalho num *flow shop* simples – Fonte: Baker e Trietsch (2009).

Por este trabalho se tratar de um *flow shop* permutacional, todas as tarefas J_j têm que passar por todas as máquinas na mesma ordem, ou seja, a ordem de processamento das tarefas em cada máquina é a mesma.

Diante desse contexto, é necessário ressaltar alguns pressupostos importantes do problema: a) cada máquina pode processar apenas uma tarefa de cada vez; b) as operações nas diversas máquinas, uma vez iniciadas não devem ser interrompidas; c) os tempos de

processamento das tarefas são fixos e determinados; d) as tarefas têm diferentes datas de liberação; e) os tempos de *setup* das operações nas diversas máquinas podem ser antecipados, ou seja, iniciar antes da liberação da operação, e independentes da sequência de operações em cada máquina; f) as máquinas estão sempre disponíveis para processamento, ou seja, não foram consideradas paradas por quebras; g) não há outros recursos limitantes.

Quanto à chegada das tarefas, Haupt (1989) afirma que podem ocorrer de duas formas: de forma estática, quando as tarefas chegam simultaneamente e são sequenciadas no mesmo instante (não há data de liberação) ou de forma dinâmica, em que são permitidas chegadas contínuas, mas não ao mesmo tempo.

Portanto, neste estudo serão propostos e testados vários métodos de solução heurística para programação da produção em sistemas *flow shop* permutacional com o objetivo de minimizar conjuntamente o *makespan* (duração total da programação) e o tempo médio de fluxo das tarefas.

Após estas exposições, surge o problema de pesquisa que orientará os caminhos deste estudo: **como programar de forma eficaz e eficiente o problema de *flow shop* permutacional bicritério com datas de liberação e *setup* independente?**

1.1 Objetivos

Neste trabalho, o objetivo principal é criar e avaliar o desempenho de Heurísticas Construtivas para o problema tratado, conforme já definido anteriormente, que forneçam a programação do ambiente produtivo em tempo computacional viável.

Como objetivos específicos, esta pesquisa visa organizar a literatura específica da área em relação aos trabalhos que abordam problemas relacionados, estudar as características estruturais do problema tratado, analisar a influência dos métodos de solução para diferentes pesos das medidas de desempenho consideradas (*makespan* e tempo médio de fluxo) e verificar o desempenho das heurísticas para classes de problemas-testes com diferente tamanhos (portes dos problemas dados pelo número de tarefas e de máquinas, além de diferentes intervalos de tempos de *setup* e datas de liberação).

1.2 Metodologia de Pesquisa

Segundo as definições de Martins (2010) e Nakano (2010), esta pesquisa possui abordagem quantitativa, pois preocupa-se com mensurabilidade, causalidade, generalização e replicação. Pode também ser classificada como experimento, uma vez que testa o relacionamento entre as variáveis da pesquisa operacionalizada, manipulando variáveis independentes (neste caso, a programação do problema) para se observar o resultado na variável dependente (representada aqui pelas medidas de desempenho *makespan* e *flowtime*).

Além disso, de acordo com Jung (2004), este trabalho se classifica como pesquisa aplicada quanto à natureza, por gerar conhecimento com finalidades de aplicação prática, pesquisa exploratória quanto aos objetivos, pois visa melhoria teórico-prática de sistemas, processos e produtos, e inovação pela proposição de novos modelos, além de ser feita a partir de impulsos criativos, simulações e experimentações, podendo originar novos modelos destinados a invenções, inovações e a otimização, e pesquisa experimental quanto aos procedimentos, para a obtenção de novos conhecimentos e produtos tecnológicos, requerendo uma manipulação de variáveis detalhada e sistemática, e originando inovações a partir de ensaios e estudos dinâmicos em laboratório.

2. REFERENCIAL TEÓRICO

Para que fosse desenvolvido este trabalho, foram buscadas referências consolidadas, para fazer com que o texto tenha um amparo sólido do que está sendo proposto, além de contribuir para o entendimento do problema. Para tanto, esta seção foi subdivida em três partes.

A primeira parte versa sobre os parâmetros dos problemas e os indicadores de desempenho usados para saber qual método de solução é mais vantajoso dependendo do objetivo da programação. Na segunda parte, verifica-se a explanação sobre a notação, que visa auxiliar na classificação de um problema de sequenciamento. E a terceira e última parte aborda uma análise histórica dos artigos publicados sobre *flow shop* bicritério *makespan* e tempo médio de fluxo.

De acordo com Tubino (2007), as organizações comumente são analisadas como um sistema que transforma, via um processamento, entradas (insumos) em saídas (produtos) úteis aos clientes. Este sistema é chamado de sistema produtivo.

Assim, a programação da produção dentro do sistema produtivo, refere-se à ordenação de tarefas a serem executadas, em uma ou diversas máquinas, considerando-se uma base de tempo, ou seja, trata-se da determinação de quando e onde cada operação necessária para a fabricação de um produto deve ser realizada. As tarefas são conhecidas, determinadas e devem ser executadas. Cada tarefa corresponde a um dado conjunto de operações que tem uma sequência a ser seguida para a execução completa.

2.1 Parâmetros dos problemas e indicadores de desempenho

Existem diversos **parâmetros para os problemas de programação da produção**, mas por conveniência e foco deste trabalho, serão apenas citadas as variáveis de entrada e as variáveis que descrevem a solução do problema tratado. De acordo com Farber e Coves (2005) e Fuchigami (2005), podemos descrever:

- Tarefa (J_j): uma atividade processada por uma máquina ou estação de trabalho é chamada de tarefa. Numa linha de produção, diferentes tarefas possuem diversos tempos de processamento, de acordo com o tipo de máquina. O número de tarefas

(*job*) é representado pela letra "*n*", ou seja, o conjunto de tarefas é denotado por: $J = \{J_1, J_2, ..., J_j, ..., J_n\}$.

- Máquina (M_k): cada recurso produtivo, estação de trabalho ou etapa de produção é denominada genericamente de máquina. O número de máquinas de um problema é representado pela letra "*m*", ou seja, o conjunto de máquinas é denotado por: $M = \{M_1, M_2, ..., M_k, ..., M_m\}$. No problema clássico de *flow shop*, as *m* máquinas estão alinhadas em série e todas as tarefas J_j têm que passar por todas as máquinas.

- Tempo de processamento (p_{jk}): tempo em que a tarefa J_j permanece na máquina M_k enquanto está sendo processada.

- Data de liberação ou *release date* (r_j): instante em que a tarefa J_j está pronta para ser processada.

- Tempo de *setup* (s_{jk}): tempo de preparação ou configuração, necessário para preparar a máquina M_k, que processará a tarefa J_j. Neste estudo, está sendo abordado o tempo de *setup* independente da sequência, ou seja, o tempo de *setup* depende apenas da tarefa a ser processada.

- Data de término ou *completion time* (C_{jk}): instante em que a tarefa J_j termina o seu processamento na máquina M_k. É utilizada a notação C_j para a data de término da última operação da tarefa. O *Completion time* se difere do *makespan*, pois o *makespan* é sempre o valor do término da última tarefa da última máquina, e refere-se a totalidade da duração da programação.

- Tempo de fluxo ou *flowtime* (F_j): tempo de permanência da tarefa na fábrica ou oficina, ou seja, resulta da diferença entre a data de término e a liberação da tarefa (Fj=Cj–rj)

Os **indicadores de desempenho** são critérios ou medidas utilizadas para se determinar a eficácia (qualidade da solução) de um determinado método de programação. Cada objetivo traçado pela empresa está atrelado a uma medida de desempenho que pode ser simples ou composta (multiobjetivo).

Para Davis, Aquilano e Chase (2001), as medidas de desempenho indicam a qualidade de um programa de produção, ou seja, são usadas para medir a eficácia de uma

decisão. Os indicadores mais comuns para aferir os métodos de solução, com base em Fuchigami (2005); Reid; Sanders (2005) e Gaither; Frazier (2002), encontram-se a seguir listados:

- *Makespan*: duração total da programação. Mede a eficiência operacional informando o tempo necessário para se executar um conjunto de tarefas. Equivale à maximização da utilização dos recursos;

- Tempo médio de fluxo ou tempo total de fluxo: a média ou a soma dos tempos de fluxo das tarefas, conforme já definido anteriormente, equivale à minimização do estoque em processamento, contribuindo com a redução de custos de armazenamento.

- Número médio de tarefas no sistema: analisa o estoque de produtos em processo e também afeta o *flowtime*. Se o número de tarefas do sistema for alto, maiores e mais longas serão as filas e, portanto, maiores serão os tempos de fluxo das tarefas.

- Número de ordens atrasadas: quantidade de ordens que deixaram de ser entregues ao cliente dentro do prazo determinado.

- Atraso da Tarefa: diferença entre o momento em que a tarefa é terminada e aquele em que ela supostamente seria concluída no prazo. Quando uma tarefa é concluída antes do programado, ela tem um adiantamento.

2.2 Notação de um problema de produção

Um problema de produção é denotado por meio de uma notação matemática. Pinedo (2012) organiza esses parâmetros em um trio $\alpha \mid \beta \mid \gamma$ que auxilia a classificação dos problemas de sequenciamento. O trio determina o problema, descrevendo "α" como o ambiente de produção com o número de máquinas ou estágios, β fornecendo detalhes das propriedades e limitações dos recursos e tarefas, e γ contendo o objetivo a ser otimizado (medida de desempenho, geralmente de minimização).

Na notação de três campos, o problema tratado nesta pesquisa é assim referido:

$$Fm|prmu, r_j, s_{jk}|\alpha C_{max} + (1-\alpha)\bar{F}, \quad (2.1)$$

onde Fm indica que é um ambiente *flow shop* com um número genérico *m* de máquinas, $prmu$ denota que é um *flow shop* permutacional, r_j é a restrição de presença de diferentes datas de liberação das tarefas e s_{jk} a restrição de tempos de *setup* independentes da sequência. A expressão $\alpha C_{max} + (1-\alpha)\bar{F}$ representa uma função objetivo ponderada, com peso α, e duas medidas de desempenho (*makespan* C_{max} e tempo médio de fluxo $\bar{F}$). Cabe ressaltar que o valor do C_{max} na função objetivo ponderada, citada anteriormente, é resultado da fórmula $C_{max} = max\ C_j - min r_j$, ou seja, o valor da duração total da programação é subtraído pela menor data de liberação. Já o *flowtime* é resultado da expressão $\bar{F} = \sum F_J/n$, ou seja, o tempo médio de fluxo é igual à soma do tempo de fluxo de cada tarefa dividido pelo número de tarefas. É importante citar que o peso da função objetivo, usualmente indicada por α, não é o mesmo da notação genérica α | β | γ. Este peso α geralmente assume valores entre 0 e 1.

2.3 Trabalhos publicados abordando *flow shop* bicritério com *makespan* e *flowtime*

Para uma sólida e estruturada pesquisa, realizou-se uma busca na literatura sobre os problemas bicritérios, tendo como função objetivo a minimização do *makespan* e do tempo médio de fluxo. Além disso, é importante ressaltar que essa revisão se aplicou apenas para problemas em ambiente *flow shop*, com número mínimo de 3 máquinas, independente das restrições. É possível encontrar na literatura específica uma inumerável quantidade de trabalhos abordando *flow shop* bicritério com *makespan* e *flowtime* com apenas duas máquinas. Como pode ser visto na revisão sobre *flow shop* multi-objetivo de Yenisey e Yagmahan (2014), todos os quinze artigos descritos pelos autores sobre *flow shop* com minimização do *makespan* e *flowtime* consideram apenas duas máquinas. Isto indica o potencial de pesquisa nos problemas com número de máquinas maior do que dois.

Ho e Chang (1991) criaram uma heurística baseada nas lacunas (*gaps*) entre a última tarefa da primeira máquina e as últimas das máquinas seguintes, ou seja, uma solução baseada em relações nos tempos de processamento das tarefas. Gangadharan e Rajendran (1994) desenvolveram uma heurística bem sucedida baseada no algoritmo

Simulated Annealing (SA), algoritmo este que visa minimizar *makespan* e *flowtime*. Rajendran (1995) desenvolveu um novo algoritmo para o problema de sequenciamento bicritério, buscando minimizar o *makespan* e o *flowtime*. O algoritmo proposto foi avaliado em um grande número de problemas aleatórios e de grande porte, e produziu melhores resultados que a heurística de Ho e Chang (1991).

Sridhar e Rajendran (1996) propuseram um algoritmo que apresenta um bom desempenho para a programação num sistema de manufatura celular baseado no tempo médio de fluxo (*flowtime*). Já Framinan et al. (2002) se basearam na heurística NEH (Nawaz et al., 1983) para construir dois algoritmos para a escolha de uma sequência que minimiza os custos associados ao *makespan* e ao *flowtime*.

Allahverdi (2003) propôs três heurísticas compostas por três fases. Uma sequência inicial é obtida na primeira fase, uma técnica de inserção é aplicada à sequência inicial para melhorar a solução, na segunda fase, e na terceira fase há uma troca de pares à sequência obtida a partir da segunda fase para melhorar ainda mais a solução. Segundo o autor, os experimentos computacionais mostram que tais heurísticas superam as existentes na literatura anteriores a sua pesquisa. Rajendran e Ziegler (2004) propuseram um algoritmo de colônias de formigas, onde a ideia principal foi simular o rastro de feromônio utilizado pelas formigas reais como um meio de comunicação e *feedback* entre formigas.

Yeh e Allahverdi (2004) desenvolveram limitantes inferiores e superiores e utilizaram o método *branch-and-bound* para chegar numa solução com até 16 tarefas, e os resultados computacionais mostram que essas relações de dominância são eficientes. Ravindran et al. (2005) criam os algoritmos HAMC1, HAMC2 e HAMC3 para a análise de eficácia. Os resultados dos problemas foram comparados com os procedimentos de soluções propostas pelo Rajendran (1995) e apresentaram melhor desempenho. No mesmo ano, Varadharajan e Rajendran (2005) criaram o *Multi-Objective Simulated-annealing Algorithm* (MOSA), que visa obter soluções eficientes por meio da implementação de uma função de probabilidade.

Pasupathy et al. (2006) analisaram o algoritmo genético denominado PAG – ALS Este algoritmo faz uso de um arquivo de soluções melhoradas por meio de busca local. Tangestiren et al. (2007) apresentaram uma heurística PSO (*particle swarm optimization*) para os mesmos problemas apresentados por Rajendran e Ziegler (2004). Eles

desenvolvem uma meta-heurística chamada de SPV *(smallest position value*), para habilitar o PSO, que aplicado nos problemas de sequenciamento minimizaria o *makespan* e o *flowtime*. Branco et al. (2008) apresentam o método MC-FS (Multi - Critério *Flow Shop*) para solução do problema multicritério a programação da produção em sistemas *no-wait flow shop* (situação onde as operações de uma determinada tarefa uma vez iniciada são processadas sem interrupções nas consecutivas máquinas).

Rajkumar et al. (2009) apresentam o algoritmo de melhoria genética (IGA). Baseado no algoritmo NEH devido ao bom desempenho em relação ao *makespan*, o algoritmo IGA é testado e o seu desempenho gera bons resultados. O algoritmo IGA foi comparado com os algoritmos HAMC1, HAMC2 e HAMC3 (de Varadharajan e Rajendran, 2005) e apresentou melhores resultados. Por fim, Eren (2010) desenvolveu um modelo de programação para o problema de *flow shop* bicritério: soma ponderada das datas de término e *makespan*. Os resultados dos testes computacionais mostraram que o modelo proposto é eficiente na resolução de problemas com até 18 tarefas e 06 máquinas. Para resolver os problemas de grande porte com até 100 tarefas e 10 máquinas, o autor cita que deveriam ser utilizados métodos heurísticos específicos para este fim.

A Figura 2 apresenta a organização dos trabalhos citados, em ordem cronológica e a relação entre eles. Os artigos são representados por blocos, com os nomes dos autores e o ano de publicação dos seus trabalhos, enquanto as setas indicam quando um trabalho serviu como base de comparação para o desenvolvimento de outro, ou seja, um trabalho contribuiu na elaboração do outro. Para exemplificar, podemos citar que o trabalho desenvolvido por Ho e Chang (1991) serviu de base para a elaboração do artigo do Gangadharan e Rajendran (1994).

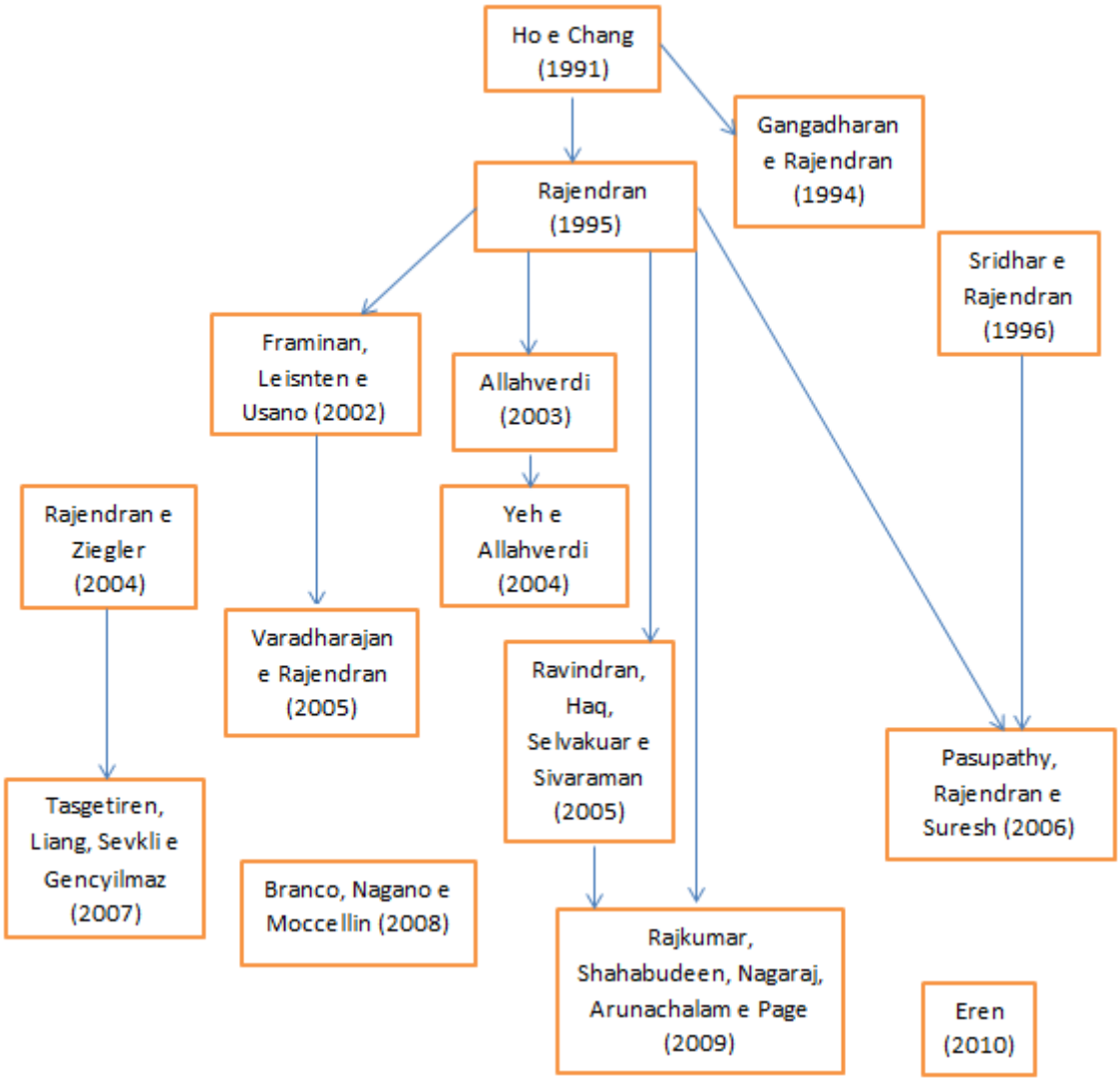

Figura 2 – Organização esquemática dos trabalhos bicritério *makespan* e *flowtime*. Fonte: elaborado pelos autores.

A Tabela 1 especifica os artigos apresentados na Figura 2, esclarecendo o ano de publicação do artigo, o(s) autor (es), as técnicas de solução, as restrições utilizadas em cada caso e por fim os pesos atribuídos aos valores de α em cada caso. Com esta tabela é mais fácil de identificar características mais peculiares de cada trabalho que compõe esta árvore de artigos bicritério.

Ano	Autor (es)	Técnica de Solução	Peso α
1991	Ho e Chang	Heurística baseada no Algoritmo CDS	0,5
1994	Gangadharan e Rajendran	Meta - Heurística: *Simulated Annealing*	0,5
1995	Rajendran	Heurística baseada no Algoritmo CDS	0,5
1996	Sridhar e Rajendran	Algoritmo Genético	0,5
2002	Framinan, Leisten e Usano	Heurística baseada no Algoritmo NEH	0,5
2003	Allahverdi	Heurística baseada no Algoritmo NEH	0,1; 0,5; 0,9
2004	Rajendran e Ziegler	Meta-Heurística: *Ant-colony*	0,5
2004	Yeh e Allahverdi	Método de solução exata *Branch-and-Bound*	0; 0,25; 0,50; 0,75; 1,0
2005	Varadharajan e Rajendran	Meta - Heurística: MOSA - M*ulti-Objective Simulated-Annealing Algorithm*	0,5
2005	Ravindran, Haq, Selvakuar e Sivaraman	Heurísticas HAMC1, HAMC2 e HAMC3	0,5
2006	Pasupathy, Rajendran e Suresh	Algoritmo Genético	0,5
2007	Tangestiren, Liang, Sevkli e Gencyilmaz	Heurística PSO - *Particle Swarm Optimization*	0,5
2008	Branco, Nagano e Moccellin	Heurística MC – FS - Multi-Critério *Flow Shop*	0; 0,25; 0,50; 0,75; 1,0
2009	Rajkumar	Algoritmo Genético	0,5
2010	Eren	Heurísticas SH-1, SH-2 e SH-3, baseadas no Algoritmo NEH	0,25; 0,5; 0,75

Tabela 1 – Detalhes da organização dos trabalhos bicritério *makespan* e *flowtime*. Fonte: elaborado pelos autores.

3. MÉTODOS DE SOLUÇÃO PROPOSTOS

3.1 Regras de Prioridade

As Regras de Prioridades definem a ordem em que as tarefas são realizadas nas máquinas. Não existem regras de sequenciamento que sejam eficientes em todas as situações, ou seja, com as diversas medidas de desempenho e em problemas com diferentes restrições. Segundo Tubino (2007), pesquisas mostram que a eficiência de uma regra de sequenciamento dependerá das informações das tarefas, dos tamanhos dos conjuntos de tarefas a serem processadas e das condições do sistema produtivo. Isso faz com que uma regra obtenha bons índices em uma situação, em outras não.

Para o problema em estudo, foram definidas oito regras, fundamentadas nas reconhecidas regras SPT (*Shortest Processing Time*) e LPT (*Longest Processing Time*), que foram adaptadas ao problema tratado neste trabalho. Os valores de ordenação definidos para cada tarefa J_j são apresentados na Tabela 2.

Regra	Ordenação	Critério
R_1	Crescente	r_j
R_2	Crescente	$\max\{r_j, s_{j1}\} + p_{j1}$
R_3	Crescente	$s_{j1} + p_{j1}$
R_4	Crescente	$\max\{r_j, s_{j1}\} + \sum_{k=2}^{m} s_{jk}$
R_5	Crescente	$\max\{r_j, s_{j1}\} + \sum_{k=1}^{m} p_{jk}$
R_6	Crescente	$\max\{r_j, s_{j1}\} + \sum_{k=2}^{m} s_{jk} + \sum_{k=1}^{m} p_{jk}$
R_7	Decrescente	$s_{jm} + p_{jm}$
R_8	--	Aleatória

Tabela 2 – Regras de Prioridade propostas – Fonte: elaborada pelos autores.

Uma expressão utilizada em várias regras é aquela para o cálculo do maior valor entre a data de liberação e o *setup* da tarefa na primeira máquina ($max\{r_j, s_{j1}\}$), que

representa o instante mais cedo de início do processamento da tarefa J_j. Esta expressão já desconsidera a parte antecipada do tempo de *setup* para que este intervalo não seja somado duas vezes. O critério utilizado em cada regra é explicado a seguir.

R_1: Ordenação em fila (crescente) pela data de liberação;

R_2: Ordenação crescente pela data de término da primeira operação considerada na primeira posição da sequência, ou seja, a soma do instante mais cedo de início da tarefa J_j com o tempo de processamento na primeira máquina;

R_3: Ordenação crescente pela soma dos tempos de processamento e de *setup* da tarefa J_j na primeira máquina;

R_4: Ordenação crescente pela soma do instante mais cedo de início da tarefa J_j e os tempos de *setup* da segunda máquina em diante;

R_5: Ordenação crescente pela soma do instante mais cedo de início da tarefa J_j e os tempos de processamento em todas as máquinas;

R_6: Ordenação crescente pela soma do instante mais cedo de início da tarefa J_j, os tempos de processamento em todas as máquinas e os tempos de *setup* da segunda máquina em diante;

R_7: Ordenação decrescente pela soma dos tempos de processamento e de *setup* da tarefa J_j na última máquina;

R_8: Ordenação aleatória, para fins de comparação.

A regras SPT é mais apropriada para quando se tem o objetivo de reduzir o tempo médio de fluxo, assim prioriza as tarefas de menores cargas nas primeiras máquinas, por isso é interessante uma ordenação crescente. Já a LPT programa as tarefas tendo como prioridade as com maior tempo de processamento, ordenando de forma decrescente. Os desempates ocorrem pela menor soma dos tempos de processamento (p_j) em todas as máquinas, e caso o empate persista, o desempate deverá ocorre pela menor soma dos tempos de *setup* (s_{jk}) em todas as máquinas.

3.2 Heurísticas Construtivas

As heurísticas H_1, H_2 e H_3 que serão apresentadas a seguir, baseiam-se nas heurísticas NEH, N&M e no trabalho de Eren (2010). A seguir, estão descritos os passos de cada heurística.

Heurística NEH

O método heurístico proposto por NEH (Nawaz; Enscore Jr.; Ham, 1983) propõe a solução heurística do problema Fm| |Cmax. A seguir, estão detalhados os passos do algoritmo NEH:

- Passo 1 – ordene as tarefas pela regra LPT (considerando a soma dos tempos de processamento em todas as máquinas).
- Passo 2 – com as duas primeiras tarefas da ordenação obtida, encontre a subsequência (entre as duas possíveis) com o melhor *makespan*.
- Passo 3 – sem alterar as posições relativas das tarefas já alocadas, insira a próxima tarefa da ordenação obtida no passo 1 em todas as posições possíveis da subsequência atual e considere a que fornece o melhor *makespan*.
- Passo 4 – repita o passo 3 até que todas tarefas estejam programadas.

Heurística N&M

Moccellin (1995) apresentou a seguinte proposição:

Limitante superior *UBX*:

Para qualquer sequência σ *com as tarefas* J_u *e* J_v *programadas respectivamente nas posições j e j+1, com j = 1, ..., n-1, tem-se o limitante superior para* X_{uv}^{k}:

$$UBX_{uv}^{k} = \max[0, UBX_{uv}^{k-1} + (p_{v(k-1)} - p_{uk})] \quad (3.1)$$

com $UBX_{uv}^{0} = 0$ *e k = 1, ..., m.*

Aplicando recursivamente esta relação para k = 1, ..., m, obtém-se o limitante superior UBX_{uv}^{m} para o tempo ocioso da última máquina M_m entre quaisquer pares adjacentes de tarefas J_u e J_v, independentemente das suas posições.

Sabe-se que para uma determinada sequência de tarefas, o *makespan* pode ser expresso por:

$$C_{\max} = \sum_{j=1}^{n} p_{jm} + \sum_{j=0}^{n-1} X_{[j][j+1]}^{m} \tag{3.2}$$

onde $X_{[j][j+1]}^{m}$ é o tempo ocioso da última máquina M_m entre o final da tarefa na posição j e o início da tarefa na posição j+1.

Se considerar $X_{[j][j+1]}^{m}$ como as "distâncias" entre as tarefas nas posições j e j+1 da sequência, o problema de programação em *flow shop* permutacional torna-se análogo ao Problema do Caixeiro Viajante e a sequência que minimiza o *makespan* é dada pela menor rota entre a tarefa fictícia 0 e a última $J_{[n]}$. Entretanto, evidentemente não se conhece os valores dos tempos ociosos $X_{[j][j+1]}^{m}$ antes de se definir uma sequência específica. O problema do caixeiro viajante pode ser entendido como o problema de um turista que deseja visitar um conjunto de países, passando exatamente uma vez por cada um e retornando ao ponto de partida no final do seu percurso.

Deste modo, para qualquer sequência de tarefas, pode-se estabelecer o seguinte limitante superior para o *makespan*:

$$UBM = \sum_{j=1}^{n} p_{jm} + \sum_{j=0}^{n-1} UBX_{[j][j+1]}^{m} \ . \tag{3.3}$$

Este problema pode ser resolvido heuristicamente pela analogia direta com o Problema do Caixeiro Viajante. Existem vários métodos de solução para esta finalidade, como o *Farthest Insertion Travelling Salesman Procedure (FITSP)* e o *Nearest Insertion Travelling Salesman Procedure (NITSP).*

De forma diferente, pode-se também calcular o *makespan* de uma determinada sequência por meio da seguinte expressão:

$$C_{\max} = \sum_{j=1}^{n} p_{[j]1} + \sum_{k=2}^{m} p_{[n]k} + \sum_{k=1}^{m-1} Y_{[n]}^{k} \tag{3.4}$$

onde $Y_{[n]}^{k}$ é o tempo de espera da tarefa na *n*-ésima posição da sequência, entre o fim da operação na máquina M_k e o início da operação na máquina M_{k+1}.

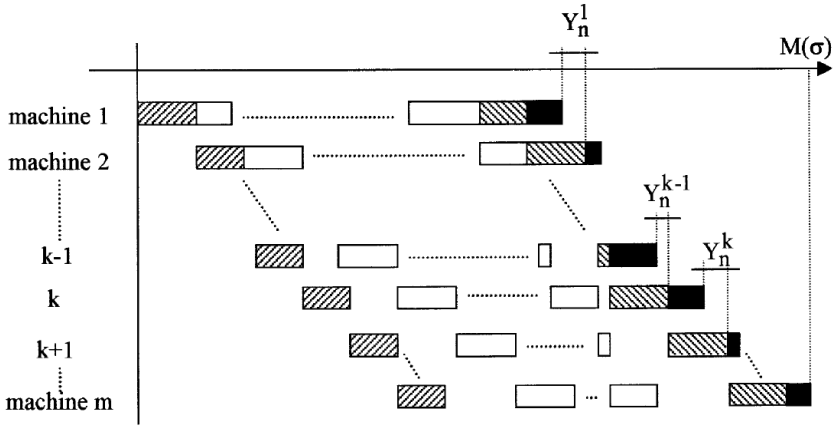

Figura 3 - Tempo total para completar a programação (NAGANO; MOCCELLIN, 2002).

Aqui também, é claro, não se conhecem previamente os valores de $Y_{[n]}^{k}$ antes de se considerar uma determinada sequência. Contudo, pelas relações de viabilidade entre pares de tarefas adjacentes, pode-se calcular um limitante inferior para o tempo de espera entre essas tarefas sem precisar considerar as suas posições na sequência.

Com base nisto e no limitante superior *UBX*, Nagano e Moccellin (2002) definiram a seguinte proposição:

Limitante inferior *LBY*:

Para uma sequência arbitrária σ *com as tarefas* J_u *e* J_v *programadas respectivamente nas posições j e j+1, com j = 1, ..., n-1, tem-se o limitante inferior para* Y_{uv}^{k}:

$$LBY_{uv}^{k} = \max[0, (p_{u(k+1)} - p_{vk}) - UBX_{uv}^{k}]. \tag{3.5}$$

com k = 1, ..., m.

Proposições da Heurística N&M

Nagano e Moccellin (2002) propuseram a heurística construtiva N&M para o problema $Fm||C_{\max}$, semelhante à NEH. A única diferença da NEH são nos passos 1 e 2, em relação à ordenação inicial das tarefas a serem programadas. A Heurística N&M, apresentada na Figura 4, utiliza o limitante inferior *LBY*.

HEURÍSTICA N&M

PASSO 1 – Calcule para cada tarefa $I_v = \sum_{k=1}^{m} p_{vk} - \max_{\substack{u=1,\ldots,n \\ u \neq v}} \left\{ \sum_{k=1}^{m-1} LBY_{uv}^{k} \right\}$.

PASSO 2 – Sequencie as tarefas em ordem decrescente de I_v.

PASSO 3 – Com as duas primeiras tarefas, encontre a melhor sequência parcial, calculando o *makespan* para as duas possibilidades.

Mantenha a posição relativa desta melhor sequência parcial nos passos seguintes.

PASSO 4 – Insira a próxima tarefa da lista ordenada em todas as posições possíveis da sequência parcial, mantendo a posição relativa das tarefas já consideradas.

Mantenha a melhor sequência parcial em relação ao *makespan*.

PASSO 5 – Se todas as tarefas já foram programadas, PARE. Caso contrário, volte ao Passo 4.

FIGURA 4 - Algoritmo N&M (NAGANO; MOCCELLIN, 2002).

Heurísticas de Eren (2010)

Eren (2010) desenvolveu as heurísticas SH1, SH2 e SH3 para o problema de *flow shop* com setup dependente e bicritério com *makespan* e *flowtime*, com base na Heurística NEH. A diferença fundamental está na ordenação inicial, que considera respectivamente as seguintes Regras de Prioridade:

- TSPT - *Total Smallest Processing Time* (menor tempo de processamento total): prioriza a tarefa com a menor soma dos tempos de processamento de todas as máquinas;

- LSPT - *Last Smallest Processing Time* (último menor tempo de processamento): prioriza a tarefa com o menor tempo de processamento na última máquina;

- TLPT - *Total Longest Processing Time* (maior tempo total de processamento): prioriza as tarefas com a maior soma dos tempos de processamento de todas as máquinas.

Conforme apresentado anteriormente, a heurística N&M apresenta uma nova proposta de ordenação inicial para o método de inserção da heurística NEH, que é clássica e já foi muito utilizada como base para novas heurísticas e adaptações para problemas com diferentes variáveis. É este o caso das heurísticas de Eren (2010), que adaptaram a ideia da heurística NEH com diferentes ordenações iniciais.

Assim, salienta-se que as heurísticas H_1, H_2 e H_3 foram desenvolvidas nesta pesquisa com base nas heurísticas NEH, N&M e nas SH1, SH2 e SH3 de Eren (2010).

3.2.1 Heurística H_1

- ***Passo 1*** **–** Ordene as tarefas pela regra R_2, ou seja, pela ordem crescente do valor $(max\{r_j, s_{j1}\}+ p_{j1})$.

- ***Passo 2*** **–** Com as duas primeiras tarefas da ordenação obtida, encontre a subsequência (entre as duas possíveis) com o melhor valor da função objetivo $\mathbf{Z = \alpha\, C_{max} + (1 - \alpha)\, \bar{F}}$**.**

- ***Passo 3*** **–** Sem alterar as posições relativas das tarefas já alocadas, insira a próxima tarefa da ordenação obtida no Passo 1 em todas as posições possíveis da subsequência e considere a programação que fornece a melhor função objetivo Z.

- ***Passo 4*** **–** Repita o Passo 3 até que todas as tarefas estejam programadas.

Ilustração Numérica

- **Heurística H_1**

Considere um *flow shop* com três máquinas e cinco tarefas, de acordo com os dados da Tabela 3, com o objetivo de minimizar o *makespan* e o tempo médio de fluxo com pesos iguais, ou seja, $\alpha = 0{,}5$:

	J_1	J_2	J_3	J_4	J_5
p_{j1}	4	7	2	3	4
p_{j2}	2	1	5	4	3
p_{j3}	7	2	4	3	2
s_{j1}	4	5	3	2	5
s_{j2}	2	7	2	3	4
s_{j3}	3	6	5	4	2
r_j	8	6	7	10	9

Tabela 3 – Tempos de Processamento, *Setup* e Liberação do Problema – Fonte: Elaborado pelos autores.

Passo 1: aplicando a regra R_2, teremos a sequência inicial $\{J_3 - J_1 - J_5 - J_4 - J_2\}$, com desempates pela menor soma dos tempos de processamento (p_j) em todas as máquinas, e em seguida, caso haja necessidade, pela menor soma dos tempos de *setup* (s_{jk}) em todas as máquinas.

Passo 2: Entre as duas subsequências possíveis com as duas primeiras tarefas da ordenação inicial $\{ J_1 - J_3 \}$ e $\{ J_3 - J_1 \}$, a segunda obteve melhor resultado, com **Z = 22**. Para que sejam calculados o *makespan* e o tempo médio de fluxo foram elaborados gráficos de Gantt para todos os passos, conforme Figuras 5 e 6, a seguir.

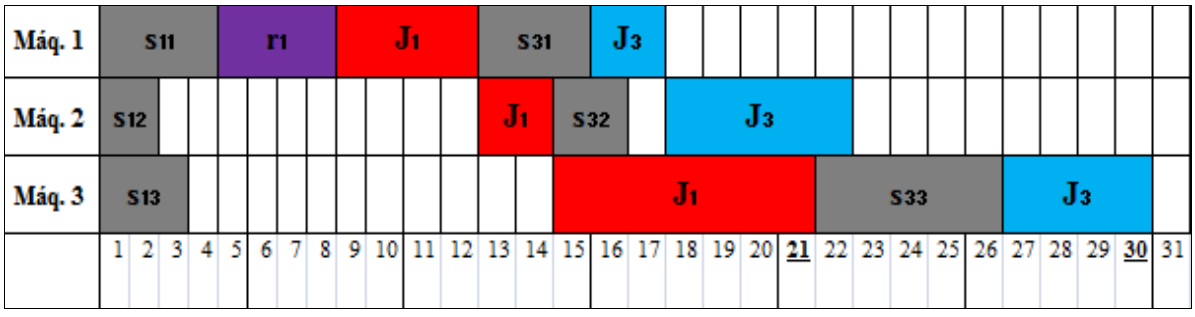

Figura 5 – Subsequência $\{J_1 - J_3\}$ – Fonte: elaborado pelos autores.

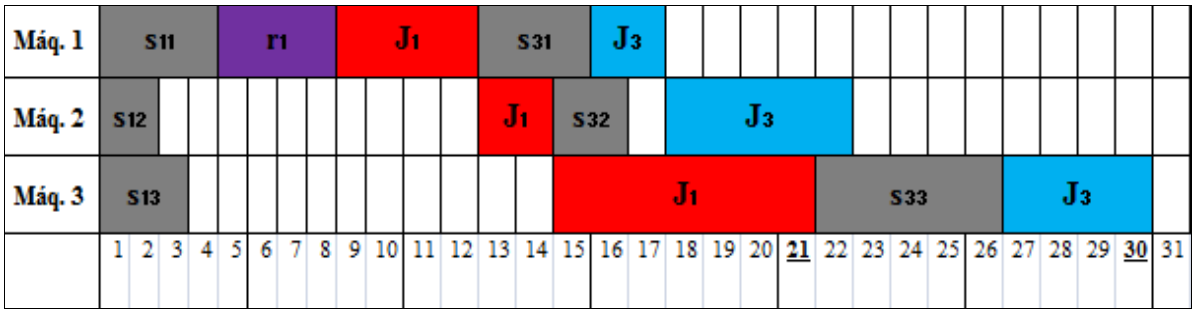

Figura 6 – Subsequência $\{J_3 - J_1\}$ – Fonte: elaborado pelos autores.

Passo 3:

Iteração 1 - Sem alterar as posições relativas das tarefas já alocadas, neste caso $\{J_3 - J_1\}$, inserindo a próxima tarefa obtida no passo 1 (tarefa J_5), encontramos as três subsequências $\{J_3 - J_1 - J_5\}$, $\{J_3 - J_5 - J_1\}$ e $\{J_5 - J_3 - J_1\}$. Nesse caso a subsequência $\{J_3 - J_1 - J_5\}$ obteve melhor desempenho, com **Z = 25,50**;

Iteração 2 - Sem alterar as posições relativas das tarefas já alocadas, neste caso $\{J_3 - J_1 - J_5\}$, inserindo a próxima tarefa obtida no passo 1 (tarefa J_4), encontramos as quatro subsequências $\{J_3 - J_1 - J_5 - J_4\}$, $\{J_3 - J_1 - J_4 - J_5\}$, $\{J_3 - J_4 - J_1 - J_5\}$ e $\{J_4 - J_3 - J_1 - J_5\}$. Nesse caso as subsequências $\{J_3 - J_1 - J_5 - J_4\}$ e $\{J_3 - J_4 - J_1 - J_5\}$ obtiveram os melhores desempenhos, com **Z = 30,62**;

Iteração 3 - Sem alterar as posições relativas das tarefas já alocadas, neste caso $\{J_3 - J_1 - J_5 - J_4\}$, inserindo a próxima tarefa obtida no passo 1 (tarefa J_2), encontramos as cinco subsequências $\{J_3 - J_1 - J_5 - J_4 - J_2\}$, $\{J_3 - J_1 - J_5 - J_2 - J_4\}$, $\{J_3 - J_1 - J_2 - J_5 - J_4\}$, $\{J_3 - J_2 - J_1 - J_5 - J_4\}$ e $\{J_2 - J_3 - J_1 - J_5 - J_4\}$. Nesse caso a subsequência $\{J_3 - J_1 - J_5 - J_4 - J_2\}$ obteve melhor desempenho, com **Z = 36,90**. Após a inserção da tarefa J_2 na interação 3 da subsequência $\{J_3 - J_4 - J_1 - J_5\}$, o valor da função objetivo foi igual, ou seja, **Z = 36,90.**

3.2.2 Heurística H_2

- ***Passo 1*** **–** Ordene as tarefas pela regra R_5.
- ***Passo 2*** **–** Com as duas primeiras tarefas da ordenação obtida, encontre a subsequência (entre as duas possíveis) com o melhor valor da função objetivo **Z = $\alpha\, C_{max} + (1 - \alpha)\, \bar{F}$.**

- ***Passo 3*** **–** Sem alterar as posições relativas das tarefas já alocadas, insira a próxima tarefa da ordenação obtida no Passo 1 em todas as posições possíveis da subsequência e considere a programação que fornece a melhor função objetivo Z.

- ***Passo 4*** **–** Repita o Passo 3 até que todas as tarefas estejam programadas.

Ilustração Numérica

- **Heurística H_2**

Para a ilustração numérica da H_2, considerar também um *flow shop* com três máquinas e cinco tarefas, de acordo com os tempos de processamento dados na Tabela 3, e pesos iguais, ou seja, $\alpha = 0{,}5$:

Passo 1: aplicando a regra R_5, teremos a sequência inicial $\{J_2 - J_5 - J_3 - J_4 - J_1\}$, com desempates pela menor soma dos tempos de processamento (p_j) em todas as máquinas, e em seguida, caso haja necessidade, pela menor soma dos tempos de *setup* (s_{jk}) em todas as máquinas.

Passo 2: Entre as duas subsequências possíveis com as duas primeiras tarefas da ordenação inicial $\{ J_2 - J_5 \}$ e $\{ J_5 - J_2 \}$, a primeira obteve melhor resultado, com **Z = 21,25**. Para que sejam calculados o *makespan* e o tempo médio de fluxo foram elaborados gráficos de Gantt para todos os passos, conforme Figuras 7 e 8, a seguir.

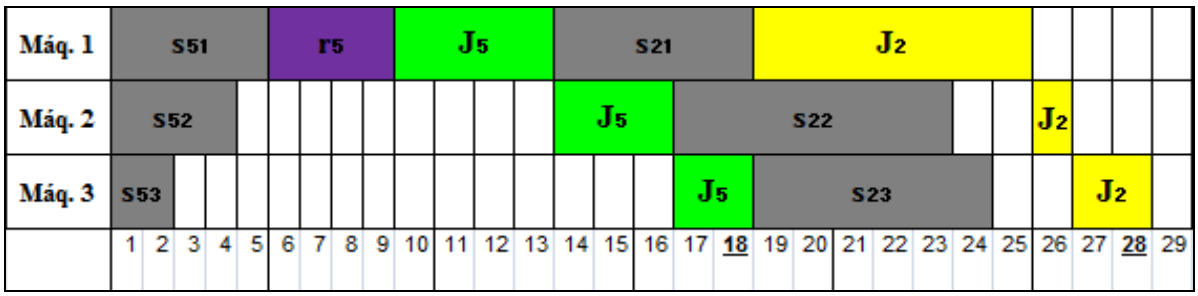

Figura 7 – Subsequência $\{J_5 - J_2\}$ – Fonte: elaborado pelos autores.

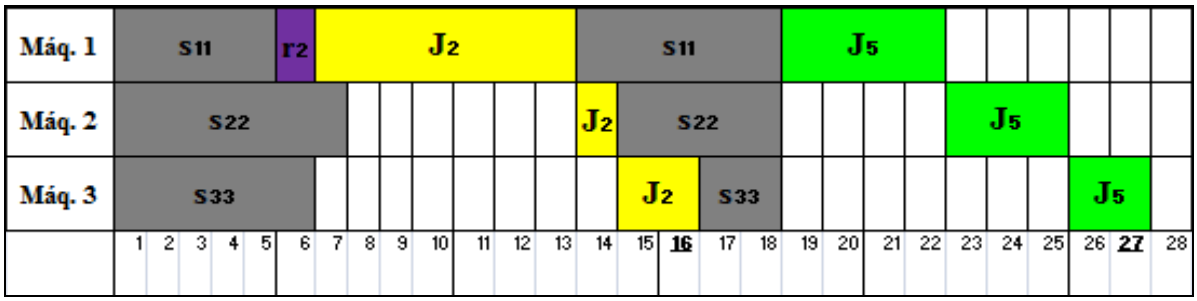

Figura 8 – Subsequência $\{J_2 - J_5\}$ – Fonte: elaborado pelos autores.

Passo 3:

Iteração 1 - Sem alterar as posições relativas das tarefas já alocadas, neste caso $\{J_2 - J_5\}$, inserindo a próxima tarefa obtida no passo 1 (tarefa J_3), encontramos as três subsequências $\{J_2 - J_5 - J_3\}$, $\{J_2 - J_3 - J_5\}$ e $\{J_3 - J_2 - J_5\}$. Nesse caso a subsequência $\{J_2 - J_3 - J_5\}$ obteve melhor desempenho, com **Z = 25,50**;

Iteração 2 - Sem alterar as posições relativas das tarefas já alocadas, neste caso $\{J_2 - J_3 - J_5\}$, inserindo a próxima tarefa obtida no passo 1 (tarefa J_4), encontramos as quatro subsequências $\{J_2 - J_3 - J_5 - J_4\}$, $\{J_2 - J_3 - J_4 - J_5\}$, $\{J_2 - J_4 - J_3 - J_5\}$ e $\{J_4 - J_2 - J_3 - J_5\}$. Nesse caso a subsequência $\{J_2 - J_4 - J_3 - J_5\}$ obteve melhor desempenho, com **Z = 30,12**;

Iteração 3 - Sem alterar as posições relativas das tarefas já alocadas, neste caso $\{J_2 - J_4 - J_3 - J_5\}$, inserindo a próxima tarefa obtida no passo 1 (tarefa J_1), encontramos as cinco subsequências $\{J_2 - J_4 - J_3 - J_5 - J_1\}$, $\{J_2 - J_4 - J_3 - J_1 - J_5\}$, $\{J_2 - J_4 - J_1 - J_3 - J_5\}$, $\{J_2 - J_1 - J_4 - J_3 - J_5\}$ e $\{J_1 - J_2 - J_4 - J_3 - J_5\}$. Nesse caso as subsequências $\{J_2 - J_4 - J_3 - J_5 - J_1\}$ e $\{J_2 - J_4 - J_3 - J_1 - J_5\}$ obtiveram, os melhores desempenhos, com **Z = 37,70**.

3.2.3 Heurística H_3

- ***Passo 1*** **–** Ordene as tarefas pela regra R_6.
- ***Passo 2*** **–** Com as duas primeiras tarefas da ordenação obtida, encontre a subsequência (entre as duas possíveis) com o melhor valor da função objetivo **Z = $\alpha C_{max} + (1 - \alpha) \bar{F}$.**

- ***Passo 3*** **–** Sem alterar as posições relativas das tarefas já alocadas, insira a próxima tarefa da ordenação obtida no Passo 1 em todas as posições possíveis da subsequência e considere a programação que fornece a melhor função objetivo Z.

- ***Passo 4*** **–** Repita o Passo 3 até que todas as tarefas estejam programadas.

Ilustração Numérica

- **Heurística H_3**

Para a ilustração numérica da H_3, considerar também um *flow shop* com três máquinas e cinco tarefas, de acordo com os tempos de processamento dados na Tabela 3, e pesos iguais, ou seja, $\alpha = 0,5$:

Passo 1: aplicando a regra R_6, teremos a sequência inicial $\{J_5 - J_3 - J_1 - J_4 - J_2\}$, com desempates pela menor soma dos tempos de processamento (p_j) em todas as máquinas, e em seguida, caso haja necessidade, pela menor soma dos tempos de *setup* (s_{jk}) em todas as máquinas.

Passo 2: Entre as duas subsequências possíveis com as duas primeiras tarefas da ordenação inicial $\{ J_5 - J_3 \}$ e $\{ J_3 - J_5 \}$, a segunda obteve melhor resultado, com **Z = 18,75**. Para que sejam calculados o *makespan* e o tempo médio de fluxo foram elaborados gráficos de Gantt para todos os passos, conforme Figuras 9 e 10, a seguir.

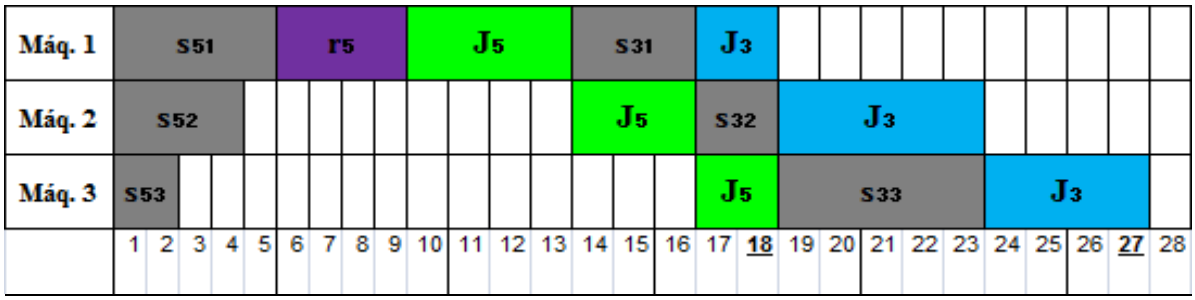

Figura 9 – Subsequência $\{J_5 - J_3\}$ – Fonte: elaborado pelos autores.

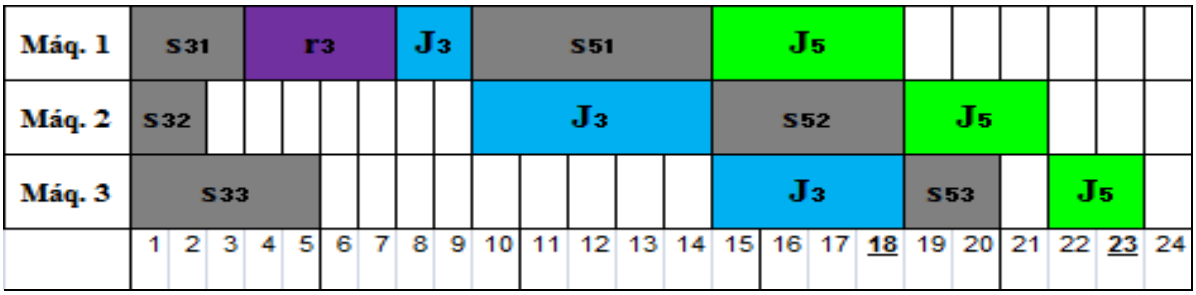

Figura 10 – Subsequência $\{J_3 - J_5\}$ – Fonte: elaborado pelos autores.

Passo 3:

Iteração 1 - Sem alterar as posições relativas das tarefas já alocadas, neste caso $\{J_3 - J_1\}$, inserindo a próxima tarefa obtida no passo 1 (tarefa J_5), encontramos as três subsequências $\{J_3 - J_1 - J_5\}$, $\{J_3 - J_5 - J_1\}$ e $\{J_5 - J_3 - J_1\}$. Nesse caso a subsequência $\{J_3 - J_1 - J_5\}$ obteve melhor desempenho, com **Z = 25,50**;

Iteração 2 - Sem alterar as posições relativas das tarefas já alocadas, neste caso $\{J_3 - J_1 - J_5\}$, inserindo a próxima tarefa obtida no passo 1 (tarefa J_4), encontramos as quatro subsequências $\{J_3 - J_1 - J_5 - J_4\}$, $\{J_3 - J_1 - J_4 - J_5\}$, $\{J_3 - J_4 - J_1 - J_5\}$ e $\{J_4 - J_3 - J_1 - J_5\}$. Nesse caso as subsequências $\{J_3 - J_1 - J_5 - J_4\}$ e $\{J_3 - J_4 - J_1 - J_5\}$ obtiveram os melhores desempenhos, com **Z = 30,62**;

Iteração 3 - Sem alterar as posições relativas das tarefas já alocadas, neste caso $\{J_3 - J_1 - J_5 - J_4\}$, inserindo a próxima tarefa obtida no passo 1 (tarefa J_2), encontramos as cinco subsequências $\{J_3 - J_1 - J_5 - J_4 - J_2\}$, $\{J_3 - J_1 - J_5 - J_2 - J_4\}$, $\{J_3 - J_1 - J_2 - J_5 - J_4\}$, $\{J_3 - J_2 - J_1 - J_5 - J_4\}$ e $\{J_2 - J_3 - J_1 - J_5 - J_4\}$. Nesse caso a subsequência $\{J_3 - J_1 - J_5 - J_4 - J_2\}$ obteve melhor desempenho, com **Z = 36,90**. Após a inserção da tarefa J_2 na interação 3 da subsequência $\{J_3 - J_4 - J_1 - J_5\}$, o valor da função objetivo foi igual, ou seja, **Z = 36,90.**

3.2.4 Heurística H_4

A heurística H_4 foi baseada na heurística CDS de Campbell, Dudek e Smith (1970), que por sua vez teve como base o pioneiro Johnson (1954), que encontra soluções ótimas para o problema $F_2||C_{max}$. O algoritmo CDS utiliza o Algoritmo de Johnson de forma heurística e obtém $m - 1$ programações viáveis, escolhendo a melhor.

O Algoritmo de Johnson original é descrito a seguir:

Algoritmo de Johnson:

Passo 1 – Determine $\min\{p_{jk}\}$.

Passo 2 – Se $\min\{p_{jk}\}$ pertence à primeira máquina, coloque a tarefa na primeira posição disponível da sequência, senão coloque-a na última posição disponível.

Passo 3 – Desconsiderando a tarefa alocada, repita os Passos 1 e 2 até que todas as tarefas sejam programadas.

O algoritmo CDS fornece uma solução heurística do problema $Fm||C_{\max}$. A seguir, estão detalhados os passos do algoritmo CDS original.

Algoritmo CDS:

- O algoritmo CDS utiliza o Algoritmo de Johnson (descrito anteriormente) de forma heurística, e obtém m–1 programações viáveis, escolhendo a melhor.

- Etapa 1 – Aplique o Algoritmo de Johnson considerando somente a primeira e a última máquina (as demais são ignoradas).
- Etapa 2 – Aplique o Algoritmo de Johnson considerando a soma dos tempos de processamento da primeira e da segunda máquina e da última e a penúltima máquina.
- Etapa (m–1) – Aplique o Algoritmo de Johnson considerando a soma dos tempos de processamentos das (m–1) primeiras máquinas e das (m–1) últimas máquinas.
- Após as m–1 etapas, considere a programação com o melhor *makespan*.

- Por exemplo, se o problema tiver 3 máquinas (m=3), o algoritmo CDS obterá 2

programações possíveis (m–1) e considerará a de melhor *makespan*. Assim, com 3 máquinas, o algoritmo CDS terá 2 etapas.

Finalmente, é apresentado o roteiro da heurística H_4 desenvolvida.

Heurística H_4

Passo 1 **–** Para cada etapa h, de 1 a m–1, faça:

Etapa 1 **–** Aplique o Algoritmo de Johnson a duas máquinas fictícias, cujos tempos de processamento das tarefas são:

$$M1: \max\{r_j, s_{j1}\} + p_{j1} \tag{3.6}$$

$$M2: p_{jm} \tag{3.7}$$

Com a sequência completa obtida e os dados do problema original, calcule a função objetivo.

Etapas *h* (h = 2, ..., m–1) **–** Aplique o Algoritmo de Johnson a duas máquinas fictícias, cujos tempos de processamento das tarefas são:

$$M1: \max\{r_j, s_{j1}\} + \sum_{k=2}^{h} s_{jk} + \sum_{k=1}^{h} p_{jk} \tag{3.8}$$

$$M2: \sum_{k=m-h+1}^{m} p_{jk} \tag{3.9}$$

Com a sequência completa obtida e os dados do problema original, calcule a função objetivo.

Passo 2 **–** Das m–1 etapas, considere como solução final a sequência que forneceu o melhor valor da função objetivo.

Ilustração Numérica

- **Heurística H_4**

Para a ilustração numérica da Heurística H_4, considere também um *flow shop* com três máquinas e cinco tarefas, utilizando os mesmos dados (Tabela 3) da ilustração da heurística H_1. A resolução do problema é de acordo com a heurística proposta H_4, baseado na heurística CDS.

Como o problema exemplo possui três máquinas, o passo 1 do algoritmo terá duas etapas (m–1).

Passo 1:

Etapa 1 - Aplicando o algoritmo de Johnson adaptado a este problema, teremos os seguintes valores dos tempos de processamento fictícios para as máquinas M_1: 12 – 13 – 9 – 13 – 13 e M_2: 7 – 2 – 4 – 3 – 2, e a sequência gerada será a $\{J_1 - J_3 - J_4 - J_5 - J_2\}$ realizando os desempates pelo índice.

Etapa 2 - Aplicando as expressões das etapas h do algoritmo, conforme descritas anteriormente, obtém-se os valores dos tempos de processamento fictícios para as máquinas M_1: 16 – 21 – 16 – 20 – 20 e M_2: 9 – 3 – 9 – 7 – 5. De acordo com o mesmo Algoritmo de Johnson, a sequência gerada é a $\{J_3 - J_1 - J_4 - J_5 - J_2\}$, com desempates também pelo índice. A seguir, foram ilustradas as sequências para uma melhor visualização do problema, conforme Figuras 11 e 12, a seguir.

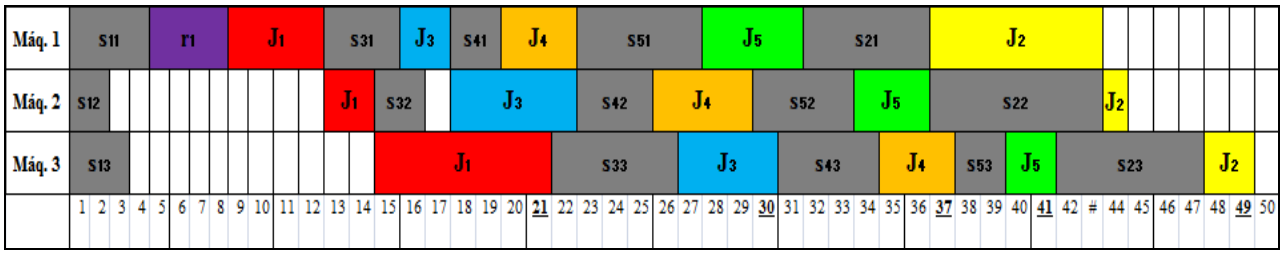

Figura 11 – Sequência $J_1 - J_3 - J_4 - J_5 - J_2$ (Etapa 1) – Fonte: Elaborado pelos autores.

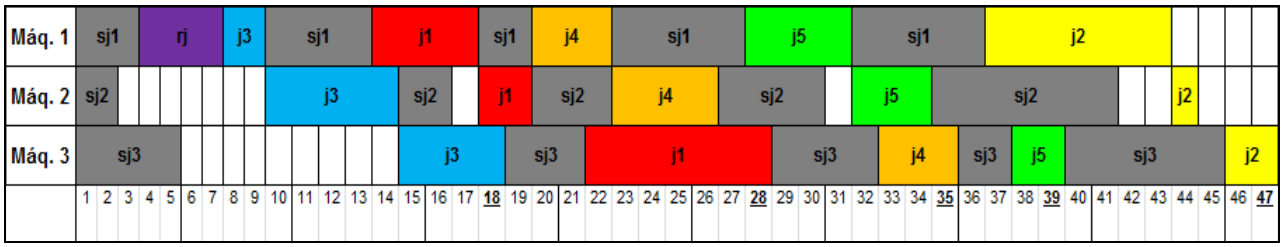

Figura 12 – Sequência $J_3 – J_1 – J_4 – J_5 – J_2$ (Etapa 2) – Fonte: Elaborado pelos autores.

A sequência que apresentou o menor valor para a função objetivo é a $\{J_3 – J_1 – J_4 – J_5 – J_2\}$, com **Z = 37,2**, enquanto a sequência $\{J_1 – J_3 – J_4 – J_5 – J_2\}$ apresentou **Z = 39,3**.

4. EXPERIMENTAÇÃO COMPUTACIONAL E RESULTADOS

A experimentação computacional desta pesquisa foi conduzida em duas fases. Na primeira, foram implementadas as oito Regras de Prioridade propostas e apresentadas no Capítulo 3. Dentre estas, foram escolhidas as três melhores como base para o desenvolvimento das Heurísticas Construtivas. Assim, a segunda fase da experimentação computacional se refere à codificação das quatro heurísticas definidas e ilustradas no Capítulo 3.

4.1 Parâmetros de Experimentação

Na experimentação computacional, tanto das Regras de Prioridade como das Heurísticas Construtivas desenvolvidas, foram testados e avaliados 39.600 problemas, definidos pelo número de tarefas (*n*), número de máquinas (*m*), intervalos de datas de liberação das tarefas (*r*) e intervalos de tempos de *setup* (*s*). Os problemas foram divididos em dois grupos, sendo o primeiro (Grupo 1) com problemas de pequeno porte, tendo sido encontrada a solução ótima por meio de enumeração completa, e o segundo (Grupo 2) com problemas de médio e grande porte. É importante salientar que a principal característica que difere o Grupo 1 do Grupo 2 é a comparação com a solução ótima e não o número de tarefas e de máquinas.

Como na primeira fase da implementação computacional os problemas foram gerados por meio de software construído especificamente para esta finalidade, optou-se por manter na segunda a fase a mesma base de dados dos problemas, para viabilizar a comparação entre os resultados dos métodos de solução.

Porém, é importante salientar que existe uma importante base de dados, bastante utilizada nos problemas de programação da produção, que foi proposta por Taillard (1993). O banco de dados de Taillard é composto por 120 problemas para o *flow shop* tradicional, divididos em 12 classes, definidas pela combinação entre o número de tarefas e máquinas de cada classe, totalizando 10 problemas por classe que variam os seus números de tarefas e máquinas.

No Grupo 1, foram gerados 18.000 problemas, cujos parâmetros foram: 5, 6, 7, 8 e 10 tarefas e 2, 3, 5 e 10 máquinas. E no Grupo 2, foram gerados 21.600 problemas, com os

seguintes parâmetros: 15, 20, 30, 50, 80 e 100 tarefas e 5, 10, 15 e 20 máquinas. Em ambos os grupos, foram considerados três intervalos de datas de liberação, U[1, 49], U[1,99] e U[1,199], três intervalos de tempos de *setup*, U[1, 49], U[1, 99] e U[1, 149] e um intervalo fixo para os tempos de processamento, com valores inteiros uniformemente distribuídos em U[1, 99].

Em ambos os grupos, foram utilizadas cinco opções de valores para o parâmetro α, que, conforme já mencionado, representa o peso das medidas da função objetivo. A Tabela 4 apresenta os valores considerados.

Opção	α	1 - α
1	0,00	1,00
2	0,25	0,75
3	0,50	0,50
4	0,75	0,25
5	1,00	0,00

Tabela 4 – Opções dos Valores de Alfa – Fonte: Elaborado pelos autores.

Na opção 1, tem-se peso zero (α = 0) para o critério *makespan* e peso 1 (1–α = 1) para o critério *flowtime*. Isso resulta no problema monocritério (apenas *flowtime*) com a notação $Fm|prmu, r_j, s_{jk}|\overline{F}$. Nas opções 2, 3 e 4, o problema continua bicritério, alternando os pesos nas opções 2 e 4, e com o mesmo peso na opção 3. Já na opção 5, ocorre o oposto da opção 1, ou seja, o *makespan* tem peso 1 (α = 1) e o *flowtime* com peso zero (1–α = 0), resultando no problema de minimização do *makespan* com notação $Fm|prmu, r_j, s_{jk}|C_{max}$.

Os resultados obtidos na experimentação computacional foram analisados por meio de Desvio Relativo Percentual - *RPD* (*relative percentage deviation*). O desvio relativo mede a variação correspondente à melhor solução obtida pelos métodos. Se o desvio relativo da solução de um método é igual a zero para um determinado problema, significa que tal método forneceu o menor valor da função objetivo, ou seja, o algoritmo apresentou a melhor programação.

Para cada problema, o *RPD* de determinado método no Grupo 1 foi assim calculado:

$$RPD_{G1} = \frac{Z - Z^*}{Z^*}, \tag{4.1}$$

onde Z é o valor da função objetivo do método avaliado e o Z^* é o valor da solução ótima do problema.

Enquanto para o Grupo 2, o cálculo foi elaborado da seguinte forma:

$$RPD_{G2} = \frac{Z - Z^b}{Z^b}, \tag{4.2}$$

onde Z é o valor da função objetivo do método avaliado e o Z^b a melhor solução obtida para o problema.

De acordo com esses parâmetros, todos os problemas foram gerados aleatoriamente e resolvidos por meio de um *software* denominado "**Simulador Industrial Fm_rj_sjk_Cmax_F**", construído especificamente para esta finalidade durante o período da pesquisa. Este *software* gera os arquivos de entrada com os dados dos problemas e produz a saída de arquivos comparativos com o valor da função objetivo e o tempo de computação de cada método para cada problema. Utilizou-se o sistema operacional *Windows* e a linguagem de programação *Delphi.*

Na primeira fase da implementação computacional, a configuração do computador utilizado são as seguintes: processador Pentium Dual - Core da Intel com 2 GHz de frequência e 3 GB de memória RAM. E na segunda fase, as configurações do computador são: processador Pentium Core I5 da Intel com 3 GHz de frequência e 8 GB de memória RAM.

4.2 Análise dos resultados das Regras de Prioridade

Após as execuções computacionais, duas Regras de Prioridade ganharam destaque. As regras R_2 e R_5 obtiveram melhor desempenho em relação às demais, tanto no Grupo 1 quanto no Grupo 2. A seguir, a Tabela 5 ilustra numericamente os resultados globais, destacando os melhores e piores. Destacados na cor azul estão os melhores desempenhos e de verde os segundos melhores. Em vermelho, estão os piores resultados.

	Alfa	R1	R2	R3	R4	R5	R6	R7	R8
GRUPO 1	0,0	17,4	9,4	13,0	16,5	7,7	10,4	25,1	23,2
	0,25	13,9	7,6	10,5	14,2	7,9	10,4	17,5	18,4
	0,5	12,5	7,3	9,8	13,6	9,1	11,5	13,4	16,3
	0,75	12,0	7,6	9,9	13,7	10,6	12,9	11,0	15,4
	1,0	12,1	8,2	10,3	14,2	12,1	14,4	9,4	15,0
GRUPO 2	0,0	7,6	4,7	5,6	5,7	1,2	1,9	10,6	8,7
	0,25	5,5	2,9	3,7	4,6	1,4	2,2	6,8	6,4
	0,5	4,5	2,2	2,8	4,3	2,0	2,9	4,6	5,3
	0,75	4,2	2,0	2,6	4,5	2,9	3,9	3,4	4,9
	1,0	4,2	2,2	2,7	5,0	3,8	4,9	2,7	4,8

Tabela 5 – Resultados da Análise Global do RPD - Fonte: Elaborado pelos autores.

Na Tabela 5, no que se refere ao Grupo 1, é notável que as regras R_2 e R_5, mostram-se as mais produtivas. Apenas na opção de alfa igual a 0,0 que a regra R_5 foi levemente superior à regra R_2, pois nos demais casos do Grupo 1, a regra R_2 obteve os melhores resultados. Isto pode também ser observado na Figura 13 que mostra o desempenho de cada método do Grupo 1.

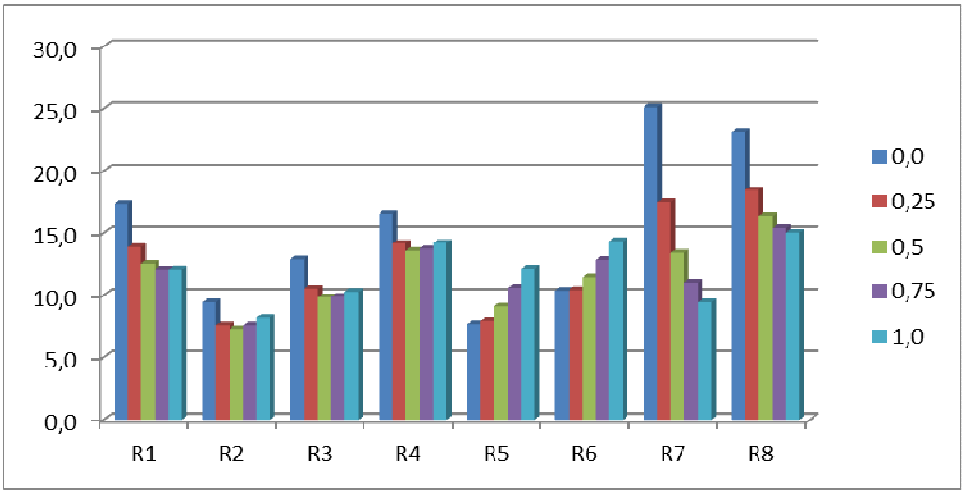

Figura 13 – Comparação dos resultados globais do RPD dos métodos no Grupo 1 – Fonte: Elaborado pelos autores

Ainda pode ser observado na Tabela 5, que quando se fala dos resultados globais referentes ao Grupo 2, algumas alterações em relação à melhor regra são percebidas. Para os 3 menores valores de alfa, ou seja, alfa igual a 0,0; 0,25 e 0,50, a regra R_5 apresentou os melhores resultados. Já para os valores de alfa iguais a 0,75 e 1,0, a regra R_2 apresenta os resultados mais satisfatórios, conforme Figura 14, referente ao Grupo 2.

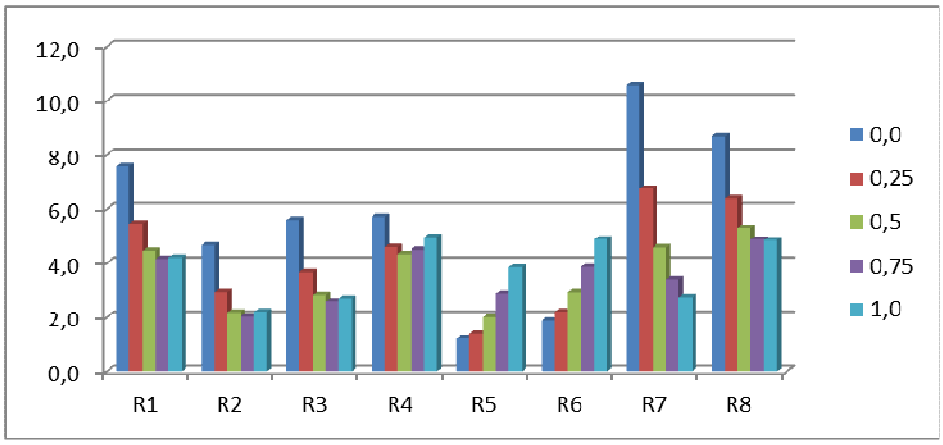

Figura 14 – Comparação dos resultados globais do RPD dos métodos no Grupo 2 – Fonte: Elaborado pelos autores

Após essa análise percebeu-se que as regras R_2 e R_5 obtiveram juntas quase a totalidade de melhores resultados. Além disso, as oito regras, todas forneceram melhores resultados que a R_8, a regra aleatória.

Foram elaborados gráficos com resultados parciais do desempenho das tarefas que melhor se sobressaíram. Para cada valor de alfa, avaliaram-se as três regras que mais se destacaram, ou seja, as regras que obtiveram os menores valores de RPD. A seguir, as análises mais detalhadas deste estudo para melhor entendimento.

4.2.1 Análise dos resultados para $\alpha = 0$

Para alfa igual a zero, a regra R_5 forneceu os menores valores, e consequentemente as melhores porcentagens de desvio relativo em todos os parâmetros referentes ao Grupo 1. A medida que o número de tarefas e intervalos de data de liberação aumentavam, os valores do RPD também aumentavam. Já no que diz respeito ao aumento do número de máquinas e aos intervalos *setup*, o efeito foi o contrário, ou seja, a medida que aumentavam, os valores do RPD iam diminuindo.

No Grupo 2, em geral a regra R_5 também mostrou um bom desempenho, com algumas exceções. Com o aumento do número de tarefas, a regra R_6 vai melhorando os resultados, até ultrapassar a regra R_5 em problemas com 80 e 100 tarefas. Na análise dos problemas por número de máquinas e dos intervalos de datas de liberação e de *setup*, embora haja variação nos resultados dos métodos nos diferentes valores destes parâmetros,

em todos os casos a ordem de superioridade se manteve com a regra R_5 em primeiro lugar, a R_6 em segundo e a R_2 em terceiro, conforme demostrado na Figura 15.

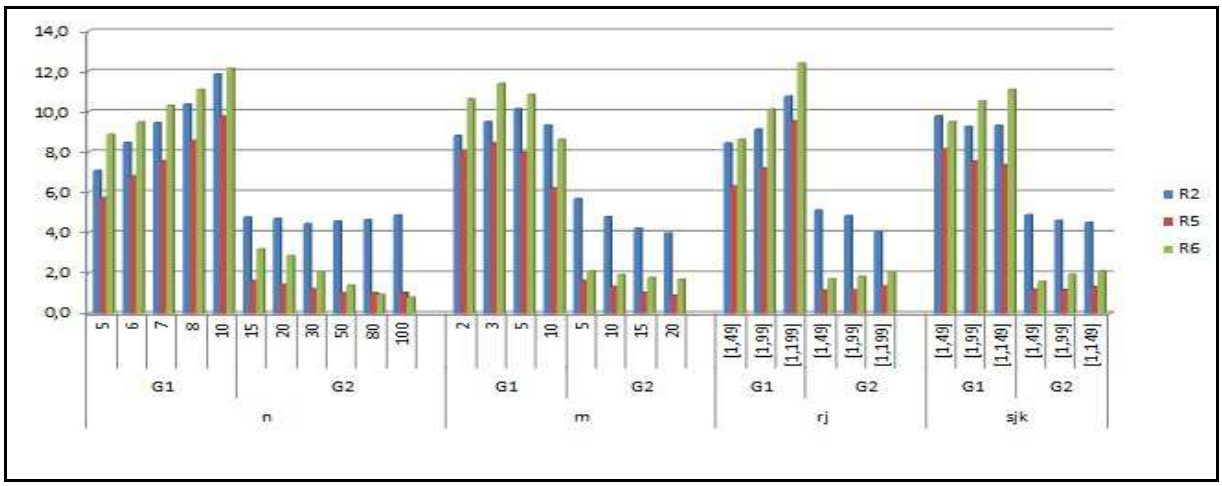

Figura 15 – Valores do RPD para $\alpha = 0,0$ – Fonte: Elaborado pelos autores.

4.2.2 Análise dos resultados para $\alpha = 0,25$

Neste caso, a regra R_2 possui a grande maioria dos melhores desempenhos, enquanto a regra R_5 obteve melhor desempenho em apenas dois casos: quando o número de máquinas é igual a 10 e no maior intervalo de tempos de *setup* [1,149]. Isso mostra que à medida que o número de máquinas ou intervalo de *setup* for aumentado, a regra R_5 apresenta os melhores resultados no Grupo 1. Ainda neste grupo, quanto maior o número de tarefas, maior o valor do RPD, e isso se repete nos intervalos de data de liberação, ou seja, a medida que aumentamos o intervalo, os valores também aumentam.

Com alfa igual a 0,25 para as tarefas do Grupo 2, a regra R_5 conseguiu as melhores porcentagens de desvio relativo e obteve os melhores resultados em todos os parâmetros. Os valores do RPD decrescem a medida que o número de tarefas, máquinas e intervalos de *setup* vão aumentando. Já nos intervalos de liberação, a medida que são maiores, os valores do RPD também aumentam. Tais afirmações estão demostradas graficamente na Figura 16.

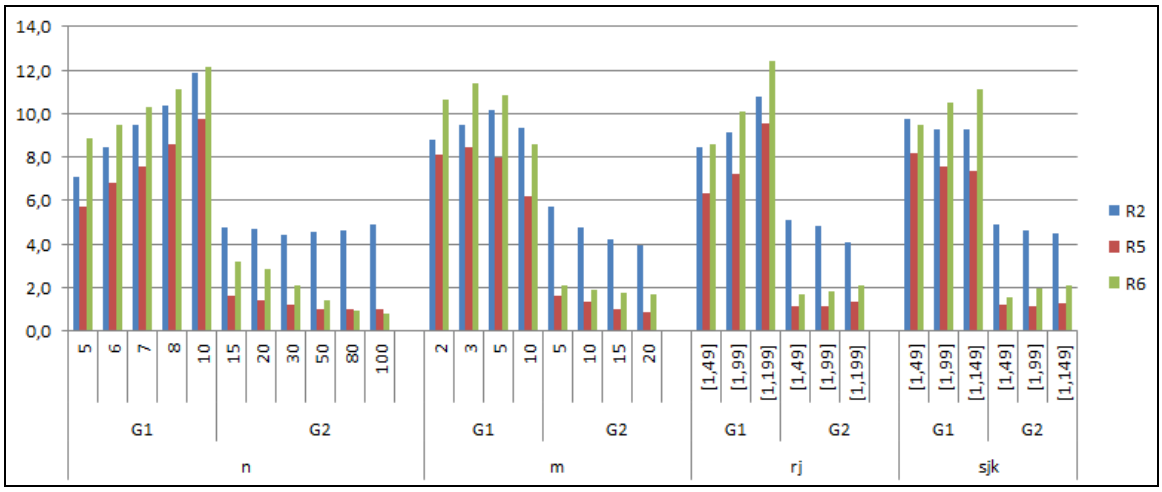

Figura 16 – Valores do RPD para α = 0,25 – Fonte: Elaborado pelos autores.

4.2.3 Análise dos resultados para α = 0,5

Quando se aumenta o valor de alfa para 0,5, o desempenho da regra R_2 é o melhor em todos os RPD do Grupo 1. Os valores do RPD são crescentes à medida que aumentam o número de tarefas, o número de máquinas e os intervalos de data de liberação. Nos intervalos de *setup*, ocorre o contrário, a medida que são maiores, os valores do RPD vão diminuindo.

No Grupo 2, com a regra R_5 se obteve os melhores resultados, mas a regra R_2 conseguiu se sobressair nos seguintes critérios: quando o número de tarefas é igual a 15 e 20; quando o número de máquinas é o menor (m = 5); quando o intervalo das datas de liberação é o maior [1,199] e quando o intervalo dos tempos de *setup* é o menor [1,49]. Apenas nesses 3 casos a R_2 foi superior. À medida que o numero de tarefas e máquinas vão aumentando, os valores do RPD diminuem, conforme Figura 17.

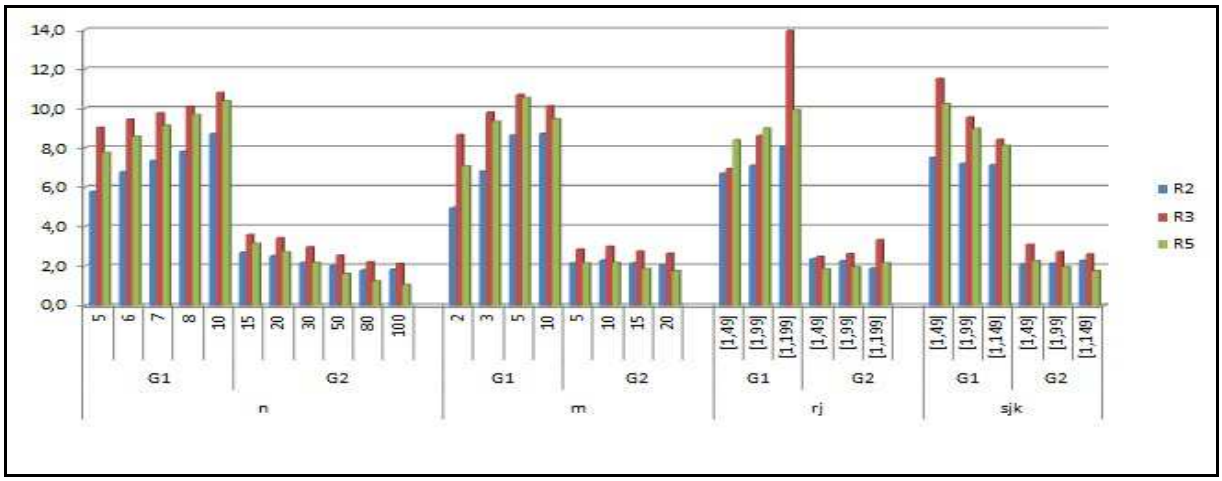

Figura 17 – Valores do RPD para $\alpha = 0,5$ – Fonte: Elaborado pelos autores.

4.2.4 Análise dos resultados para $\alpha = 0,75$

Para alfa igual a 0,75, a regra R_2 teve o melhor desempenho. Com esta regra conseguiu-se os melhores resultados perante as demais regras e venceu em todas as opções do número de tarefas ou máquinas e também em intervalos de data de liberação ou *setup* em ambos os grupos. Percebe-se no Grupo 1 que, a medida que os números de tarefas, máquinas e intervalo de data de liberação vão aumentando, os valores do RPD também vão aumentando. No caso dos intervalos de *setup*, ocorre o contrário, os valores do RPD diminuem à medida que o intervalo vai aumentando.

Já no Grupo 2, quando o número de máquinas e o intervalo de data de liberação vão aumentando, os valores vão decrescendo. Em relação ao intervalo de *setup*, quanto maior o intervalo, menor é o valor do RPD correspondente àquele intervalo. A Figura 18 ilustra graficamente a análise dos resultados para alfa igual a 0,75.

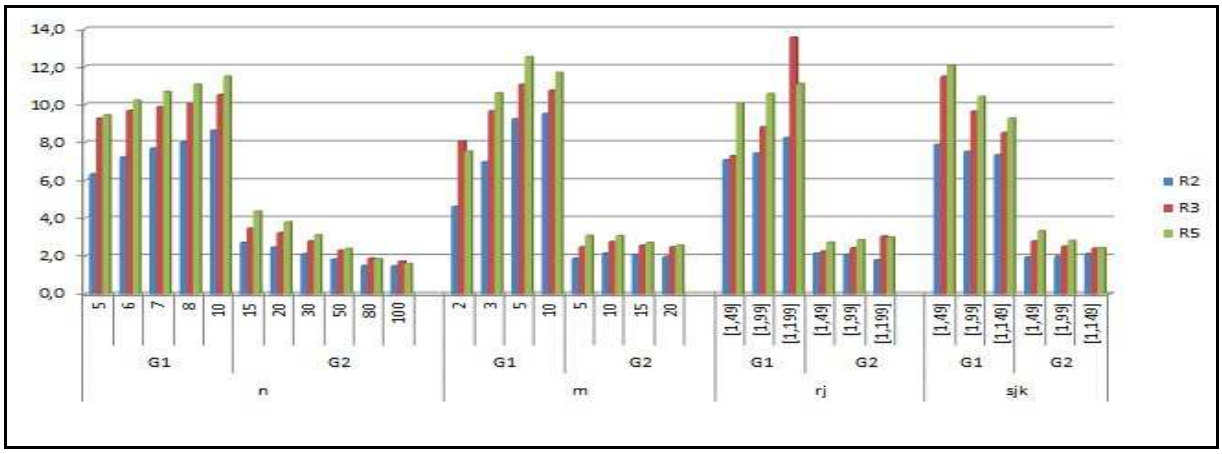

Figura 18 – Valores do RPD para $\alpha = 0,75$ – Fonte: Elaborado pelos autores.

4.2.5 Análise dos resultados para $\alpha = 1$

Quando aumentamos o valor de alfa, ou seja, com o valor igual a 1,0, a situação em relação à anterior (alfa = 0,75) não se difere, ou seja, a regra R_2 permanece com os melhores resultados, conforme é demostrado na Figura 19. Nota-se no Grupo 1 que, a medida que os números de tarefas, máquinas e intervalo de data de liberação vão aumentando, os valores do RPD também crescem. No caso dos intervalos de *setup*, os valores do RPD diminuem à medida que o intervalo vai aumentando.

Já no Grupo 2, quando o número de máquinas e o intervalo de data de liberação vão aumentando, os valores vão decrescendo. Em relação ao intervalo de *setup*, quanto maior o intervalo, maior é o valor do RPD correspondente àquele intervalo.

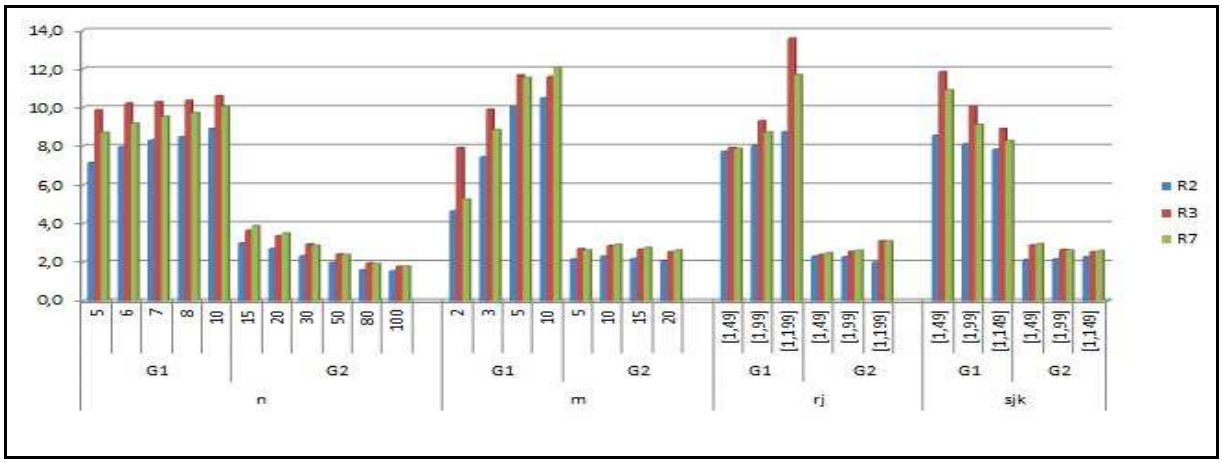

Figura 19 – Valores do RPD para $\alpha = 1$ – Fonte: Elaborado pelos autores.

A seguir é apresentada a Tabela 6 que ilustra numericamente as três regras que apresentaram o melhor desempenho e os seus respectivos valores do RPD, para cada opção dos valores de alfa. Consta apenas uma única exceção: para alfa igual 0,5, no Grupo 2, em problemas com 100 tarefas, a regra que ficou em segundo lugar foi a R_4, com RPD de 1,6, que não consta na tabela. Portanto, neste caso, a ordem de superioridade foi: R_5, R_4, R_2 e R_3. Para cada opção de parâmetro, a regra que obteve o melhor resultado está identificada em azul e a que ficou em segundo lugar em verde.

			Alfa = 0			Alfa = 0,25			Alfa = 0,5			Alfa = 0,75			Alfa = 1		
			R2	R5	R6	R2	R5	R6	R2	R3	R5	R2	R3	R5	R2	R3	R7
G1	n	5	7,1	5,7	8,9	5,8	6,4	9,1	5,8	9,0	7,8	6,3	9,3	9,5	7,2	9,9	8,7
		6	8,5	6,8	9,5	6,9	7,2	9,7	6,8	9,5	8,6	7,2	9,7	10,2	8,0	10,2	9,2
		7	9,5	7,6	10,3	7,6	7,9	10,4	7,4	9,8	9,1	7,7	9,9	10,7	8,3	10,3	9,5
		8	10,4	8,6	11,1	8,3	8,6	11,0	7,8	10,1	9,7	8,0	10,1	11,1	8,5	10,4	9,7
		10	11,9	9,8	12,1	9,4	9,5	11,9	8,7	10,8	10,4	8,7	10,5	11,5	8,9	10,6	10,1
G2		15	4,8	1,6	3,2	3,2	2,2	3,5	2,7	3,6	3,2	2,7	3,5	4,4	3,0	3,7	3,9
		20	4,7	1,4	2,8	3,1	1,9	3,2	2,5	3,4	2,7	2,5	3,2	3,8	2,7	3,4	3,5
		30	4,5	1,2	2,1	2,8	1,5	2,4	2,2	3,0	2,2	2,1	2,8	3,1	2,3	3,0	2,9
		50	4,6	1,0	1,4	2,8	1,1	1,8	2,0	2,6	1,6	1,8	2,3	2,4	2,0	2,4	2,4
		80	4,7	1,0	1,0	2,7	0,9	1,3	1,8	2,2	1,2	1,5	1,9	1,9	1,6	2,0	1,9
		100	4,9	1,0	0,8	2,8	0,8	1,1	1,8	2,1	1,1	1,5	1,7	1,6	1,6	1,8	1,8
G1	m	2	8,8	8,1	10,6	6,0	7,1	9,7	5,0	8,7	7,1	4,6	8,1	7,5	4,7	7,9	5,3
		3	9,5	8,4	11,4	7,3	8,4	11,2	6,8	9,8	9,3	7,0	9,7	10,6	7,4	9,9	8,9
		5	10,1	8,0	10,9	8,6	8,8	11,4	8,6	10,7	10,6	9,2	11,1	12,5	10,1	11,7	11,6
		10	9,3	6,2	8,6	8,4	7,4	9,4	8,7	10,1	9,5	9,5	10,8	11,7	10,5	11,6	12,1
G2		5	5,7	1,6	2,1	3,2	1,6	2,7	2,2	2,9	2,2	1,9	2,5	3,1	2,2	2,7	2,6
		10	4,8	1,3	1,9	3,0	1,5	2,3	2,3	3,0	2,2	2,2	2,8	3,1	2,3	2,9	2,9
		15	4,2	1,0	1,8	2,8	1,3	2,0	2,2	2,8	1,9	2,1	2,6	2,7	2,2	2,7	2,8
		20	4,0	0,9	1,7	2,7	1,2	1,9	2,1	2,7	1,8	2,0	2,5	2,6	2,1	2,5	2,6
G1	rj	[1,49]	8,4	6,3	8,6	6,9	6,9	9,1	6,7	6,9	8,4	7,1	7,3	10,1	7,7	7,9	7,9
		[1,99]	9,1	7,2	10,1	7,4	7,7	10,3	7,1	8,6	9,0	7,4	8,8	10,6	8,0	9,3	8,7
		[1,199]	10,8	9,5	12,4	8,5	9,1	11,8	8,0	14,0	9,9	8,2	13,5	11,1	8,7	13,6	11,7
G2		[1,49]	5,1	1,1	1,7	3,2	1,3	2,0	2,4	2,5	1,9	2,2	2,3	2,7	2,3	2,4	2,5
		[1,99]	4,8	1,2	1,8	3,0	1,3	2,2	2,3	2,6	2,0	2,1	2,4	2,9	2,3	2,6	2,6
		[1,199]	4,0	1,4	2,1	2,5	1,6	2,4	1,9	3,3	2,2	1,8	3,1	3,0	2,0	3,1	3,1
G1	sjk	[1,49]	9,8	8,1	9,5	7,8	8,7	9,9	7,5	11,5	10,2	7,9	11,5	12,1	8,6	11,9	10,9
		[1,99]	9,3	7,5	10,5	7,5	7,8	10,5	7,2	9,6	9,0	7,5	9,6	10,4	8,1	10,1	9,1
		[1,149]	9,3	7,4	11,1	7,5	7,3	10,9	7,1	8,4	8,1	7,3	8,5	9,3	7,8	8,9	8,3
G2		[1,49]	4,9	1,2	1,6	2,9	1,5	1,9	2,1	3,1	2,3	1,9	2,8	3,3	2,1	2,9	3,0
		[1,99]	4,6	1,2	1,9	2,9	1,4	2,2	2,2	2,7	2,0	2,0	2,5	2,8	2,2	2,6	2,6
		[1,149]	4,5	1,3	2,1	2,9	1,3	2,5	2,3	2,6	1,8	2,1	2,4	2,4	2,3	2,5	2,6

Tabela 6 – Valores do RPD detalhado por parâmetros – Fonte: Elaborado pelos autores.

4.2.6 Tempo computacional das Regras de Prioridade

No que diz respeito aos tempos computacionais, foram elaboradas tabelas com os tempos de CPU de cada regra na experimentação computacional. A seguir, os tempos de execução dos problemas do Grupo 1.

Grupo 1							
R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
0,078	0,078	0,031	0,030	0,030	0,064	0,078	0,109

Tabela 7 – Tempos Totais de CPU do Grupo 1 para cada regra (em segundos) - Fonte: Elaborado pelos autores.

Como os tempos de CPU para a resolução dos problemas pelas Regras de Prioridade foram irrelevantes, os valores da Tabela 7 indicam o tempo total de CPU que uma regra levou para fornecer a solução de todos os problemas do grupo nas cinco opções de alfa. Já o tempo consumido para encontrar a solução ótima pelo processo de enumeração completa foi consideravelmente superior, conforme previsto. Assim, a Tabela 8 apresenta os tempos (em horas) para cada opção de alfa e a totalização para os cinco alfas.

Enumeração Completa					
$\alpha = 0$	$\alpha = 0,25$	$\alpha = 0,5$	$\alpha = 0,75$	$\alpha = 1$	Tempo Total
8,317	8,316	8,315	8,315	8,316	41,579

Tabela 8 – Tempos de CPU para Enumeração Completa (em horas) – Fonte: Elaborado pelos autores.

Na resolução do Grupo 2, naturalmente os problemas levaram mais tempo de CPU do que os do Grupo 1, por se tratarem de problemas de médio e grande portes. Entretanto, ainda assim os tempos computacionais são bastante aceitáveis; cada regra consumiu em torno de 1 segundo para resolver todos os problemas do grupo nas cinco opções de alfa, como mostra a Tabela 9.

Grupo 2							
R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
1,205	1,113	0,876	1,098	0,995	1,024	1,048	1,081

Tabela 9 – Tempos Totais de CPU do Grupo 2 para cada regra (em segundos) – Fonte: Elaborado pelos autores.

4.3 Análise dos resultados das Heurísticas Construtivas

Os resultados globais da implementação computacional das heurísticas propostas mostraram que a H_1, a H_2 e a H_3 obtiveram desempenhos muito similares e, além disso, muito próximos da solução ótima. Conforme pode ser observado na Tabela 10, estas três heurísticas ficaram levemente acima de apenas 1% de diferença da solução ótima. Este é um resultado bastante expressivo para heurísticas que consumiram tempo computacional viável.

A Tabela 10, também expressa numericamente os dados globais, destacando os melhores e os piores resultados. Destacados na cor azul estão os melhores desempenhos e de verde os segundos melhores. Em vermelho estão os piores resultados.

Embora o resultado das heurísticas H_1, H_2 e H_3 tenham ficado bastante próximos, a heurística H_1 obteve o melhor desempenho nas cinco opções de alfa do Grupo 1 e em três opções de alfa do Grupo 2 (revezando apenas com a heurística H3, com valores de alfa 0,0 e 0,25).

Dentre as quatro propostas, a heurística H_4 atingiu os piores resultados em termos de RPD, com desempenhos variáveis em cada uma das opções de alfa dos dois grupos. Esta variabilidade não se observou nas três primeiras heurísticas, ou seja, as três melhores possuem também resultados estáveis para as opções de alfa e de grupo.

	Alfa	H1	H2	H3	H4
GRUPO 1	0,0	1,1	1,3	1,2	19,7
	0,25	1,2	1,3	1,3	13,1
	0,5	1,4	1,5	1,5	9,6
	0,75	1,5	1,6	1,6	7,4
	1,0	1,5	1,7	1,6	6,0
GRUPO 2	0,0	0,7	1,0	0,5	15,0
	0,25	0,6	0,8	0,5	10,9
	0,5	0,6	0,8	0,6	8,7
	0,75	0,6	0,9	0,7	7,3
	1,0	0,6	0,9	0,8	6,4

Tabela 10 – Resultados da Análise Global do RPD das Heurísticas - Fonte: Elaborado pelos autores.

Para melhor visualização dos resultados globais, os gráficos das Figuras 20 e 21 apresentam os RPD do Grupo 1 e do Grupo 2 separadamente, para cada opção de alfa.

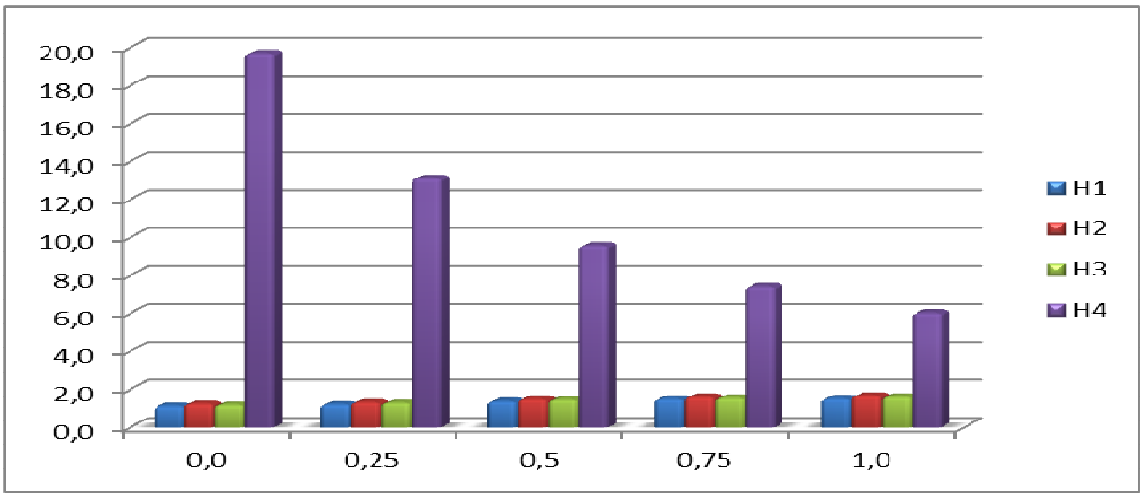

Figura 20 – Comparação dos resultados globais do RPD das heurísticas no Grupo 1 – Fonte: Elaborado pelos autores.

Conforme já descrito e que também pode ser visto na Figura 20, no Grupo 1 os desempenhos das heurísticas H_1, H_2 e H_3 são bastante próximos e com valores do RPD levemente crescentes (ou seja, pioram) como o aumento do valor do alfa. Além disso, é possível visualizar melhor a discrepância dos resultados da heurística H_4 em relação às demais. É possível verificar também na H_4, opostamente às três primeiras heurísticas, a tendência decrescente dos valores do RPD (indicando melhoria) com o aumento do valor do alfa. Ou seja, a heurística H_4 melhora seu desempenho quanto maior o peso do *makespan* na função objetivo.

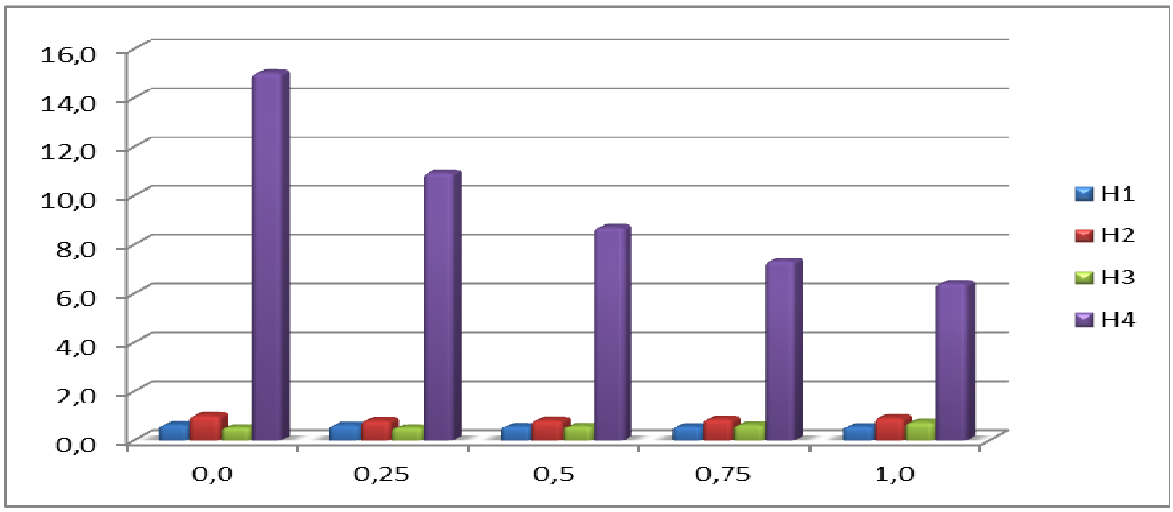

Figura 21 – Comparação dos resultados globais do RPD das heurísticas no Grupo 2 – Fonte: Elaborado pelos autores.

A Figura 21, com os resultados do Grupo 2, mostra também a discrepância dos resultados da heurística H_4 em relação às demais e a tendência decrescente (de melhoria)

nos valores do RPD da heurística H_4 com o aumento do valor de alfa. Por outro lado, no Grupo 2, os resultados das heurísticas H_1, H_2 e H_3 ficaram mais próximos entre si e sem tendência de crescimento, indicando maior estabilidade do que no Grupo 1. Visando uma maior exploração dos resultados das heurísticas propostas, a seguir serão apresentadas análises detalhadas para cada opção de alfa e para os demais parâmetros considerados na experimentação. Os resultados da heurística H_4 foram elaborados em gráficos diferentes das demais heurísticas (H_1, H_2 e H_3), visto que analisados separadamente é possível uma melhor visualização do problema já que os resultados são valores maiores e muito diferentes dos demais.

4.3.1 Análise dos resultados das heurísticas para $\alpha = 0$

Os gráficos das Figuras 22 e 23 apresentam as análises para alfa igual a zero detalhada por parâmetros (número de tarefas, número de máquinas, intervalos de datas de liberação e de tempos de *setup*) e separadamente para o Grupo 1 e para o Grupo 2.

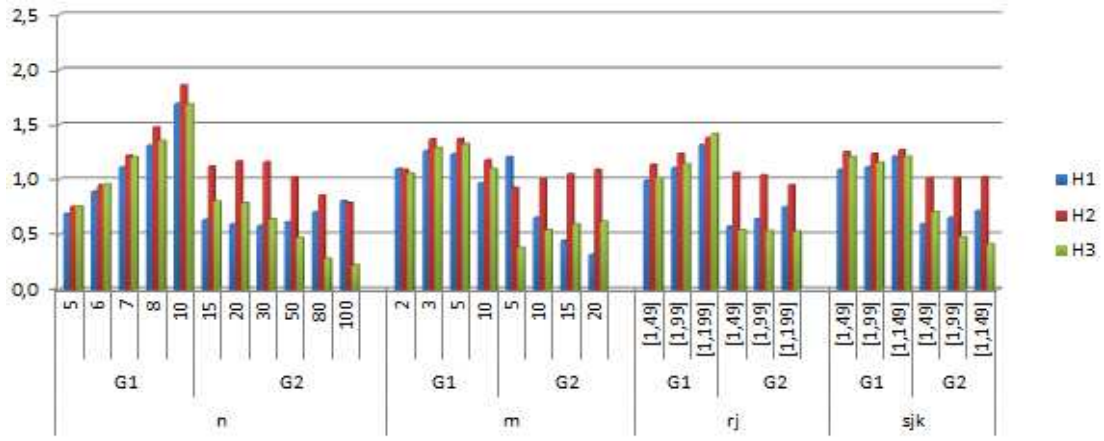

Figura 22 – Valores do RPD das heurísticas H_1, H_2 e H_3 para $\alpha = 0$ – Fonte: Elaborado pelos autores.

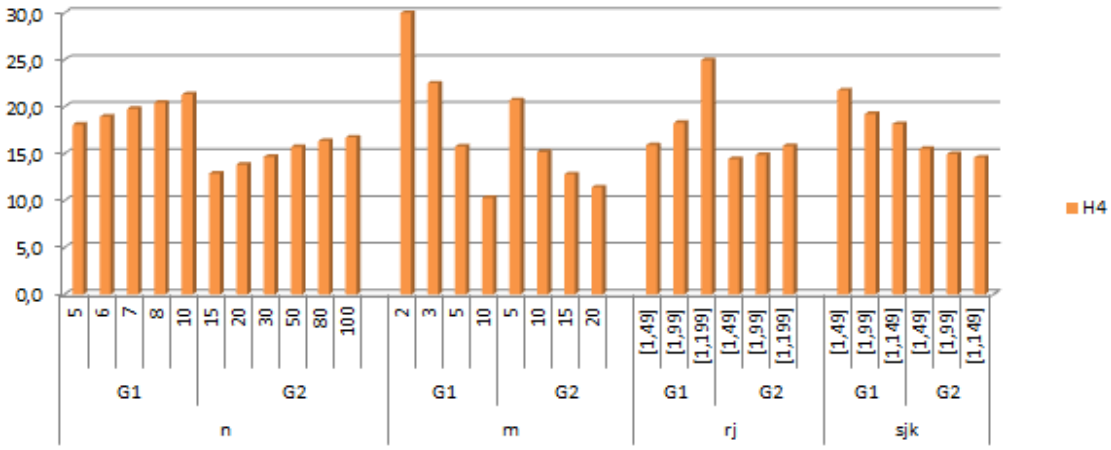

Figura 23 – Valores do RPD da heurística H_4 para $\alpha = 0$ – Fonte: Elaborado pelos autores.

Nesta opção de alfa igual a zero, que é o caso em que o problema se torna monocritério com minimização apenas do tempo médio de fluxo, as heurísticas H_1 e H_3 obtiveram os menores valores do RPD em todos os parâmetros, tanto no Grupo 1 como no Grupo 2, com diferenças muito pequenas. Também com pouca diferença no desempenho, a Heurística H_2 ficou sempre em terceiro lugar.

Em geral, os resultados para os diferentes parâmetros tiveram comportamentos similares. A maior variação foi observada na diferença dos resultados dos Grupos 1 e 2, principalmente com a heurística H_4. Na maioria dos casos, os desvios do Grupo 1 ficaram levemente superiores aos do Grupo 2.

Além disso, no Grupo 1, houve um crescimento nos desvios das quatro heurísticas com o aumento no número de tarefas, o que não é claramente identificado no Grupo 2, especificamente nas três primeiras heurísticas. As tabelas com os valores utilizados na construção dos gráficos deste capítulo encontram-se no Apêndice.

4.3.2 Análise dos resultados das heurísticas para $\alpha = 0,25$

A primeira opção de alfa que torna o problema de fato bicritério é com $\alpha = 0,25$, ainda assim conferindo maior peso ao tempo médio de fluxo do que ao *makespan*. As Figuras 24 e 25 mostram os resultados detalhados para esta opção de alfa.

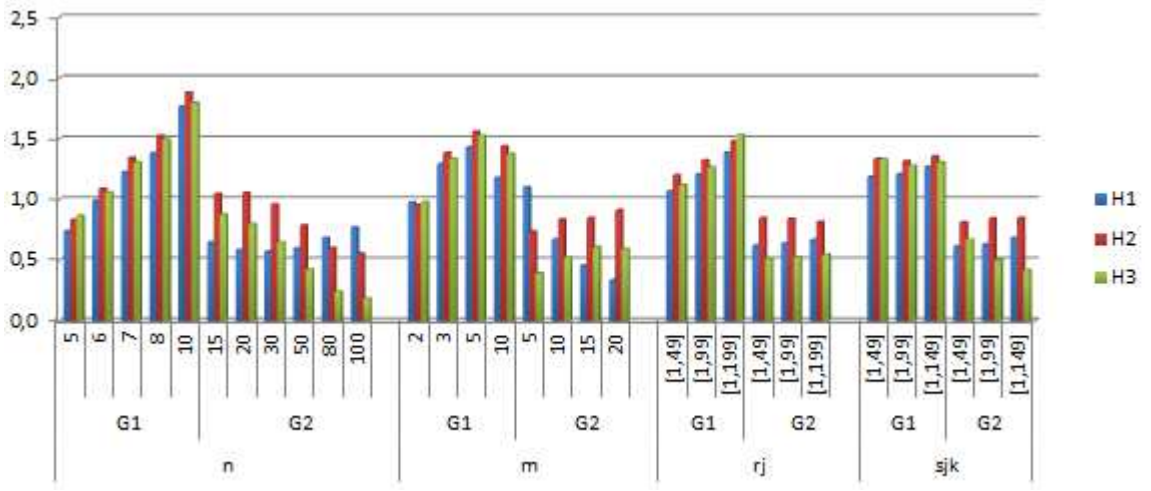

Figura 24 – Valores do RPD das heurísticas H_1, H_2 e H_3 para $\alpha = 0,25$ – Fonte: Elaborado pelos autores.

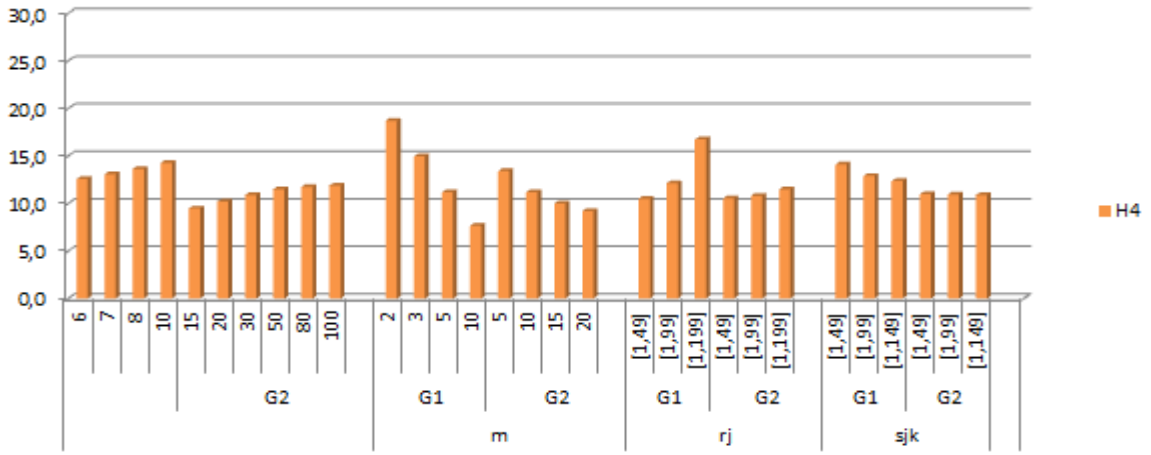

Figura 25 – Valores do RPD da heurística H_4 para $\alpha = 0,25$ – Fonte: Elaborado pelos autores.

Pode-se notar que os gráficos de alfa igual a zero e igual a 0,25 são bastante parecidos, com as mesmas tendências. Novamente, as heurísticas H_1, H_3 e H_2 tiveram os melhores resultados, nesta ordem de superioridade, e com diferenças muito pequenas entre si e com grande número de empates, tanto no Grupo 1 como no Grupo 2.

4.3.3 Análise dos resultados das heurísticas para $\alpha = 0,5$

O problema bicritério com pesos iguais para *makespan* em tempo médio de fluxo se dá com alfa igual a 0,5, e os resultados são apresentados no gráfico das Figuras 26 e 27 logo a seguir.

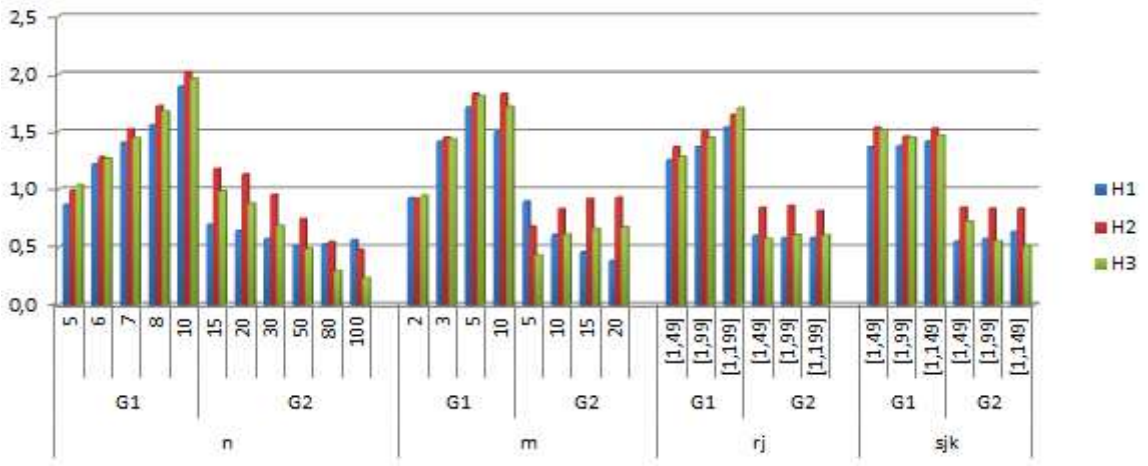

Figura 26 – Valores do RPD das heurísticas H_1, H_2 e H_3 para $\alpha = 0,5$ – Fonte: Elaborado pelos autores.

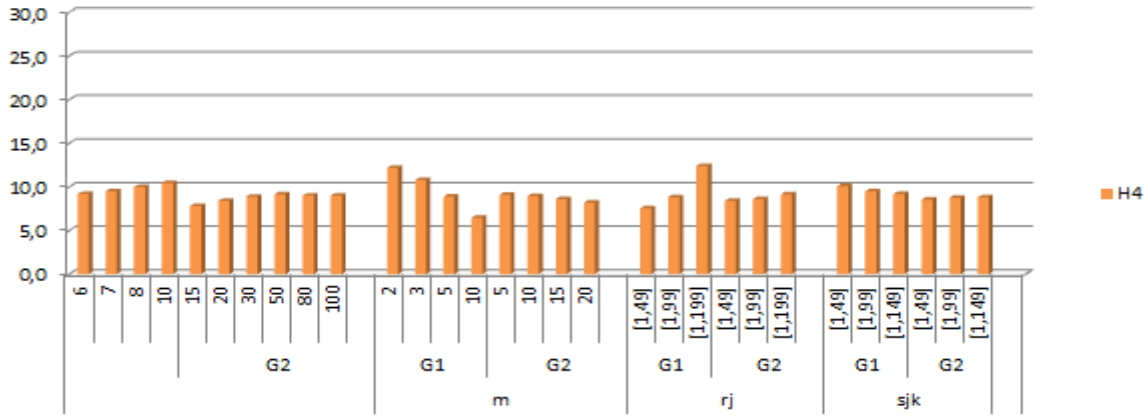

Figura 27 – Valores do RPD da heurística H_4 para $\alpha = 0,5$ – Fonte: Elaborado pelos autores.

Novamente, os resultados da opção de alfa igual a 0,5 mostram as mesmas tendências nos gráficos das Figuras 26 e 27, tanto em relação à ordem de superioridade das heurísticas (H_1, H_3, H_2 e H_4), como no desempenho dos dois grupos e nos vários parâmetros do problema.

Aqui se confirma a tendência da heurística H_4 de reduzir os valores dos seus desvios com o aumento do valor do alfa. Ou seja, ela melhora o seu desempenho quanto maior for o peso do *makespan* na função objetivo.

4.3.4 Análise dos resultados para $\alpha = 0,75$

A última opção de alfa para o problema verdadeiramente bicritério é com $\alpha = 0,75$, que atribui desta vez maior peso ao *makespan* em relação ao tempo médio de fluxo. Os resultados detalhados são mostrados nas Figuras 27 e 28.

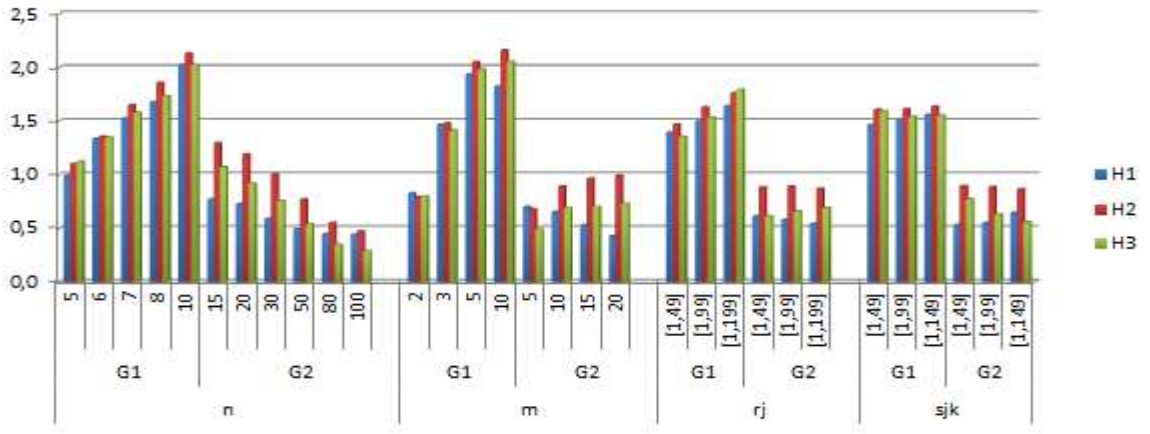

Figura 28 – Valores do RPD das heurísticas H_1, H_2 e H_3 para $\alpha = 0,75$ – Fonte: Elaborado pelos autores.

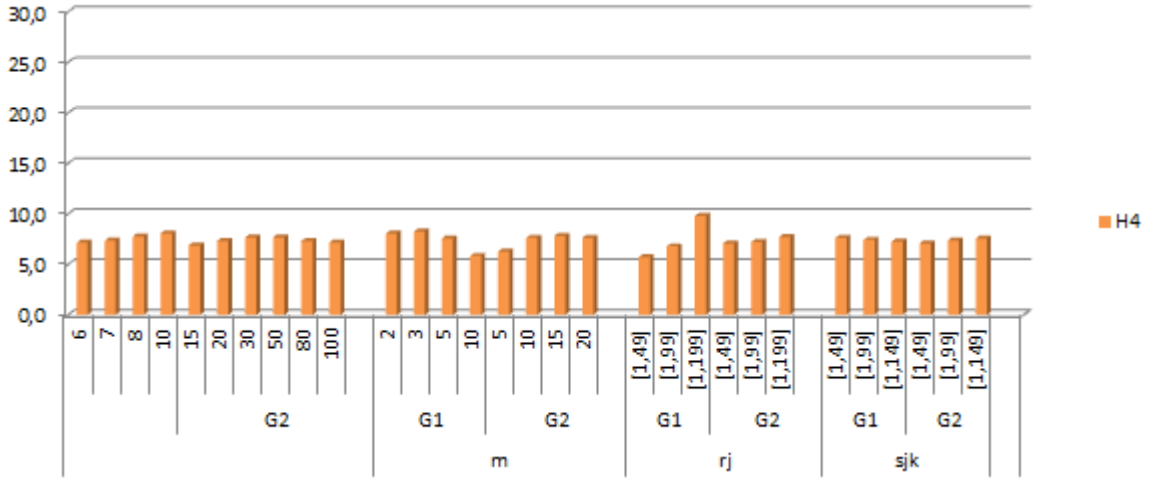

Figura 29 – Valores do RPD da heurística H_4 para $\alpha = 0,75$ – Fonte: Elaborado pelos autores.

Aqui todas as tendências apontadas anteriormente se confirmam, tanto em relação ao desempenho das heurísticas nos dois grupos e nos diferentes parâmetros, como também a redução do RPD da heurística H_4 em relação às outras opções de alfa.

Portanto, é possível afirmar que o valor de alfa praticamente não influencia no desempenho das melhores heurísticas H_1, H_2 e H_3, enquanto que influencia fortemente nos resultados da heurística H_4.

4.3.5 Análise dos resultados para $\alpha = 1$

Por fim, a última opção de alfa ($\alpha = 1$) é aquela que torna novamente o problema monocritério, porém agora apenas com a minimização do *makespan* como medida de desempenho (e tempo médio de fluxo com peso zero). É importante salientar que embora o *makespan* para *flow shop* seja uma medida clássica e já muito explorada na literatura, a contribuição desta parte específica desta pesquisa reside no fato de se ter considerado no problema restrições raramente encontradas conjuntamente, que é a presença de datas de liberação diferentes de zero e tempos de *setup* independentes da sequência. As Figuras 30 e 31 detalham os resultados para a opção de alfa igual a 1.

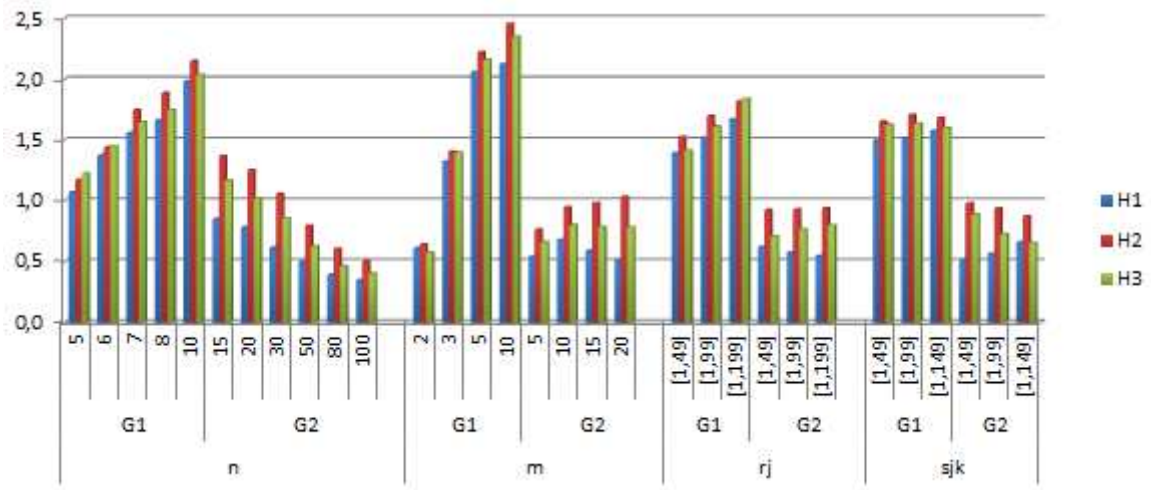

Figura 30 – Valores do RPD das heurísticas H_1, H_2 e H_3 para $\alpha = 1$ – Fonte: Elaborado pelos autores.

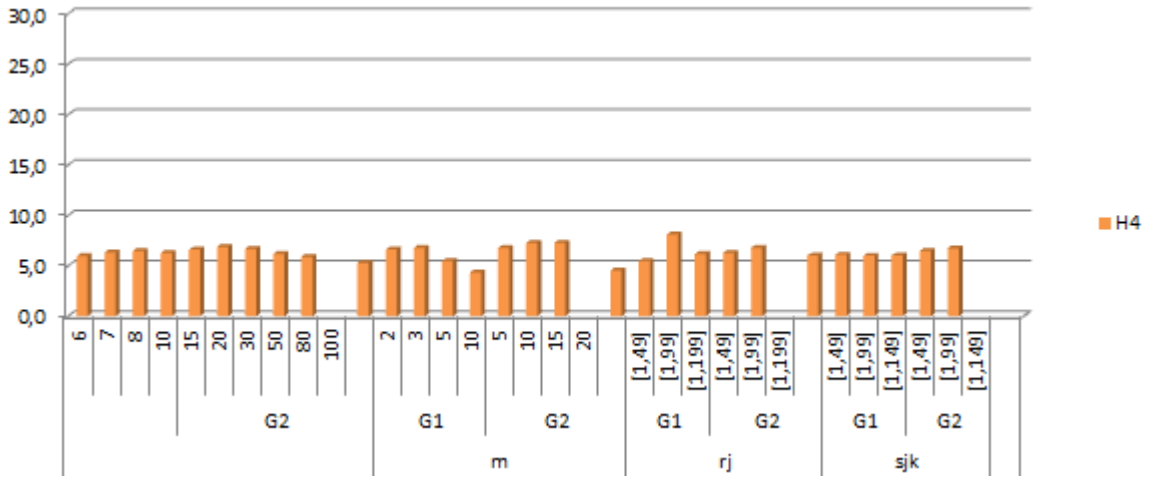

Figura 31 – Valores do RPD da heurística H_4 para $\alpha = 1$ – Fonte: Elaborado pelos autores.

Ao se acrescentar a análise do gráfico da Figura 26 às análises efetuadas anteriormente, as observações apontadas tornam-se conclusivas, pois demonstram a consistência das tendências mencionadas. Conforme enfatizado, a ordem de superioridade no desempenho das heurísticas se manteve em H_1, H_3, H_2 e H_4, com as três primeiras tendo diferenças muito pequenas e muitos empates.

Além disso, não houve variação significativa nos resultados das heurísticas construtivas quando se alterava o valor de um parâmetro do problema; a maior delas foi no Grupo 1, em que o aumento do número de tarefas provocou o crescimento do RPD das quatro heurísticas. Portanto, infere-se que para este problema, os valores dos parâmetros como número de tarefas, número de máquinas, intervalos de datas de liberação e de tempos de *setup* pouco influenciam no resultado dos métodos.

A seguir é apresentada a Tabela 11 que ilustra numericamente as quatro heurísticas construtivas propostas e os seus respectivos valores do RPD, para cada opção de alfa. Esta análise visa detectar a existência de resultados diferentes dos analisados anteriormente para alguma classe específica de problemas e que não se possa perceber em resultados baseados em média.

Em cada classe de problemas, o resultado da melhor heurística está ilustrado de azul, a que ficou em segundo lugar, de verde. É importante salientar que os desempates foram decididos pela próxima casa decimal. Na prática, estas diferenças não têm representatividade alguma.

			Alfa = 0				Alfa = 0,25				Alfa = 0,5				Alfa = 0,75				Alfa = 1			
			H1	H2	H3	H4	H1	H2	H3	H4	H1	H2	H3	H4	H1	H2	H3	H4	H1	H2	H3	H4
G1	n	5	0,7	0,8	0,8	18,1	0,7	0,8	0,9	12,1	0,9	1,0	1,0	8,8	1,0	1,1	1,1	6,8	1,1	1,2	1,2	5,6
		6	0,9	1,0	1,0	18,9	1,0	1,1	1,1	12,6	1,2	1,3	1,3	9,2	1,3	1,4	1,3	7,2	1,4	1,4	1,4	5,8
		7	1,1	1,2	1,2	19,8	1,2	1,3	1,3	13,1	1,4	1,5	1,4	9,5	1,5	1,7	1,6	7,4	1,6	1,7	1,6	6,0
		8	1,3	1,5	1,4	20,4	1,4	1,5	1,5	13,6	1,6	1,7	1,7	10,0	1,7	1,9	1,7	7,8	1,7	1,9	1,7	6,3
		10	1,7	1,9	1,7	21,3	1,8	1,9	1,8	14,3	1,9	2,0	2,0	10,5	2,0	2,1	2,0	8,1	2,0	2,2	2,0	6,5
G2		15	0,6	1,1	0,8	12,9	0,6	1,0	0,9	9,5	0,7	1,2	1,0	7,8	0,8	1,3	1,1	6,9	0,9	1,4	1,2	6,3
		20	0,6	1,2	0,8	13,8	0,6	1,1	0,8	10,2	0,6	1,1	0,9	8,4	0,7	1,2	0,9	7,3	0,8	1,3	1,0	6,6
		30	0,6	1,2	0,6	14,7	0,6	1,0	0,6	10,9	0,6	1,0	0,7	8,9	0,6	1,0	0,8	7,7	0,6	1,1	0,9	6,9
		50	0,6	1,0	0,5	15,7	0,6	0,8	0,4	11,5	0,5	0,7	0,5	9,2	0,5	0,8	0,5	7,7	0,5	0,8	0,6	6,7
		80	0,7	0,9	0,3	16,4	0,7	0,6	0,2	11,7	0,5	0,5	0,3	9,0	0,4	0,6	0,3	7,3	0,4	0,6	0,5	6,2
		100	0,8	0,8	0,2	16,7	0,8	0,6	0,2	11,9	0,6	0,5	0,2	9,0	0,4	0,5	0,3	7,2	0,3	0,5	0,4	5,9
G1	m	2	1,1	1,1	1,1	30,3	1,0	1,0	1,0	18,7	0,9	0,9	0,9	12,2	0,8	0,8	0,8	8,1	0,6	0,6	0,6	5,3
		3	1,3	1,4	1,3	22,5	1,3	1,4	1,3	15,0	1,4	1,5	1,4	10,8	1,5	1,5	1,4	8,3	1,3	1,4	1,4	6,6
		5	1,2	1,4	1,3	15,8	1,4	1,6	1,5	11,2	1,7	1,8	1,8	8,9	1,9	2,0	2,0	7,6	2,1	2,2	2,2	6,8
		10	1,0	1,2	1,1	10,3	1,2	1,4	1,4	7,7	1,5	1,8	1,7	6,5	1,8	2,2	2,1	5,8	2,1	2,5	2,4	5,5
G2		5	1,2	0,9	0,4	20,7	1,1	0,7	0,4	13,4	0,9	0,7	0,4	9,1	0,7	0,7	0,5	6,3	0,5	0,8	0,7	4,3
		10	0,7	1,0	0,5	15,2	0,7	0,8	0,5	11,2	0,6	0,8	0,6	9,0	0,7	0,9	0,7	7,6	0,7	0,9	0,8	6,8
		15	0,4	1,1	0,6	12,8	0,5	0,8	0,6	10,0	0,5	0,9	0,7	8,6	0,5	1,0	0,7	7,8	0,6	1,0	0,8	7,3
		20	0,3	1,1	0,6	11,4	0,3	0,9	0,6	9,2	0,4	0,9	0,7	8,2	0,4	1,0	0,7	7,6	0,5	1,0	0,8	7,3
G1	rj	[1,49]	1,0	1,1	1,0	15,9	1,1	1,2	1,1	10,5	1,3	1,4	1,3	7,5	1,4	1,5	1,4	5,7	1,4	1,5	1,4	4,5
		[1,99]	1,1	1,2	1,1	18,3	1,2	1,3	1,3	12,2	1,4	1,5	1,5	8,8	1,5	1,6	1,5	6,8	1,5	1,7	1,6	5,5
		[1,199]	1,3	1,4	1,4	24,9	1,4	1,5	1,5	16,7	1,5	1,7	1,7	12,4	1,6	1,8	1,8	9,8	1,7	1,8	1,8	8,1
G2		[1,49]	0,6	1,1	0,5	14,4	0,6	0,8	0,5	10,5	0,6	0,8	0,6	8,4	0,6	0,9	0,6	7,1	0,6	0,9	0,7	6,2
		[1,99]	0,6	1,0	0,5	14,9	0,6	0,8	0,5	10,8	0,6	0,9	0,6	8,6	0,6	0,9	0,7	7,2	0,6	0,9	0,8	6,3
		[1,199]	0,8	1,0	0,5	15,8	0,7	0,8	0,5	11,5	0,6	0,8	0,6	9,2	0,5	0,9	0,7	7,7	0,5	0,9	0,8	6,8
G1	sjk	[1,49]	1,1	1,3	1,2	21,8	1,2	1,3	1,3	14,1	1,4	1,5	1,5	10,1	1,5	1,6	1,6	7,6	1,5	1,7	1,6	6,0
		[1,99]	1,1	1,2	1,2	19,2	1,2	1,3	1,3	12,9	1,4	1,5	1,5	9,5	1,5	1,6	1,5	7,4	1,5	1,7	1,6	6,1
		[1,149]	1,2	1,3	1,2	18,2	1,3	1,4	1,3	12,4	1,4	1,5	1,5	9,2	1,6	1,6	1,6	7,3	1,6	1,7	1,6	6,0
G2		[1,49]	0,6	1,0	0,7	15,5	0,6	0,8	0,7	11,0	0,6	0,8	0,7	8,6	0,5	0,9	0,8	7,1	0,5	1,0	0,9	6,0
		[1,99]	0,7	1,0	0,5	15,0	0,6	0,8	0,5	11,0	0,6	0,8	0,6	8,7	0,6	0,9	0,6	7,4	0,6	0,9	0,7	6,5
		[1,149]	0,7	1,0	0,4	14,6	0,7	0,8	0,4	10,9	0,6	0,8	0,5	8,8	0,6	0,9	0,6	7,6	0,7	0,9	0,7	6,7

Tabela 11 – Valores do RPD das Heurísticas detalhado por parâmetros – Fonte: Elaborado pelos autores.

Conforme já observado nas análises anteriores, a heurística H_4 teve o pior desempenho em todas as classes de problemas, com uma larga diferença em relação à demais. Já as três melhores heurísticas, pela proximidade dos resultados e o grande número de empates, acabou tendo alguma variação na primeira colocação.

4.3.6 Tempo computacional das Heurísticas Construtivas

Para os tempos computacionais das heurísticas foram elaboradas as Tabelas 12 e 13. A seguir, os tempos de execução dos problemas do Grupo 1.

Grupo 1			
H_1	H_2	H_3	H_4
1,7	2,0	1,8	0,5

Tabela 12 – Tempos Totais de CPU das Heurísticas do Grupo 1 (em segundos) – Fonte: Elaborado pelos autores.

O tempo consumido para encontrar a solução ótima pelo processo de enumeração completa é apresentado na Tabela 13, em horas, para cada opção de alfa e a totalização para os cinco alfas.

Enumeração Completa					
$\alpha = 0$	$\alpha = 0,25$	$\alpha = 0,5$	$\alpha = 0,75$	$\alpha = 1$	Tempo Total
3,079	3,077	3,078	3,078	3,078	15,390

Tabela 13 – Tempos de CPU das Heurísticas para Enumeração Completa (em horas) – Fonte: Elaborado pelos autores.

Por se ter disponível a solução ótima dos problemas do Grupo 1, é possível contabilizar a quantidade de vezes que as heurísticas atingiram a solução exata. A Tabela 14 mostra o número de soluções ótimas encontradas por cada heurística e o percentual em relação ao total de 18.000 problemas resolvidos no Grupo 1.

Soluções Ótimas					
Alfa	**H_1**	**H_2**	**H_3**	**H_4**	**Total**
0,0	6654	6284	6324	63	9.468
	34,43%	32,52%	32,72%	0,33%	
0,25	5398	5062	5064	168	8.149
	27,93%	26,19%	26,20%	0,87%	
0,5	4608	4361	4266	256	7.184
	23,84%	22,57%	22,08%	1,32%	
0,75	4184	3945	3903	368	6.766
	21,65%	20,41%	20,20%	1,90%	
1,0	7841	7351	7606	2711	10.526
	40,57%	38,04%	39,36%	14,03%	

Tabela 14 – Número de Soluções Ótimas encontradas pelas Heurísticas – Fonte: Elaborado pelos autores.

Ainda sobre a Tabela 14, a última coluna mostra o número de soluções ótimas encontradas por cada heurística, descontados os empates. Essa coluna mostra o úmero exato de soluções ótimas encontradas dentro dos 18.000 problemas testados.

Estes resultados são muito expressivos e mostram que a qualidade da solução não é produto da diluição na média, ou seja, além do resultado médio indicar a eficácia das três

melhores heurísticas, o número de problemas que elas atingiram a solução ótima atesta a sua qualidade e indica inclusive a estabilidade das soluções.

Na resolução do Grupo 2, naturalmente os problemas levaram mais tempo de CPU do que os do Grupo 1, por se tratarem de problemas de médio e grande portes. A Tabela 15 apresenta os tempos de CPU das heurísticas no Grupo 2, em segundos.

Grupo 2			
H_1	H_2	H_3	H_4
1414,1	1422,5	1423,4	25,5

Tabela 15 – Tempos Totais de CPU do Grupo 2 para cada Heurística (em segundos) – Fonte: Elaborado pelos autores.

Como pode ser visto na Tabela 15, a grande revelação aqui é o tempo de CPU gasto pela heurística H_4, que ficou extremamente abaixo das demais heurísticas. A heurística H_4 consumiu menos de meio minuto, em média, para todos os problemas do Grupo 2, enquanto as demais levaram em torno de 23 minutos cada.

Isto pode ser explicado pelo grande consumo computacional exigido pelo método de inserção utilizado nas heurísticas H_1, H_2 e H_3 e pela quantidade bem menor de testes feitos pela heurística H_4, que em cada problema, constrói m–1 sequências diretamente, sem etapas de inserção, e escolhe a melhor. Portanto, vê-se aqui a principal vantagem da heurística H_4 em relação às demais.

Entretanto, a excelente qualidade da solução fornecida pelas heurísticas H_1, H_2 e H_3, que obtiveram resultados pouco acima de 1% de desvio da solução ótima, não justifica a utilização da H_4. Para problemas de médio e grande portes, o tempo consumido por estas três melhores heurísticas é ainda suficientemente aceitável.

5. CONSIDERAÇÕES FINAIS

Esta pesquisa investigou o problema de programação em *flow shop* bicritério com minimização ponderada do *makespan* e do tempo de fluxo, incluindo características realísticas como datas de liberação das tarefas diferentes de zero e tempos de *setup* independentes da sequência. Este é um problema bastante comum nas indústrias e foi pouco explorado na literatura específica, não tendo sido encontrado nenhum trabalho abordando exatamente o mesmo ambiente de produção.

Assim, as contribuições deste estudo abrangem tanto a teoria como a prática. No campo teórico, os trabalhos publicados relacionados ao problema tratado foram organizados e classificados em ordem cronológica e de embasamento em métodos de solução prévios. Foram revisados estudos de problemas de *flow shop* bicritério *makespan* e tempo de fluxo com número de máquinas genérico maior que dois. Estes trabalhos foram dispostos em estrutura de árvore, indicando a evolução e construção do conhecimento em relação ao problema em questão.

Já as contribuições práticas foram ainda mais expressivas e encorajadoras, chegando a um resultado heurístico com desvio apenas um pouco acima de 1% da solução ótima, em média, e com uma quantidade considerável de problemas em que se obteve a solução exata, ou seja, a solução fornecida pela melhor heurística é praticamente ótima, e ainda com tempo computacional aceitável.

Também pode-se citar como uma contribuição significativa, conferindo maior mérito ao trabalho, o fato do *flow shop* tratado se desdobrar em três problemas diferentes, dependendo do peso das medidas da função objetivo. Quando o *makespan* tem peso zero, o problema é monocritério de tempo médio de fluxo, e quando o peso do tempo médio de fluxo é zero, o problema passa a ser de minimização do *makespan*. Nas outras possibilidades de combinação de pesos, o problema continua bicritério.

A estrutura do problema foi analisada em profundidade, assim como os trabalhos publicados relacionados, e com este embasamento foram feitas duas rodadas de experimentação computacional. Na primeira, foram implementadas oito regras de prioridade desenvolvidas nesta pesquisa, que já atingiram resultados bastante animadores, conforme discutido nos capítulos anteriores. As três melhores foram utilizadas na proposição de três métodos heurísticos da segunda rodada e mais um quarto, que não

requer ordenação inicial. Estas quatro heurísticas construtivas foram elaboradas a partir de algoritmos clássicos e outros recentes da literatura específica.

Vale ressaltar que as regras de prioridade para programação da produção são métodos de solução bastante elementares racionalmente, embora exijam um considerável conhecimento técnico da área e um exercício reflexivo, além do esforço no desenvolvimento do software de implementação de tais regras. Por outro lado, as heurísticas construtivas em geral pressupõem métodos de solução mais eficazes do que as regras de prioridade simplistas, devido à ampliação da gama de sequências avaliadas e a flexibilidade da busca pela solução conferida pelos métodos de solução (como é o caso da heurística clássica NEH, por exemplo).

Na experimentação computacional foram resolvidos 39.600 problemas, sendo delineada em dois grupos, em que o primeiro é composto por problemas de pequeno porte e se utilizou como base de comparação a solução ótima obtida pelo método de enumeração completa, e o segundo é formado por problemas de médio e grande portes. A comparação dos resultados ocorreu pelo Desvio Relativo Percentual (RPD, em inglês) e pelo tempo de CPU.

As três melhores heurísticas construtivas (H_1, H_2, H_3) utilizam diferentes regras de ordenação inicial e em seguida aplicam o método de inserção, empregado em heurísticas clássicas como a NEH e a N&M. A pior heurística construtiva (H_4) teve desvios bem maiores do que as demais, porém tempo de CPU muito menor, sem entretanto, justificar o seu uso pela falta de qualidade da solução.

Concluindo, pode-se afirmar que os objetivos desta pesquisa foram plenamente alcançados, tanto na parte conceitual como também na área experimental, pela ampla amostra de problemas-testes analisados e os significativos resultados fornecidos pelas heurísticas.

Como sugestão para trabalhos futuros, indica-se um aumento do número de tarefas deste estudo, visando obter outra realidade quando diante de problemas de maior porte. Indica-se também um detalhamento na análise do funcionamento da heurística H_4, visando eventualmente uma melhoria no seu desempenho, a proposição e implementação de métodos de solução exata como modelos de Programação Linear Inteira Mista e algoritmo *Branch-and-bound* e a consideração de novas restrições ao problema, como prazos de

entrega e/ou tempos de *setup* dependentes da sequência, e consequentemente outras medidas de desempenho ponderadas, incluindo medidas de atraso e/ou adiantamento.

REFERÊNCIAS

ALLAHVERDI, A. (2003). The two-and m-machine flow shop scheduling problems with bicriteria of makespan and mean flowtime**. European journal of operational research**, 147(2), 373-396.

BAKER, K. R. (1974). **Introduction to sequencing and scheduling**. New York: John Wiley & Sons, Inc.

BAKER, K. R., & TRIETSCH, D. (2009). Principles of sequencing and scheduling. John Wiley & Sons.

BRANCO, F. J. C., NAGANO, M. S., & MOCCELLIN, J. V. (2008). Análise multi-critério na programação da produção em sistemas no-wait flow shop. **Revista Gestão Industrial**, 4(2).

CAMPBELL, H. G., DUDEK, R. A., & SMITH, M. L. (1970). A heuristic algorithm for the n job, m machine sequencing problem. **Management science**, 16(10), B-630.

DAVIS, M. M., CHASE, R. B., & AQUILANO, N. J. (2001). Fundamentos da administração da produção. Bookman.

EREN, T. (2010). A bicriteria< i> m-machine flowshop scheduling with sequence-dependent setup times. **Applied Mathematical Modelling**, 34(2), 284-293.

FARBER, G, COVES, A. (2005). Overview on sequencing in mixed model flowshop production line with static and dynamic context. Universitat Politècnica de Catalunya.

FRAMINAN, J. M., LEISTEN, R., & RUIZ-USANO, R. (2002). Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimisation. **European Journal of Operational Research**, 141(3), 559-569.

FUCHIGAMI, H.Y. (2005). Métodos heurísticos construtivos para o problema de programação da produção em sistemas flow shop híbridos com tempos de preparação das máquinas assimétricos e dependentes da sequência. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.

FUCHIGAMI, H.Y. (2014). Introdução ao Sequenciamento da Produção. Catalão: UFG. Material didático. Versão 6.0.

GAITHER, N.; FRAZIER, G. (2002). **Administração da produção e operações**. 8. Ed. São Paulo: Pioneira.

GANGADHARAN, R., & RAJENDRAN, C. (1994). A simulated annealing heuristic for scheduling in a flowshop with bicriteria. **Computers & Industrial Engineering**,27(1), 473-476.

HAUPT, R. (1989). A survey of priority rule-based scheduling. **Operations-Research-Spektrum**, 11(1), 3-16.

HO, J.C., CHANG, Y.-L. (1991). A new heuristic for the n-job, m-machine flowshop problem, **European Journal of Operational Research** 52, 194-202. International Journal of Advanced Manufacturing Technology, 27, 804-815.

JOHNSON, S.M. (1954). Optimal two and three-stage production schedules with set-up times included. **Naval Research Logistics Quarterly** 1, 61–68.

JUNG, C.F. (2004). Metodologia para pesquisa & desenvolvimento: aplicada a novas tecnologias, produtos e processos. Rio de Janeiro: Axcel Books.

MARTINS, R.A. (2010). Abordagens Quantitativa e Qualitativa. In: **MIGUEL, P.A.C. (Org.). Metodologia de pesquisa em Engenharia de Produção e Gestão de Operações**. Rio de Janeiro: Elsevier; p. 45-61, cap. 3.

MOCCELLIN, J. V. (1995). A new heuristic method for the permutation flow shop scheduling problem. **Journal of the Operational Research Society**, 883-886.

MOCCELLIN, J.V. (2005). Técnicas de Sequenciamento e Programação de Operações em Máquinas. 74p. Publicação Escola de Engenharia de São Carlos, Universidade de São Paulo, Apostila.

NAGANO, M. S., & MOCCELLIN, J. V. (2002). A high quality solution constructive heuristic for flow shop sequencing. **Journal of the Operational Research Society**, 53(12), 1374-1379.

NAKANO, D. (2010). Métodos de Pesquisa Adotados na Engenharia de Produção e Gestão de Operações. In: MIGUEL, P.A.C. (Org.). Metodologia de pesquisa em Engenharia de Produção e Gestão de Operações. Rio de Janeiro: Elsevier; p. 63-72, cap. 4.

NAWAZ, M., ENSCORE JR, E. E., & HAM, I. (1983). A heuristic algorithm for the< i> m</i>-machine,< i> n</i>-job flow-shop sequencing problem. **Omega**,11(1), 91-95.

PASUPATHY, T., RAJENDRAN, C., & SURESH, R. K. (2006). A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs. **Omega, The International Journal of Advanced Manufacturing Technology**, 27(7-8), 804-815.

PINEDO, M. L. (2012). Scheduling: Theory, Algorithms, and Systems. New York: Springer. 4a ed.

RAJENDRAN, C. (1995). Heuristics for scheduling in flowshop with multiple objectives. **European journal of operational research**, 82(3), 540-555.

RAJENDRAN, C.; ZIEGLER, H. (2004) Ant-colony algorithms for permutation flowshop scheduling to minize makespna/total flowtime of jobs. **European Journal of Operation Research**, Amsterdam, v. 155, n.2, p. 426-436, June.

RAJKUMAR, R., SHAHABUDEEN, P., NAGARAJ, P., ARUNACHALAM, S., & PAGE, T. (2009). A Bi-Criteria Approach to the M-machine Flowshop Scheduling Problem. **Studies in Informatics and Control**, 18(2), 127-136.

RAVINDRAN D., NOORUL HAQ A., SELVAKUAR S.J. & SIVARAMAN R., (2005). Flow shop scheduling with multiple objective of minimizing makespan and total flowtime. **International Journal of Advanced Manufacturing Technology**, Vol. 25, pp.1007–1012.

REID, R. D.; SANDERS, N. R. (2005). Gestão de Operações. Rio de Janeiro: LTC.

SRIDHAR, J., & RAJENDRAN, C. (1996). Scheduling in flowshop and cellular manufacturing systems with multiple objectives—a genetic algorithmic approach. **Production Planning & Control**, 7(4), 374-382.

TAILLARD, E. (1990). Benchmarks for basic scheduling problems. **European Journal of Operational Research**. Amsterdam, V. 64, n. 2, p 278-285, jan 1993.

TASGETIREN, M. F., LIANG, Y. C., SEVKLI, M. E GENCYILMAZ, G. (2007) A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem, **European Journal of Operational Research**, 177, 1930-1947.

TUBINO, D. F. (2007) Planejamento e controle da produção - Teoria e prática. 1. Ed. São Paulo: Atlas.

VARADHARAJAN, T. K.; RAJENDRAN, C. (2005). A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. **European Journal of Operational Research**, n.167, p. 772-795.

YEH, W. C., & ALLAHVERDI, A. (2004). A branch-and-bound algorithm for the three-machine flowshop scheduling problem with bicriteria of makespan and total flowtime. **International Transactions in Operational Research**, 11(3), 323-339.

YENISEY, M. M., & YAGMAHAN, B. (2014). Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. **Omega**, 45, 119-135.

ZACCARELLI, S. (1987). Programação e Controle da Produção 8ª ed. Livraria Pioneira, São Paulo.

GLOSSÁRIO

- ***Flow shop:*** todas as tarefas têm o mesmo fluxo de processamento nas máquinas.
- ***Flow shop permutacional:*** trata-se de um *flow shop* onde a ordem de processamento das tarefas é a mesma em todas as máquinas.
- ***Makespan:*** duração total da programação; data de término da última tarefa.
- ***Release Date:*** data de liberação de uma tarefa, ou seja, ponto onde ela possa começar a ser executada.
- ***Setup Time:*** tempo de preparação da máquina.
- ***Flowtime:*** tempo de fluxo ou tempo de permanência de uma tarefa.
- ***Completion Time*****:** data de término da tarefa na última máquina.
- ***Due Date:*** data de entrega ou prazo de término de uma tarefa.

APÊNDICE A

Grupo 1

		Alfa = 0								Alfa = 0,25								Alfa = 0,5								Alfa = 0,75								Alfa = 1							
		R1	R2	R3	R4	R5	R6	R7	R8	R1	R2	R3	R4	R5	R6	R7	R8	R1	R2	R3	R4	R5	R6	R7	R8	R1	R2	R3	R4	R5	R6	R7	R8	R1	R2	R3	R4	R5	R6	R7	R8
n	**5**	14,2	7,1	11,5	14,6	5,7	8,9	22,7	21,2	11,5	5,8	9,5	12,6	6,4	9,1	15,9	17,0	10,5	5,8	9,0	12,2	7,8	10,3	12,2	15,1	10,4	6,3	9,3	12,5	9,5	11,9	10,0	14,4	10,7	7,2	9,9	13,1	11,2	13,5	8,7	14,2
	6	16,2	8,5	12,1	15,5	6,8	9,5	24,0	22,7	13,0	6,9	9,9	13,4	7,2	9,7	16,7	18,2	11,8	6,8	9,5	12,9	8,6	10,9	12,9	16,2	11,5	7,2	9,7	13,2	10,2	12,4	10,6	15,4	11,6	8,0	10,2	13,7	11,9	14,0	9,2	15,2
	7	17,5	9,5	12,8	16,7	7,6	10,3	25,3	22,8	14,0	7,6	10,4	14,3	7,9	10,4	17,6	18,2	12,6	7,4	9,8	13,7	9,1	11,5	13,5	16,1	12,2	7,7	9,9	13,9	10,7	13,0	11,1	15,2	12,2	8,3	10,3	14,4	12,3	14,5	9,5	14,9
	8	18,6	10,4	13,5	17,3	8,6	11,1	26,1	23,9	14,8	8,3	10,9	14,8	8,6	11,0	18,2	19,0	13,2	7,8	10,1	14,1	9,7	12,0	13,9	16,8	12,7	8,0	10,1	14,2	11,1	13,4	11,4	15,8	12,6	8,5	10,4	14,6	12,5	14,8	9,7	15,4
	10	20,5	11,9	14,8	18,5	9,8	12,1	27,5	25,2	16,2	9,4	11,8	15,8	9,5	11,9	19,1	19,9	14,3	8,7	10,8	15,0	10,4	12,8	14,6	17,4	13,5	8,7	10,5	14,9	11,5	13,9	11,8	16,2	13,1	8,9	10,6	15,1	12,7	15,1	10,1	15,5
m	**2**	20,6	8,8	13,7	17,9	8,1	10,6	34,6	29,0	14,4	6,0	10,2	14,0	7,1	9,7	21,2	20,7	11,1	5,0	8,7	12,2	7,1	9,8	13,5	16,3	9,3	4,6	8,1	11,4	7,5	10,4	8,6	13,7	8,2	4,7	7,9	11,1	8,2	11,1	5,3	12,1
	3	19,1	9,5	13,7	17,6	8,4	11,4	27,9	25,8	15,0	7,3	10,8	14,9	8,4	11,2	19,0	20,2	13,2	6,8	9,8	14,2	9,3	12,2	14,0	17,5	12,5	7,0	9,7	14,2	10,6	13,4	10,9	16,2	12,3	7,4	9,9	14,5	12,0	14,8	8,9	15,5
	5	16,6	10,1	13,1	16,7	8,0	10,9	21,8	21,4	14,2	8,6	11,0	15,0	8,8	11,4	16,5	18,2	13,6	8,6	10,7	14,9	10,6	12,9	13,8	17,1	13,8	9,2	11,1	15,5	12,5	14,7	12,4	16,8	14,3	10,1	11,7	16,4	14,5	16,6	11,6	17,0
	10	13,3	9,3	11,3	13,9	6,2	8,6	16,2	16,4	12,0	8,4	10,0	12,8	7,4	9,4	13,4	14,7	12,0	8,7	10,1	13,1	9,5	11,1	12,4	14,5	12,6	9,5	10,8	13,9	11,7	13,0	12,1	14,8	13,4	10,5	11,6	14,8	13,9	15,0	12,1	15,5
rj	**[1,49]**	18,6	8,4	8,7	15,6	6,3	8,6	21,5	19,3	15,0	6,9	7,1	13,7	6,9	9,1	15,0	15,7	13,6	6,7	6,9	13,3	8,4	10,5	11,4	14,1	13,0	7,1	7,3	13,6	10,1	12,2	9,2	13,5	12,9	7,7	7,9	14,2	11,8	13,8	7,9	13,4
	[1,99]	17,7	9,1	11,1	16,7	7,2	10,1	23,2	21,4	14,2	7,4	9,1	14,5	7,7	10,3	16,2	17,2	12,8	7,1	8,6	13,9	9,0	11,5	12,4	15,4	12,4	7,4	8,8	14,1	10,6	13,0	10,1	14,6	12,4	8,0	9,3	14,6	12,2	14,6	8,7	14,4
	[1,199]	15,9	10,8	19,0	17,3	9,5	12,4	30,7	28,7	12,4	8,5	15,3	14,4	9,1	11,8	21,3	22,5	11,1	8,0	14,0	13,5	9,9	12,4	16,5	19,5	10,8	8,2	13,5	13,5	11,1	13,5	13,6	18,1	10,9	8,7	13,6	13,8	12,4	14,7	11,7	17,3
sjk	**[1,49]**	17,4	9,8	15,4	18,3	8,1	9,5	29,9	27,0	13,8	7,8	12,4	15,1	8,7	9,9	20,7	21,4	12,5	7,5	11,5	14,0	10,2	11,3	15,8	18,9	12,2	7,9	11,5	13,9	12,1	13,1	12,8	17,7	12,3	8,6	11,9	14,2	13,9	14,8	10,9	17,2
	[1,99]	17,1	9,3	12,4	16,3	7,5	10,5	23,8	22,3	13,7	7,5	10,2	14,1	7,8	10,5	16,6	17,9	12,4	7,2	9,6	13,6	9,0	11,5	12,8	15,9	12,0	7,5	9,6	13,8	10,4	12,8	10,5	15,0	12,0	8,1	10,1	14,3	11,9	14,2	9,1	14,7
	[1,149]	17,7	9,3	11,0	15,0	7,4	11,1	21,6	20,2	14,1	7,5	9,0	13,4	7,3	10,9	15,1	16,1	12,6	7,1	8,4	13,1	8,1	11,7	11,6	14,3	12,0	7,3	8,5	13,5	9,3	12,8	9,6	13,5	11,9	7,8	8,9	14,1	10,5	14,0	8,3	13,2

Grupo 2

		Alfa = 0								Alfa = 0,25								Alfa = 0,5								Alfa = 0,75								Alfa = 1							
		R1	R2	R3	R4	R5	R6	R7	R8	R1	R2	R3	R4	R5	R6	R7	R8	R1	R2	R3	R4	R5	R6	R7	R8	R1	R2	R3	R4	R5	R6	R7	R8	R1	R2	R3	R4	R5	R6	R7	R8
n	**15**	8,7	4,8	6,1	7,8	1,6	3,2	10,9	10,6	6,7	3,2	4,3	6,4	2,2	3,5	7,4	8,3	5,8	2,7	3,6	6,0	3,2	4,4	5,5	7,2	5,6	2,7	3,5	6,1	4,4	5,5	4,5	6,9	5,7	3,0	3,7	6,5	5,6	6,6	3,9	6,8
	20	8,4	4,7	6,0	7,2	1,4	2,8	10,6	10,0	6,4	3,1	4,2	5,9	1,9	3,2	7,1	7,7	5,5	2,5	3,4	5,6	2,7	4,0	5,2	6,6	5,2	2,5	3,2	5,7	3,8	5,0	4,1	6,2	5,2	2,7	3,4	6,1	4,9	6,1	3,5	6,2
	30	7,7	4,5	5,6	6,2	1,2	2,1	10,0	9,0	5,6	2,8	3,7	5,0	1,5	2,4	6,5	6,7	4,7	2,2	3,0	4,8	2,2	3,2	4,5	5,7	4,4	2,1	2,8	5,0	3,1	4,2	3,4	5,3	4,5	2,3	3,0	5,4	4,1	5,2	2,9	5,3
	50	7,1	4,6	5,3	5,0	1,0	1,4	10,0	7,9	5,1	2,8	3,4	4,0	1,1	1,8	6,2	5,7	4,1	2,0	2,6	3,8	1,6	2,4	4,1	4,6	3,8	1,8	2,3	4,0	2,4	3,4	3,0	4,2	3,8	2,0	2,4	4,5	3,3	4,4	2,4	4,2
	80	6,8	4,7	5,2	4,2	1,0	1,0	10,7	7,4	4,6	2,7	3,2	3,3	0,9	1,3	6,5	5,1	3,5	1,8	2,2	3,0	1,2	1,9	4,1	4,0	3,1	1,5	1,9	3,3	1,9	2,7	2,7	3,5	3,1	1,6	2,0	3,8	2,7	3,6	1,9	3,5
	100	6,7	4,9	5,3	3,9	1,0	0,8	11,4	7,2	4,4	2,8	3,2	3,0	0,8	1,1	6,8	4,9	3,3	1,8	2,1	2,8	1,1	1,6	4,2	3,6	2,8	1,5	1,7	3,0	1,6	2,4	2,6	3,1	2,8	1,6	1,8	3,5	2,4	3,3	1,8	3,1
m	**5**	10,3	5,7	6,8	7,0	1,6	2,1	16,7	11,8	7,2	3,2	4,1	5,7	1,6	2,7	10,0	8,3	5,7	2,2	2,9	5,4	2,2	3,6	6,2	6,6	5,1	1,9	2,5	5,7	3,1	4,8	3,9	5,9	5,2	2,2	2,7	6,4	4,3	6,2	2,6	5,9
	10	7,6	4,8	5,8	5,7	1,3	1,9	10,5	8,9	5,5	3,0	3,9	4,6	1,5	2,3	6,8	6,6	4,6	2,3	3,0	4,4	2,2	3,1	4,7	5,6	4,3	2,2	2,8	4,6	3,1	4,0	3,6	5,2	4,3	2,3	2,9	5,1	4,1	5,1	2,9	5,1
	15	6,5	4,2	5,1	5,2	1,0	1,8	8,1	7,4	4,8	2,8	3,5	4,2	1,3	2,0	5,4	5,6	4,0	2,2	2,8	3,9	1,9	2,6	3,9	4,7	3,8	2,1	2,6	4,0	2,7	3,4	3,2	4,4	3,8	2,2	2,7	4,4	3,6	4,3	2,8	4,4
	20	5,9	4,0	4,7	5,0	0,9	1,7	7,0	6,7	4,3	2,7	3,3	4,0	1,2	1,9	4,8	5,0	3,6	2,1	2,7	3,6	1,8	2,4	3,5	4,3	3,4	2,0	2,5	3,7	2,6	3,1	2,9	4,0	3,5	2,1	2,5	4,0	3,4	3,9	2,6	4,0
rj	**[1,49]**	8,2	5,1	5,3	5,7	1,1	1,7	10,4	8,4	6,0	3,2	3,4	4,5	1,3	2,0	6,5	6,1	4,9	2,4	2,5	4,2	1,9	2,7	4,3	5,0	4,5	2,2	2,3	4,4	2,7	3,7	3,1	4,6	4,5	2,3	2,4	4,9	3,7	4,7	2,5	4,6
	[1,99]	7,8	4,8	5,4	5,9	1,2	1,8	10,4	8,5	5,6	3,0	3,5	4,8	1,3	2,2	6,6	6,3	4,6	2,3	2,6	4,5	2,0	2,9	4,4	5,2	4,3	2,1	2,4	4,7	2,9	3,9	3,2	4,8	4,3	2,3	2,6	5,1	3,9	4,9	2,6	4,8
	[1,199]	6,7	4,0	6,1	5,6	1,4	2,1	11,0	9,1	4,8	2,5	4,2	4,5	1,6	2,4	7,2	6,8	4,0	1,9	3,3	4,3	2,2	3,1	5,0	5,7	3,8	1,8	3,1	4,5	3,0	4,0	3,8	5,3	3,8	2,0	3,1	4,9	3,9	5,0	3,1	5,2
sjk	**[1,49]**	8,0	4,9	6,4	7,3	1,2	1,6	11,9	9,9	5,7	2,9	4,1	5,4	1,5	1,9	7,6	7,2	4,6	2,1	3,1	4,7	2,3	2,7	5,1	5,9	4,2	1,9	2,8	4,6	3,3	3,8	3,7	5,4	4,3	2,1	2,9	4,9	4,5	4,9	3,0	5,3
	[1,99]	7,4	4,6	5,4	5,5	1,2	1,9	10,2	8,4	5,4	2,9	3,6	4,5	1,4	2,2	6,5	6,2	4,4	2,2	2,7	4,3	2,0	2,9	4,4	5,1	4,1	2,0	2,5	4,5	2,8	3,9	3,3	4,7	4,1	2,2	2,6	5,0	3,8	4,9	2,6	4,7
	[1,149]	7,2	4,5	5,0	4,4	1,3	2,1	9,6	7,8	5,3	2,9	3,3	3,9	1,3	2,5	6,2	5,8	4,5	2,3	2,6	4,0	1,8	3,1	4,3	4,9	4,1	2,1	2,4	4,4	2,4	3,9	3,2	4,5	4,2	2,3	2,5	5,0	3,2	4,8	2,6	4,5

APÊNDICE B

Grupo 1

		Alfa = 0				Alfa = 0,25				Alfa = 0,5				Alfa = 0,75				Alfa = 1			
		H1	H2	H3	H4	H1	H2	H3	H4	H1	H2	H3	H4	H1	H2	H3	H4	H1	H2	H3	H4
n	**5**	0,7	0,8	0,8	18,1	0,7	0,8	0,9	12,1	0,9	1,0	1,0	8,8	1,0	1,1	1,1	6,8	1,1	1,2	1,2	5,6
	6	0,9	1,0	1,0	18,9	1,0	1,1	1,1	12,6	1,2	1,3	1,3	9,2	1,3	1,4	1,3	7,2	1,4	1,4	1,4	5,8
	7	1,1	1,2	1,2	19,8	1,2	1,3	1,3	13,1	1,4	1,5	1,4	9,5	1,5	1,7	1,6	7,4	1,6	1,7	1,6	6,0
	8	1,3	1,5	1,4	20,4	1,4	1,5	1,5	13,6	1,6	1,7	1,7	10,0	1,7	1,9	1,7	7,8	1,7	1,9	1,7	6,3
	10	1,7	1,9	1,7	21,3	1,8	1,9	1,8	14,3	1,9	2,0	2,0	10,5	2,0	2,1	2,0	8,1	2,0	2,2	2,0	6,5
m	**2**	1,1	1,1	1,1	30,3	1,0	1,0	1,0	18,7	0,9	0,9	0,9	12,2	0,8	0,8	0,8	8,1	0,6	0,6	0,6	5,3
	3	1,3	1,4	1,3	22,5	1,3	1,4	1,3	15,0	1,4	1,5	1,4	10,8	1,5	1,5	1,4	8,3	1,3	1,4	1,4	6,6
	5	1,2	1,4	1,3	15,8	1,4	1,6	1,5	11,2	1,7	1,8	1,8	8,9	1,9	2,0	2,0	7,6	2,1	2,2	2,2	6,8
	10	1,0	1,2	1,1	10,3	1,2	1,4	1,4	7,7	1,5	1,8	1,7	6,5	1,8	2,2	2,1	5,8	2,1	2,5	2,4	5,5
rj	**[1,49]**	1,0	1,1	1,0	15,9	1,1	1,2	1,1	10,5	1,3	1,4	1,3	7,5	1,4	1,5	1,4	5,7	1,4	1,5	1,4	4,5
	[1,99]	1,1	1,2	1,1	18,3	1,2	1,3	1,3	12,2	1,4	1,5	1,5	8,8	1,5	1,6	1,5	6,8	1,5	1,7	1,6	5,5
	[1,199]	1,3	1,4	1,4	24,9	1,4	1,5	1,5	16,7	1,5	1,7	1,7	12,4	1,6	1,8	1,8	9,8	1,7	1,8	1,8	8,1
sjk	**[1,49]**	1,1	1,3	1,2	21,8	1,2	1,3	1,3	14,1	1,4	1,5	1,5	10,1	1,5	1,6	1,6	7,6	1,5	1,7	1,6	6,0
	[1,99]	1,1	1,2	1,2	19,2	1,2	1,3	1,3	12,9	1,4	1,5	1,5	9,5	1,5	1,6	1,5	7,4	1,5	1,7	1,6	6,1
	[1,149]	1,2	1,3	1,2	18,2	1,3	1,4	1,3	12,4	1,4	1,5	1,5	9,2	1,6	1,6	1,6	7,3	1,6	1,7	1,6	6,0

Grupo 2

		Alfa = 0				Alfa = 0,25				Alfa = 0,5				Alfa = 0,75				Alfa = 1			
		H1	H2	H3	H4	H1	H2	H3	H4	H1	H2	H3	H4	H1	H2	H3	H4	H1	H2	H3	H4
n	**15**	0,6	1,1	0,8	12,9	0,6	1,0	0,9	9,5	0,7	1,2	1,0	7,8	0,8	1,3	1,1	6,9	0,9	1,4	1,2	6,3
	20	0,6	1,2	0,8	13,8	0,6	1,1	0,8	10,2	0,6	1,1	0,9	8,4	0,7	1,2	0,9	7,3	0,8	1,3	1,0	6,6
	30	0,6	1,2	0,6	14,7	0,6	1,0	0,6	10,9	0,6	1,0	0,7	8,9	0,6	1,0	0,8	7,7	0,6	1,1	0,9	6,9
	50	0,6	1,0	0,5	15,7	0,6	0,8	0,4	11,5	0,5	0,7	0,5	9,2	0,5	0,8	0,5	7,7	0,5	0,8	0,6	6,7
	80	0,7	0,9	0,3	16,4	0,7	0,6	0,2	11,7	0,5	0,5	0,3	9,0	0,4	0,6	0,3	7,3	0,4	0,6	0,5	6,2
	100	0,8	0,8	0,2	16,7	0,8	0,6	0,2	11,9	0,6	0,5	0,2	9,0	0,4	0,5	0,3	7,2	0,3	0,5	0,4	5,9
m	**5**	1,2	0,9	0,4	20,7	1,1	0,7	0,4	13,4	0,9	0,7	0,4	9,1	0,7	0,7	0,5	6,3	0,5	0,8	0,7	4,3
	10	0,7	1,0	0,5	15,2	0,7	0,8	0,5	11,2	0,6	0,8	0,6	9,0	0,7	0,9	0,7	7,6	0,7	0,9	0,8	6,8
	15	0,4	1,1	0,6	12,8	0,5	0,8	0,6	10,0	0,5	0,9	0,7	8,6	0,5	1,0	0,7	7,8	0,6	1,0	0,8	7,3
	20	0,3	1,1	0,6	11,4	0,3	0,9	0,6	9,2	0,4	0,9	0,7	8,2	0,4	1,0	0,7	7,6	0,5	1,0	0,8	7,3
rj	**[1,49]**	0,6	1,1	0,5	14,4	0,6	0,8	0,5	10,5	0,6	0,8	0,6	8,4	0,6	0,9	0,6	7,1	0,6	0,9	0,7	6,2
	[1,99]	0,6	1,0	0,5	14,9	0,6	0,8	0,5	10,8	0,6	0,9	0,6	8,6	0,6	0,9	0,7	7,2	0,6	0,9	0,8	6,3
	[1,199]	0,8	1,0	0,5	15,8	0,7	0,8	0,5	11,5	0,6	0,8	0,6	9,2	0,5	0,9	0,7	7,7	0,5	0,9	0,8	6,8
sjk	**[1,49]**	0,6	1,0	0,7	15,5	0,6	0,8	0,7	11,0	0,6	0,8	0,7	8,6	0,5	0,9	0,8	7,1	0,5	1,0	0,9	6,0
	[1,99]	0,7	1,0	0,5	15,0	0,6	0,8	0,5	11,0	0,6	0,8	0,6	8,7	0,6	0,9	0,6	7,4	0,6	0,9	0,7	6,5
	[1,149]	0,7	1,0	0,4	14,6	0,7	0,8	0,4	10,9	0,6	0,8	0,5	8,8	0,6	0,9	0,6	7,6	0,7	0,9	0,7	6,7