UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

PROGRAMA DE MESTRADO EM LOGÍSTICA E PESQUISA OPERACIONAL

ANA BEATRIZ GENTIL DE FARIAS

SISTEMAS PRODUÇÃO *FLOW SHOP* HÍBRIDOS COM TEMPOS EXPLÍCITOS DE PREPARAÇÃO DAS MÁQUINAS E PROCESSAMENTO CONTÍNUO DAS TAREFAS

FORTALEZA

2014

ANA BEATRIZ GENTIL DE FARIAS

SISTEMAS DE PRODUÇÃO *FLOW SHOP* HÍBRIDOS COM TEMPOS EXPLÍCITOS DE PREPARAÇÃO DAS MÁQUINAS E PROCESSAMENTO CONTÍNUO DAS TAREFAS

Tese ou Dissertação apresentada ao Programa de Pós-Graduação em Logística e Pesquisa Operacional da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Mestre em Logística e Pesquisa Operacional. Área de concentração: Pesquisa Operacional.

Orientador: Prof. Dr. João Vitor Moccelin.

FORTALEZA

2014

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca de Pós Graduação em Engenharia

F238 s Farias, Ana Beatriz Gentil de.
Sistemas produção flow shop híbridos com tempos explícitos de preparação das máquinas e processamento contínuo das tarefas / Ana Beatriz Gentil de Farias. – 2014.
71 f. : il. color., enc. ; 30 cm.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós – Graduação em Logística e Pesquisa Operacional, Fortaleza, 2014.
Área de Concentração: Pesquisa Operacional.
Orientação: Prof. Dr. João Vitor Moccelin.

1. Logística. 2. Pesquisa Operacional. I. Título.

CDD 658.78

ANA BEATRIZ GENTIL DE FARIAS

SISTEMAS DE PRODUÇÃO *FLOW SHOP* HÍBRIDOS COM TEMPOS EXPLÍCITOS DE PREPARAÇÃO DAS MÁQUINAS E PROCESSAMENTO CONTÍNUO DAS TAREFAS

Tese ou Dissertação apresentada ao Programa de Pós-Graduação em Logística e Pesquisa Operacional da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em logísitca e pesquisa operacional. Área de concentração: Pesquisa Operacional.

Aprovada em: 24 / 03 / 2014.

BANCA EXAMINADORA

Prof. Dr. João Vitor Moccellin
(Orientador/UFC)

Prof. Dr. Bruno de Athayde Prata
(Examinador Interno/UFC)

Prof. Dr. Jobson de Queiroz Oliveira
(Examinador Externo/UECE)

Aos que já deixaram, aos que deixam e aos que deixarão algo de positivo na minha vida.

AGRADECIMENTO

Agradeço a todos os meus familiares, em especial aos meus irmãos Celina, Eduardo e Renata pela amizade, apoio e momentos de alegria e até os de tristeza compartilhados e superados.

Ao meu pai, Bernardo, e a minha mãe, Helena, por sempre me mostrarem a importância dos estudos e por sempre terem tentado despertar em mim um senso crítico dos fatos.

Ao meu namorado, melhor amigo e companheiro de todas as horas, Ramon pelo carinho, compreensão, amor e solidariedade inefável. E por sempre me apoiar em todas as minhas decisões.

Ao meu professor orientador João Vitor Moccelin pela paciência, dedicação e amizade. Com certeza um profissional que tem bastante conhecimento para transmitir, e que admiro bastante.

Ao querido professor João Bosco Arruda, que estimo demais por ser uma pessoa muito boa e crítica. Aprendi muito com professor Bosco.

Ao professor Maxweel Veras Rodrigues, excelente professor, pessoa e amigo.

Aos meus colegas e amigos do colégio, que conheci na graduação, no mestrado, e na vida de uma maneira geral. Em especial e em ordem alfabética: Abrãao Ramos, Ana Nery, Juarez Alves, Leonardo Serafim, Mislene Rosado, Professor Móises, Pedro Felipe, Raphael Braga, Rúbia e Samille de Fátima.

Aos meus colegas e amigos da ASTEF (Associação Técnico-Científica Paulo de Frontim). Em especial, Dalva Carvalho e Alessandra Vidal pelo apoio pessoal e profissional. Obrigada por acreditarem tanto em mim, muitas vezes vocês me deram força mesmo sem perceberem. Da ASTEF, também agradeço a Andrea Prudente, Eveline Oliveira Viana, Gilcemar Lourenço, Helena Bacelar de Castro, Maria Luiza, Rita Maria da Silva, Vânia Gadelha, Sandra Mesquita e Socorro Catajy. Foi muito bom ter conhecido e trabalhado com essa equipe.

Ao diretor do Centro de Tecnologia e diretor presidente da ASTEF (Associação Técnico-Científica Paulo de Frontim), professor José de Paula Barros Neto, pelo aprendizado e por me ter permitido trabalhar na fundação.

Ao professor do GESLOG, Fernando Nunes, por ter sido meu orientador na ASTEF e pelos ensinamentos transmitidos.

Por último agradeço novamente a minha família, amigos e namorado, pelo apoio. Nos últimos dois anos, em especial no último semestre, com certeza amadureci bastante do ponto de vista profissional, e para cumprir tantos compromissos firmados, muitas vezes me privei de momentos com meus entes queridos. Obrigada por me darem força de seguir sempre na luta!

“Mais que de máquinas, precisamos de humanidade.” Charles Chaplin

RESUMO

O estudo trata dos problemas de Programação da Produção em ambiente *Flow Shop* com máquinas múltiplas nos estágios de produção, nos quais o tempo de *setup* não está inserido no tempo de processamento das operações. Podendo ser um sistema produtivo em que o *setup* depende ou não da sequência das tarefas, nos quais as tarefas são realizadas sem interrupção entre operações sucessivas, também chamado sistema de produção *no-wait*. Sendo proposto neste trabalho, vinte métodos de soluções heurísticas definidos por Regras de Prioridade, as quais fornecem uma ordenação das tarefas a ser seguida na sua programação, que é feita tarefa por tarefa, sucessivamente.

Destas vinte heurísticas propostas, sete foram implementadas para o caso de *setup*independente da sequência de operações das tarefas, e treze heurísticas testadas para *setup*dependente.

Para a realização da experimentação computacional foi desenvolvido um software específico. Para avaliação do *software* foi gerado um conjunto de trinta e dois mil problemas-teste que são diferenciados em função do número de tarefas (n), número de estágios de produção (g), níveis de flexibilidade (f) e intervalos de tempo de setup (s). Na sequência o resultado destes problemas-testes são avaliados por algumas medidas estatísticas.

Palavras-chave: *Flow Shop, No-Wait*.

ABSTRACT

The study deals with the problems of Production Scheduling in Flow Shop environment with multiple machines in the production stages, where the setup time is not inserted in the processing time of the operations. May be a production system in which the setup depends on whether or not the sequence of tasks, where tasks are performed without interruption between successive operations, also called production system no-wait. Being proposed in this work, twenty methods of heuristic solutions defined by Priority Rules, which provide an ordering of the tasks to be followed in its programming, which is done each task, successively.

Twenty of these proposed heuristics, seven were implemented for the case of set operations independent of the sequence of tasks, and thirteen dependent heuristics tested for setup.

To perform the computational experience we developed a specific software. For evaluation of the software generated a set of thirty-six thousand test-problems that are differentiated according to the number of tasks (n), number of production stages (g), levels of flexibility (f) and time intervals of setup (s). Following the result of these problems-tests are evaluated by some statistical measures.

Keywords:*Flow Shop, No-Wait.*

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SÍMBOLOS

b_{jqk}	Data de início da tarefa *j*, processada pela máquina *q* do estágio *k*.
C_{jqk}	Data de término da tarefa *j*, processada pela máquina *q* do estágio *k*.
C_{qk}	Carga da máquina *q* do estágio *k* para um determinado instante do processo de programação das tarefas. A carga de uma máquina *q* é o maior valor até então atingido pela variável L_{iqk}.
f	Nível de flexibilidade.
g	Número de estágios de produção.
L_{iqk}	Data de liberação da máquina *q*(*q* = 1, 2, ..., mk) do estágio *k*, após o processamento da tarefa $i \in \{1, 2, ..., n\}$.
LP	Regra de Prioridade que ordena as tarefas de acordo com a ordem não crescente dos tempos totais de processamento.
LPS	Regra de Prioridade que ordena as tarefas de acordo com a ordem não crescente dos tempos totais de processamento somados aos tempos de preparação das máquinas.
LP<u>S</u>	Regra de Prioridade que ordena as tarefas de acordo com a ordem não crescente da somatória para todos os estágios de produção dos tempos de processamento da tarefa *j* com as médias aritméticas dos tempos de setup para execução da tarefa *j* no estágio de produção.
LPS dep	Regra de Prioridade que ordena as tarefas de acordo com a ordem não crescente da somatória para todos os estágios de produção dos tempos de processamento da tarefa *j* com os maiores tempos de setup para execução da tarefa *j* em cada estágio de produção *k*.
LPs dep	Regra de Prioridade que ordena as tarefas de acordo com a ordem não crescente da somatória para todos os estágios de produção dos tempos de processamento da tarefa *j* com os menores tempos de setup para execução da tarefa *j* em cada estágio de produção *k*.
LS	Regra de Prioridade que ordena as tarefas de acordo com a ordem não crescente dos tempos totais de preparação das máquinas.
L<u>S</u>	Regra de Prioridade que ordena as tarefas de acordo com a ordem não crescente da somatória das médias aritméticas dos tempos de setup para execução da tarefa *j* no estágio de produção *k*.
mk	Número de máquinas paralelas idênticas do estágio *k*, sendo determinado pelo nível de flexibilidade *f*.

n	Número de tarefas.
P_j	Somatória dos tempos de processamento da tarefa *j* em todos os estágios de produção.
P_{jk}	Tempo de processamento da tarefa *j* no estágio de produção *k*.
PS_j	Somatória dos tempos de processamento da tarefa *j* com os tempos de preparação para execução da tarefa *j*, em todos os estágios de produção.
$\underline{PS}_j$	Somatória para todos os estágios de produção dos tempos de processamento da tarefa *j* com as médias aritméticas dos tempos de setup para execução da tarefa *j* no estágio de produção *k*.
RAND	Regra de Prioridade que ordena aleatoriamente as tarefas. Esta regra de prioridade tem o objetivo de avaliar comparativamente o desempenho das demais regras que são "estruturadas".
R_{jqk}	Data de término da preparação (setup) da máquina *q* do estágio *k*, para execução da tarefa *j*.
S_{ijk}	Tempo de preparação de alguma máquina do estágio de produção *k* para execução da tarefa *j* , a qual é precedida diretamente pela tarefa *i*.
S_j	Somatória dos tempos de preparação das máquinas em todos os estágios de produção para execução da tarefa $_j$.
$\underline{S}_j$	Somatória das médias aritméticas dos tempos de setup para execução da tarefa *j* no estágio de produção *k*.
S_{jk}	Tempo de preparação das máquinas paralelas idênticas do estágio de produção *k* para execução da tarefa *j*.
$\overline{S}_{jk}$	Média aritmética dos tempos de setup no estágio de produção *k*, considerando todas as (n-1) tarefas *i* que podem preceder diretamente a tarefa *j*, incluindo o tempo de setup quando a tarefa *j* for a primeira a ser processada pela máquina.
SP	Regra de Prioridade que ordena as tarefas de acordo com a ordem não decrescente dos tempos totais de processamento
SPS	Regra de Prioridade que ordena as tarefas de acordo com a ordem não decrescente dos tempos totais de processamento somados aos tempos de preparação das máquinas.
SPS dep	Regra de Prioridade que ordena as tarefas de acordo com a ordem não decrescente da somatória para todos os estágios de produção dos tempos de processamento da tarefa *j* com os maiores tempos de setup para execução da tarefa *j* em cada estágio de produção *k*.

SPs dep	Regra de Prioridade que ordena as tarefas de acordo com a ordem não decrescente da somatória para todos os estágios de produção dos tempos de processamento da tarefa *j* com os menores tempos de setup para execução da tarefa *j* em cada estágio de produção *k*.
SPS	Regra de Prioridade que ordena as tarefas de acordo com a ordem não decrescente da somatória para todos os estágios de produção dos tempos de processamento da tarefa *j* com as médias aritméticas dos tempos de setup para execução da tarefa *j* no estágio de produção *k*.
SS	Regra de Prioridade que ordena as tarefas de acordo com a ordem não decrescente dos tempos totais de preparação das máquinas
SS	Regra de Prioridade que ordena as tarefas de acordo com a ordem não decrescente da somatória das médias aritméticas dos tempos de setup para execução da tarefa *j* no estágio de produção.

SUMÁRIO

CAPÍTULO 1 - INTRODUÇÃO

Neste capítulo, é apresentado o projeto da Dissertação de Mestrado, cuja estrutura é dividida em cinco seções, explicitadas a seguir. Na primeira seção, são comentadas algumas considerações iniciais sobre o problema de programação da produção. Na segunda seção são apresentados a hipótese, a relevância do tema e a definição do problema da pesquisa. Na terceira seção, são apresentados os objetivos da dissertação, geral e específico. Na quarta seção, apresenta-se de forma resumida a metodologia do presente trabalho. Por fim, na quinta seção, é apresentada a estrutura da Dissertação, descrevendo-se, sucintamente, os conteúdos de seus capítulos.

1.1. Considerações iniciais

A pesquisa operacional, apresenta relevância crescente para diversos segmentos industriais, visto que, em decorrência da globalização um diferencial para produto(s) e/ou serviço(s) tornou-se além da qualidade, o tempo.

Neste sentido, a aplicação de algoritmos de sequenciamento para resolução de problemas da programação da produção é bastante útil, pois pode permitir a eliminação e/ou diminuição de tempo ocioso de máquina após realizado o *setup*, dentre outras vantagens.

Compreende-se por programação da produção, segundo Moccelin (2005) como um problema de n tarefas {J1, J2,J3,...,JN} que devem ser processadas em m máquinas {M1,M2,M3,...MM} que estão disponíveis. O processamento de uma tarefa Jj em uma máquina Mk é denominado operação (opkj). Existe um tempo de processamento pkj associado a cada operação. Cada tarefa Jj possui uma data de liberação Ij (release date), a partir da qual a tarefa pode ser executada e uma data de entrega dj (due date), referente à data em que a tarefa deve estar concluída.

A complexidade de tais processos produtivos, pode ser aumentada, tornando-o um problema não básico, quando restrições adicionais são acrescentadas ao problema tradicional. Por exemplo: considerando processamento contínuo; tempo de *setup* não incluso no tempo de processamento das máquinas, dentre outras situações.

Usualmente, os problemas de programação de tarefas em sistemas de produção são classificados em função do fluxo das operações nas máquinas, conforme apresentado na sequência:

Tabela 1. Classificação dos problemas de programação da produção.

Classificação	**Descrição**
Job Shop	Cada tarefa apresenta sua própria sequência de processamento no conjunto de máquinas.
Flow Shop	Todas as tarefas possuem a mesma sequência no conjunto de máquinas.
Open Shop	Não existe uma sequência específica ou preestabelecida para o processamento das tarefas nas máquinas.
Flow Shop Permutacional	É um Flow Shop no qual em cada máquina a sequência das tarefas é a mesma.
Máquina Única	Existe uma única máquina disponível para o processamento das tarefas.
Máquinas Paralelas	São disponíveis diversas máquinas para o processamento das tarefas.
Job Shop com Máquinas Múltiplas	É um Job Shop no qual existe um conjunto de máquinas paralelas em cada estágio de produção.
Flow Shop com Máquinas Múltiplas ou Flow Shop Híbrido (FSH)	Consiste em um *Flow Shop* no qual existe um conjunto de máquinas paralelas em cada estágio de produção.

Fonte: Adaptado pelo autor, baseado no artigo dos autores NAGANO, M.S., MOCCELIN, J.V. e LORENA, L.A.N. intitulado como: Programação da Produção Flow Shop Permutacional com minimização do tempo médio de fluxo. Disponível em: <http://www.lac.inpe.br/~lorena/nagano/sbpo-nagano-mocellin-lorena.pdf>. Acesso em: 20 de Junho de 2013.

Para a última classificação apresentada, *flow shop* híbrido (FSH), problema abordado nesta dissertação, é considerando o tempo de preparação das máquinas não incluídos no tempo de processamento das tarefas, considerando esta hipótese, existe pouca abordagem na literatura.

Contudo, neste estudo é considerado FSH, e além de tempo de *setup* distinto do tempo de processamento das tarefas, também é considerado processamento contínuo, considerando o problema com tais características, não foi encontrado ainda referência na literatura. Entretanto, o avanço obtido em sistemas *Flow Shop* Híbridos com tempos explícitos de *setup*, aliado ao ainda recente avanço de sistemas contínuos nas mesmas condições quanto às

atividades de preparação das máquinas, permite inferir que o problema deve já estar sendo estudado pelos pesquisadores da área de Programação da Produção.

1.2 Problema e hipótese

A configuração menos complexa do problema Flow Shop Híbrido (FSH), configuração básica, considera os tempos de preparação das máquinas incluídos nos tempos de processamento das operações. Essa configuração tem recebido uma considerável atenção por parte de muitos pesquisadores.

Esta consideração certamente simplifica a análise e/ou reflexos em muitas aplicações. Porém, tal hipótese pode afetar de forma relevante a qualidade da solução para muitas aplicações que requerem tratamento explícito do tempo de preparação das máquinas.

Geralmente essa agregação dos tempos (preparação e processamento) pode ser aceita quando os tempos de preparação forem independentes da sequência de operações ou quando a variabilidade não for significativa. Mesmo assim, a agregação leva a um modelo de programação da produção onde uma máquina qualquer somente será preparada para uma tarefa após o término da operação anterior dessa tarefa, mesmo que a referida máquina já esteja liberada (ou seja, já tenha processado a operação anterior).

Nesta pesquisa, o problema tratado apresenta características não consideradas no modelo FSH tradicional, além de um grau de complexidade substancialmente maior, uma vez que:

i. Os tempos de preparação das máquinas são explicitamente separados dos tempos de processamento das tarefas, podendo ser dependentes ou não da sequência de execução das tarefas.
ii. O sistema produtivo é do tipo contínuo, ou seja, uma tarefa uma vez iniciada, é executada continuamente através dos estágios de produção, não havendo portanto tempo de espera entre operações sucessivas.

1.2.1 Definição do problema de pesquisa

O presente estudo aborda problemas de Programação da Produção em ambiente *Flow Shop* com máquinas (células / instalações / equipamentos) múltiplas nos estágios de produção,

considerando as hipóteses que os tempos de preparação das máquinas não estão incluídos nos tempos de processamento das operações, podendo ser dependente, ou não, das relações de precedência entre as tarefas produtos nas máquinas, para sistemas produtivos nos quais as tarefas devem ser executadas sem interrupção entre operações sucessivas, também conhecidos como sistemas de produção contínua.

Existe necessidade de aprimoramento deste tipo de algoritmos capazes de garantir maior rapidez nos fluxos produtivos, pois permitem a diminuição e/ou eliminação de tempos ociosos no processo. Para tanto, é preciso verificar qual heurística apresenta melhor desempenho para os casos estudados. Com base no que foi a apresentado, levanta-se o seguinte questionamento: Considerando FSH, processamento contínuo e tempo de preparação das máquinas distinto do tempo de processamento das tarefas. Qual a melhor heurística para *setup* dependente? Qual a melhor heurística para *setup* independente da sequência de tarefas?

1.3 Objetivos da pesquisa

1.3.1 Objetivo Geral

O objetivo principal deste estudo, é a comparação em termos da qualidade da solução obtida, neste caso minimização do tempo total de programação (*makespan*), dos métodos heurísticos utilizados.

1.3.2 Objetivo Específico

Para atingir ao objetivo maior, foram estabelecidos os seguintes objetivos específicos:

i. Caracterizar de forma clara o Problema da Produção *Flow Shop* com Máquinas Múltiplas, também conhecido como *Flow Shop* Híbridos (FSH);

ii. Implementar vinte heurísticas para FSH. Sendo sete heurísticas para setup independente e treze para setup dependente, e considerando em todos os casos o processamento continuo das tarefas e tempo de preparação das máquinas não incluído no tempo de processamento das tarefas; e

iii. Avaliar os resultados obtidos da experimentação computacional através de medidas estatísticas para todas as regras de prior

1.4 Metodologia

Para atingir aos objetivos deste estudo, houve realização de experimentação computacional, para isto sendo desenvolvido um software específico. Sete heurísticas foram implementadas para o caso de *setup* independente da sequência de operações das tarefas, e treze heurísticas testadas para *setup* dependente. Em todos os casos, foram consideradas as hipóteses de: *flow shop* híbrido, processamento contínuo das tarefas, e tempo de preparação das máquinas não incluído no tempo de processamento das tarefas.

Para avaliação do software foi gerado um conjunto de trinta e seis mil (36.000) problemas-teste que serão diferenciados em função do número de tarefas (n), número de estágios de produção (g), níveis de flexibilidade (f) e intervalos de tempo de setup (s).

1.5 Estrutura da dissertação

Além deste capítulo de introdução, que evidencia o problema da pesquisa, a metodologia e seus objetivos, geral e específicos, a dissertação está estruturada como apresentado a seguir. No capítulo 2 é apresentado a revisão da literatura, na qual o problema investigado é adequadamente situado quanto ao estado da arte.

No capítulo 3, são apresentados os métodos que estão sendo propostos para solução do problema.

No capítulo 4 é realizada a explicação do algoritmo utilizado para resolução do problema de sequenciamento da produção. Sendo este problema, flow shop híbrido com processamento contínuo das tarefas e tempo de setup não incluído no tempo de processamento. Podendo ser o caso de setup dependente ou independente da sequência de tarefas.

No capítulo 5, se inicia com a demonstração do *software* desenvolvido, dando continuidade foi feito a comparação dos métodos heurísticos propostos em termos da qualidade da solução obtida, neste caso o critério considerado foi menor *makespan*. Para comparação destes resultados obtidos da experimentação computacional, foram utilizadas algumas medidas estatísticas

A última seção refere-se as principais conclusões encontradas dos resultados obtidos, e na sequência são apresentadas as referências bibliográficas consultadas.

CAPÍTULO 02 – REVISÃO DE LITERATURA

Neste capítulo, apresenta-se uma revisão da literatura, na qual o problema investigado é adequadamente situado quanto ao estado da arte . Para melhor compreensão da evolução da resolução do problema do sequenciamento da produção, este capítulo é dividido em cinco seções. As seções mencionam os trabalhos mais relevantes para resolução de disntintos problemas do sequenciamento da produção. Sendo estes: *flow shop* híbrido básico; *flow shop* híbrido com *setups* separados; *flow shop no-wait* básico ; *flow shop* híbrido no-wait básico e *flow shop* permutacional *no-wait* com *setups* separados, respectivamente.

2.1. *Flow Shop* Híbrido Básico (FSH)

Para este tipo de problema, Shen e Chen (1972) foram os pioneiros a tratar com mk > 2, k = 1, 2 (dois estágios). Eles apresentaram a proposta de um método heurístico permutacional com o objetivo de minimizar a duração do tempo total da programação, ou também conhecido como makespan.

Após as pesquisas pioneiras, muitas outras foram desenvolvidas e reportadas na literatura acadêmica, dentre os quais: Hunsucker et al. (1989), Guinet et al. (1992), Hunsucker e Shah (1994), Vignier et al. (1995), Portmann et al. (1996), e Pocket e Moursli (2000).

Oguz et al. (2003, 2004), trataram do *flow shop* híbrido básico no caso em que uma mesma operação pode ser subdividida em operações com processamento simultâneo.

De forma similar aos resultados apresentadospor Oguz et al. (2003, 2004), nos quais uma mesma operação pode ser subdividida em operações com processamento simultâneo, Ying (2009) desenvolveu um método de solução heurística gulosa para a mesma categoria de problema. E através de experimentação computacional, teve êxito em mostrar a superioridade do método proposto quando comparado com as melhores metaheurísticas até então reportadas na literatura.

Já Choong, Phon-Amnuaisuk e Alias (2011) desenvolveram novas metaheurísticas, os denominados Algoritmos Meméticos. Nesse artigo, os autores apresentam métodos heurísticos utilizando ao mesmo tempo *Simulated Annealing*, Busca Tabu e a chamada *Particle Swarm Optimization* (PSO) introduzida por Kennedy e Eberhart (1995).

2.2. *Flow Shop* Híbrido com setups separados

Nesse ambiente de produção, o *flow shop* híbrido com tempos de *setup* separados, a literatura é relativamente recente. Mas são trabalhos com extrema importância, visto que a inclusão do tempo de preparação das máquinas no tempo total de processamento, simplifica a análise na maioria dos casos, mas pode afetar de forma significativa a qualidade da solução encontrada. Pois ocorre a situação onde uma máquina qualquer somente será preparada para uma tarefa após o término da operação anterior dessa tarefa, mesmo que a referida máquina já esteja liberada (ou seja, já tenha processado a operação anterior).

Um dos primeiros trabalhos relevantes foi desenvolvido por Huang e Li (1998) que apresentaram um ambiente *flow shop* híbrido com dois estágios e tempos de *setup* dependentes da sequência de famílias de produtos.

Fuchigami (2005) desenvolveu e avaliou o desempenho de quatro métodos heurísticos construtivos para o *flow shop* híbrido geral com tempos de *setup* dependentes em todos os estágios de produção. Os métodos tiveram como base algoritmos reportados na literatura para solução de problemas *flow shop* permutacional (Simons Jr., 1992) e máquinas paralelas (Weng, Lu e Ren, 2001).

Tang e Zhang (2005) propuseram uma nova heurística combinada com método de redes neurais artificiais para o problema com tempos de *setup* dependentes da seqüência.

Heurísticas construtivas para minimização do *makespan* foram apresentadas por Logendran, Carson e Hanson (2005). As tarefas foram agrupadas em famílias e foram considerados tempos de *setup* dependentes das máquinas e independentes das famílias de tarefas.

Ruiz e Maroto (2006) construíram uma metaheurística baseada em um algoritmo genético, para problemas complexos de programação *flow shop* híbrido.

O estudo da influência da programação do primeiro estágio para problemas com tempos de *setup* independentes da sequência foi realizado por Fuchigami e Moccellin (2006). A pesquisa comprovou a grande relevância do primeiro estágio para a minimização do *makespan*.

Fuchigami, Moccellin e Ruiz (2007) compararam o desempenho de regras de prioridade em sistemas FSH Flexíveis com tempos de setup independentes da sequência. O estudo identificou as regras mais eficientes para a minimização do *makespan*. A mesma

análise foi feita também para o ambiente *flow shop* híbrido não-flexível por Fuchigami e Moccellin (2007), acrescentando duas regras de prioridade aleatórias na comparação.

Ruiz, Şerifoğlu e Urlings (2008) abordaram um problema de FSH Flexível com máquinas paralelas não relacionadas e tempos de *setup* dependentes da seqüência. Várias outras restrições foram consideradas tais como: presença de tempos de *setup* antecipado e não antecipado, diferentes datas de liberação das tarefas, elegibilidade de máquina e restrições de precedência entre as tarefas.

Jungwattanakit et al. (2009) efetuaram uma comparação de métodos heurísticos para o problema FSH com máquinas paralelas não-relacionadas, tempos de preparação das máquinas dependentes das próprias máquinas e da sequência de processamento das tarefas.

2.3. *Flow Shop No-wait* Básico

Esse tipo de sistema produtivo pode existir tanto em Job Shops quanto em Flow Shops. A maioria dos trabalhos reportados na literatura acadêmica consideram os modelos convencionais do Flow Shop nos quais os tempos depreparação das máquinassão incluídos nos tempos de processamento das tarefas, além de uma única máquina em cada estágio de produção. Os trabalhos mais relevantes são descritos de maneira resumida na sequência.

O problema de programação da produção em um flow shop no-wait foi inicialmente proposto por van Deman e Baker (1974). Sahni e Gonzales (1976) mostraram que tal problema é NP-hard.

Rajendran e Chaudhuri (1990) desenvolveram dois métodos heurísticos construtivos utilizando regras simples de prioridade.

Bertolissi (2000) apresentou um outro método heurístico construtivo que forneceu um melhor desempenho, quanto à qualidade da solução, ao ser comparado com um dos métodos de Rajendran e Chaudhuri (1990).

Aldowaisan e Allahverdi (2004) apresentaram seis novos métodos heurísticos. Eles foram inicialmente comparados entre si, sendo o melhor deles o denominado PH1(p). A seguir, este melhor método foi comparado com os dois métodos de Rajendran e Chaudhuri (1990) e também com um algoritmo genético desenvolvido por Chen *et al.* (1996).

Kumar *et al.* (2006) propuseram um algoritmo *Psycho-clonal* que apresenta resultados mais favoráveis do que o algoritmo genético de Chen *et al.* (1996) como também em relação ao conjunto de métodos apresentados por Aldowaisan e Allahverdi (2004).

Pan *et al*. (2007) propuseram uma otimização *Discrete Particle Swarm (DPSO)* com o objetivo de otimizar a duração total da programação (makespan) e o tempo total de fluxo.

Tseng e Lin (2010), à semelhança do artigo de Qian *et al.* (2009), propuseram um método metaheurístico combinando Algoritmo Genético com um novo procedimento de busca local em vizinhança, tendo como medida de desempenho o makespan. Por meio de uma extensa experiência computacional, sua qualidade de solução e eficiência computacional são avaliadas utilizando diversas bases de problemas-teste disponíveis na literatura.

2.4. *Flow Shop* Híbrido *No-wait* Básico

Para o ambiente *Flow Shop* Híbrido (FSH) sem consideração explícita dos tempos de preparação das máquinas, foram encontrados, no exame efetuado da literatura, somente dois trabalhos. No primeiro, proposto por Sriskandarajah (1993), são apresentados alguns métodos heurísticos e avaliados os respectivos desempenhos mínimos, de forma a poder se estabelecer desvios máximos porcentuais entre as soluções heurísticas e as soluções ótimas, tendo como objetivo a minimização do *makespan*, em sistemas *flow shop* híbridos com somente dois estágios de produção.

O outro trabalho, foi efetuado por Xie et al. (2004), e abordou o mesmo ambiente FSH do trabalho acima citado de Sriskandarajah (1993). Os autores propuseram um método heurístico de "Desvio Mínimo" (Minimum Deviation) e efetuaram uma comparação de desempenho com os métodos de Sriskandarajah (1993), apresentando melhores resultados na maioria dos problemas-teste da experimentação computacional.

Finalizando esta seção referente ao modelo convencional de *flow shop no-wait*, considera-se relevante destacar que tal modelo somente pode ser considerado adequado se o tempo de preparação das máquinas for suficientemente pequeno para não influir no processamento contínuo das tarefas.

Na verdade, modelos adequados seriam aqueles em que os tempos de preparação das máquinas são considerados separados dos tempos de processamento das tarefas e necessariamente envolvendo atividades antecipadas de *setup*. Ou seja, uma vez concluída a operação de uma tarefa em uma determinada máquina (estágio de produção), a próxima máquina já deverá estar preparada para execução da respectiva operação da tarefa, observando portanto o o fato de não haver tempo de espera entre operações sucessivas. Nesta pesquisa foi adequadamente considerado o *setup* antecipado, necessário nos sistemas de produção *no-wait*.

2.5. *Flow Shop* permutacional *no-wait* com *setups* separados

Com relação ao comentário acima, a literatura que trata de *flow shop no-wait* com tempos de *setup* separados dos tempos de processamento das tarefas, porém no ambiente *Flow Shop* Permutacional, ou seja, com apenas uma máquina em cada estágio de produção, é relativamente escassa.

Allahverdi e Aldowaisan (2000) consideraram o problema com tempos de *setup* independentes da seqüência de execução das tarefas, com o objetivo de minimizar o tempo médio de fluxo. O ambiente considerado era um *Flow Shop* Permutacional com apenas três estágios de produção. Os autores obtiveram a solução ótima para condições específicas dos tempos de processamento das tarefas e tempos de *setup*, estabelecendo uma condição de dominância. Além disso, foram propostos cinco métodos heurísticos que foram avaliados por meio de uma experimentação computacional.

Em 2001, os mesmos autores (Allahverdi e Aldowaisan) desta vez trataram do problema com tempos de setup dependentes da sequência de execução das tarefas, com o objetivo de minimizar o tempo médio de fluxo, para *flowshops* com apenas dois estágios de produção. Foram propostos diversos métodos heurísticos compostos de duas fases. Na primeira, uma sequência inicial das tarefas é obtida e na segunda fase é utilizado um procedimento iterativo de troca de posições e inserção de tarefas com o objetivo de melhorar a solução inicial. Resultados experimentais mostram que o procedimento de inserção de tarefas geralmente converge para a mesma solução após um pequeno número de iterações.

Com um interesse maior teórico do que prático, Stafford e Tseng (2002) apresentaram dois modelos de Programação Linear Inteira Mista para o problema *flow shop* permutacional com um número qualquer de estágios de produção, tendo como objetivo a minimização do *makespan*. Os tempos de *setup* foram considerados dependentes da sequência de execução das tarefas.

Brown *et al.* (2004) consideraram tempos de *setup* independentes da sequência de execução das tarefas. Foram desenvolvidos dois métodos heurísticos. O primeiro, denominado *TRIPS*, é um algoritmo construtivo (tempo de computação polinomial) adequado à minimização do *makespan*. O segundo fundamenta-se na metaheurística *Simulated Annealing* e foi concebido adequadamente para a minimização do tempo total de fluxo.

Para o caso particular de apenas dois estágios de produção, Shyu *et al.* (2004) apresentaram um procedimento de otimização "colônia de formigas" (*Ant Colony*), com o

objetivo de minimizar o tempo médio de fluxo, em um *flowshop* com tempos de *setup* independentes da sequência de execução das tarefas.

Mais recentemente, Ruiz e Allahverdi (2007) apresentaram sete novos métodos heurísticos para o problema *Flow Shop* Permutacional com múltiplos estágios de produção (número qualquer de máquinas), tempos de *setup* independentes da sequência de execução das tarefas e restrições de processamento das tarefas *no-wait*. Cinco deles são construtivos com alta eficiência computacional e os outros dois baseados no método "melhorativo" de Busca Local Aleatória.

O exame da literatura indica que o problema que está sendo investigado ainda não foi reportado na literatura. Entretanto, o avanço obtido em sistemas *Flow Shop* Híbridos com tempos explícitos de *setup*, aliado ao ainda recente avanço de sistemas *no-wait* nas mesmas condições quanto às atividades de preparação das máquinas, permite inferir que o problema deve já estar sendo estudado pelos pesquisadores da área de Programação da Produção.

CAPÍTULO 03 – MÉTODOS HEURÍSTICOS CONSTRUTIVOS PARA SOLUÇÃO DO PROBLEMA DE PROGRAMAÇÃO DA PRODUÇÃO TRATADO NESTA PESQUISA

Neste estudo são implementados vinte métodos heurísticos construtivos para a solução do problema de programação da produção, considerando máquinas paralelas em cada estágio, processamento contínuo e o tempo de *setup* não está incluído no tempo de processamento da tarefa.

Para a realização da experimentação computacional foi desenvolvido um *software* específico. Sendo sete heurísticas implementadas para o caso de *setup* independente da sequência de operações das tarefas, e treze heurísticas testadas para *setup* dependente.

Para avaliação do *software* será gerado um conjunto de problemas-teste que serão diferenciados em função do número de tarefas (n), número de estágios de produção (g), níveis de flexibilidade (f) e intervalos de tempo de setup (s).

O presente trabalho propõe métodos de soluções heurísticas definidos por Regras de Prioridade, as quais fornecem uma ordenação das tarefas a ser seguida na sua programação, que é feita tarefa por tarefa, sucessivamente.

A pesquisa com regras de prioridade em ambientes complexos de programação da produção, como no caso deste trabalho, torna-se relevante e significativa. Uma vez que, em muitas situações, as soluções obtidas pelas regras de prioridade podem ser usadas como solução inicial para métodos de busca em vizinhança com o objetivo de eventualmente obter melhores soluções.

Serão consideradas regras de prioridade para problemas com tempos de *setup* independentes da sequência de execução das tarefas nas diversas máquinas, como também para problemas com tempos de *setup* dependentes.

3.1 Regras de Prioridade para *setup* independente

Para obter a solução do problema de programação com *setup* independente da sequência de execução das tarefas, foram definidas sete regras de ordenação para as tarefas. Tais propostas baseiam-se nas bem conhecidas regras SPT e LPT, as quais têm sido utilizadas

em muitos trabalhos reportados na literatura e mostrado sua eficácia em termos de qualidade de solução, como também altamente eficientes quanto ao tempo de computação.

Como se sabe, a regra SPT sequencia as tarefas pela ordem não decrescente dos tempos de processamento das tarefas, enquanto que a LPT utiliza a ordem não crescente desses tempos. Para o problema de máquina única, a regra SPT minimiza, de maneira ótima, o tempo médio de fluxo e para o ambiente com máquinas paralelas, a regra LPT tende a equilibrar a carga de trabalho das máquinas, procurando minimizar a duração total da programação (*makespan*). Deve-se salientar que neste trabalho a medida de desempenho adotada é o *makespan*.

Antes de definir as regras de prioridade é necessário introduzir três conceitos que serão utilizados por estas. O primeiro conceito importante é o valor do tempo de processamento total de uma tarefa, desprezando os tempos de *setup*. O valor do tempo de processamento total sem setups para uma tarefa *j* (P_j) é calculado como,

$$P_j = \sum_{k=1}^{g} p_{jk} \tag{1}$$

onde p_{jk} é o valor do tempo de processamento da tarefa *j* no estágio *k*, e *g* é o número de estágios.

Analogamente, é computado também o valor do tempo de *setup* total, desconsiderando o tempo de processamento, para uma dada tarefa. O tempo de *setup* total sem considerar processamento para uma tarefa *j* (S_j) é calculado como,

$$S_j = \sum_{k=1}^{g} s_{jk} \tag{2}$$

onde s_{jk} é o tempo de *setup* da tarefa *j* no estágio *k*.

Por fim, o terceiro conceito é o do tempo máximo possível de duração de uma tarefa. Este último consiste em considerar todos os tempos de processamento e *setup*. Para uma tarefa *j* o tempo máximo que poderia ser obtido será,

$$PS_j = \sum_{k=1}^{g} \left(p_{jk} + s_{jk} \right) \quad (3)$$

Baseado nas Eqs. (1) a (3), seis regras de prioridade foram desenvolvidas. Uma sétima regra também é incluida, porém esta regra considera a escolha aleatória de uma ordenação e portanto não está baseada nos conceitos propostos. As regras de prioridade são descritas a seguir

Regra 1) **LP** : ordena as tarefas de acordo com a ordem não crescente dos tempos totais de processamento, P_j.

Regra 2) **LS** : ordena as tarefas de acordo com a ordem não crescente dos tempos totais de preparação das máquinas, S_j.

Regra 3) **LPS** : ordena as tarefas de acordo com a ordem não crescente dos tempos totais de processamento somados aos tempos de preparação das máquinas, PS_j.

Regra 4) **SP** : ordena as tarefas de acordo com a ordem não decrescente dos tempos totais de processamento, P_j.

Regra 5) **SS** : ordena as tarefas de acordo com a ordem não decrescente dos tempos totais de preparação das máquinas S_j .

Regra 6) **SPS** : ordena as tarefas de acordo com a ordem não decrescente dos tempos totais de processamento somados aos tempos de preparação das máquinas PS_j.

Regra 7) **RAND** : ordena aleatoriamente as tarefas. Esta regra de prioridade tem o objetivo de avaliar comparativamente o desempenho das demais regras que são "estruturadas".

3.2 Regras de Prioridade para *setup* dependente

Antes de apresentar as regras de prioridade que serão usadas para esta classe de problemas será necessário definir algumas relações importantes.

Começaremos pela média aritmética dos tempos de *setup* de um dado estágio *k* para uma tarefa *j* considerando as n tarefas que podem ser precedidas:

$$\bar{S}_{jk} = \frac{1}{n}\sum_{i=1}^{n} s_{ijk} \tag{4}$$

A somatória das médias aritméticas apresentadas na Eq. (4) de todos os estágios:

$$\underline{S}_j = \sum_{k=1}^{g} \bar{S}_{jk} \tag{5}$$

O somatório para todos os estágios dos tempos de processamento e *setup* médios obtidos pela Eq. (4):

$$P\underline{S}_j = \sum_{k=1}^{g} \left(p_{jk} + \bar{S}_{jk} \right) \tag{6}$$

O maior tempo de *setup* para o processamento de uma tarefa *j* em um estágio *k*.

$$S_{jk} = \max_i \left(s_{ijk} \right) \tag{7}$$

O menor tempo de *setup* para o processamento de uma tarefa *j* em um estágio *k*.

$$s_{jk} = \min_i \left(s_{ijk} \right) \tag{8}$$

O somatório para todos os estágios dos tempos de processamento e *setup* máximos obtidos pela Eq. (7):

$$PS_{\max_j} = \sum_{k=1}^{g} \left(p_{jk} + S_{jk} \right) \tag{9}$$

O somatório para todos os estágios dos tempos de processamento e *setup* mínimos obtidos pela Eq. (8):

$$PS_{\min_j} = \sum_{k=1}^{g} \left(p_{jk} + s_{jk} \right) \tag{10}$$

Baseado na Eq. (1) e nas Eqs. (4) a (10) podemos apresentar as regras prioridade. As regras de prioridade são listadas a seguir:

Regra 1) **LP** : ordena as tarefas de acordo com a ordem não crescente dos tempos totais de processamento, P_j.

Regra 2) **L$\underline{S}$** : ordena as tarefas de acordo com a ordem não crescente da somatória das médias aritméticas dos tempos de *setup* para o processamento de uma tarefa j, $\underline{S_j}$.

Regra 3) **LP$\underline{S}$** : ordena as tarefas de acordo com a ordem não crescente dos valores de $P\underline{S_j}$.

Regra 4) **LPS** : ordena as tarefas de acordo com a ordem não crescente dos valores de $PSmax_j$.

Regra 5) **LPs** : ordena as tarefas de acordo com a ordem não crescente dos valores de $PSmin_j$.

Regra 6) **SP** : ordena as tarefas de acordo com a ordem não decrescente dos tempos totais de processamento, P_j.

Regra 7) **S$\underline{S}$** : ordena as tarefas de acordo com a ordem não decrescente da somatória das médias aritméticas dos tempos de *setup* para o processamento de uma tarefa j, $\underline{S_j}$.

Regra 8) **SPS** : ordena as tarefas de acordo com a ordem não decrescente dos valores de PS_j.

Regra 9) **SPS** : ordena as tarefas de acordo com a ordem não decrescente dos valores de $PSmax_j$.

Regra 10) **SPs** : ordena as tarefas de acordo com a ordem não decrescente dos valores de $PSmin_j$.

Regra 11) **RAND** : ordena aleatoriamente as tarefas. Esta regra de prioridade tem o objetivo de avaliar comparativamente o desempenho das demais regras que são "estruturadas".

CAPÍTULO 4. ALGORITMOS PARA SOLUÇÃO DO PROBLEMA TENDO COMO BASE UMA ORDENAÇÃO DAS TAREFAS

A partir de uma ordenação estabelecida para as tarefas, utilizando uma determinada regra de prioridade, a solução do problema (programação das *n*tarefas do FSH) é obtida utilizando os algoritmos definidos a seguir. Esses algoritmos foram especificamente desenvolvidos tendo em vista as características do FSH com tempos de preparação das máquinas separados dos tempos de processamento das tarefas (preparação antecipada) e condição *no-wait* na execução das tarefas.

4.1 Algoritmo I - regras de prioridade para *setup* independente

Para este algoritmo os dados que devem ser fornecidos ao resolvedor são listados na Tabela 2.

Tabela 2 – Dados de entrada para o algoritmo I.

Dado de entrada	**Descrição**
N	Número de tarefas.
G	Número de estágios de produção.
m_k	Número de máquinas em cada estágio.
$ORP_v = \{J_{[1]}, J_{[2]},..., J_{[u]},..., J_{[n]}\}$	Ordenação das n tarefas. Cada J recebe o indíce de uma tarefa.
p_{jk}	Tempo de processamento da tarefa j no estágio k.
s_{jk}	Tempo de *setup* da tarefa j no estágio k.

Para o desenvolvimento do algorítimo são definidas variáveis de programação das tarefas. A data de liberação das máquinas é uma dessas variáveis. A data de liberação é definida como uma função da tarefa, estágio e máquina no estágio. Esta representa o momento em que uma máquina em um determinado estágio é liberada de uma tarefa, estando portanto, livre para executar uma nova atividade. A data de liberação de uma máquina q do estágio k após o processamento de uma tarefa j é escrita como L_{jqk}.

A carga de uma máquina é definida em um determinado estágio do processo e representa a carga de trabalho, em tempo, que a máquina deve processar. A carga da máquina *q* para o estágio *k* é escrita como,

$$C_{qk} = \max_{j} L_{jqk} \tag{11}$$

A data de término da preparação (*setup*) da máquina *q* do estágio *k* para a execução de uma tarefa *j* é escrita como,

$$R_{jqk} = C_{qk} + s_{jk} \tag{12}$$

A Eq. (12) é definida de modo a impor a condição *no-wait* no processo.

A data de início de uma tarefa *j* em uma máquina *q* do estágio *k* é escrita como b_{jqk}.

O algorítmo I é escrito nos passos abaixo abaixo:

1) Faça todos os valores de *L*, *R*, *b* e *C* zero;
2) Faça $u = 1$;
3) Obtenha o índice da tarefa *j* pela ordenação: $J_{[u]}$;
4) Faça $k = 1$;
5) Calcule os valores da carga para cada máquina *q* (C_{qk}) segundo a Eq. (11);
6) Baseado na carga do passo 5 escolha a máquina de menor carga *q**;
7) Calcule os valores de *R* para as m_k máquinas: Eq. (12);
8) Se $k = 1$, calcule o valor da data de início como: $b_{jq^*1} = R_{jq^*k}$. Se k>1, calcule a da de início de processamento como: $b_{jq^*k} = b_{jq^*k-1} + p_{jk-1}$.
9) Calcule a data de liberação como: $L_{jq^*k-1} = b_{jq^*k}$;
10) Viabilização: Se $b_{jq^*k} \geq R_{jq^*k}$ vá para o passo 12. Senão vá para o passo 11;
11) Utilizando a máquina *q** do estágio 1 faça: $b_{jq^*1} = b_{jq^*1} + 1$. Faça $k = 2$ e volte para o passo 5;
12) Se $k<g-1$ faça: $k = k+1$ e retorne ao passo 5. Caso contrário execute o passo 13;
13) Se $u<n$ faça: $u = u+1$ e retorne ao passo 3. Caso contrário execute o passo 14;
14) Calcule o *Makespam* como: $M = \max_{q} C_{qg}$.

O algoritmo I é resumido na Figura 1.

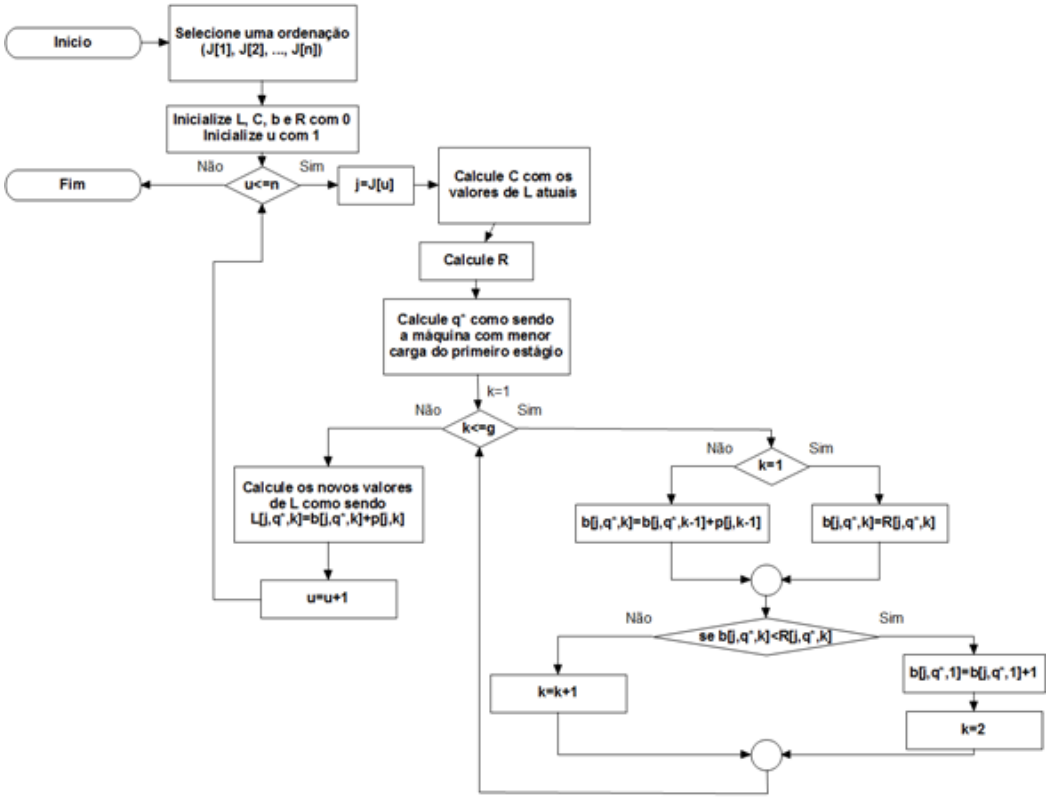

Figura 1 – Fluxograma do algoritmo I.

Um *software* escrito na linguagem *Object* Pascal e a IDE Embarcadeiro *Delphi* 2010 foi desenvolvido para a implementação das heurísticas e dos algoritmos.

4.2 Algoritmo II - regras de prioridade para *setup* dependente

Para o caso onde os tempos de preparo das máquinas de um dado estágio dependem da tarefa previamente processada naquele estágio, os dados fornecidos ao resolvedor serão os mesmos apresentados na Tabela 2 com uma única diferença: agora uma nova "coordenada" é incluida ao tempo de *setup*, ou seja, a tarefa processada no estágio imediatamente antes da tarefa a ser processada. O termo s_{jk} é reescrito como s_{ijk}, onde k é o estágio de produção, i é a tarefa anteriormente processada no estágio k e j a tarefa a ser processada no estágio k.

A Eq. (11) continua válida, porém a Eq. (12) deve ser reescrita como,

$$R_{jqk} = C_{qk} + s_{ijk} \tag{13}$$

O algoritmo para a resolver a programação é bastante semelhante ao previamente apresentado para *setup* independente, mas irá diferir sempre que o tempo de *setup* for utilizado. A escolha da máquina a ser utilizada q^* também será alterada. Por completeza, o algoritmo II é totalmente reecrito a seguir,

1) Faça todos os valores de L, R, b e C zero;
2) Faça $u = 1$;
3) Obtenha o índice da tarefa j pela ordenação: $J_{[u]}$;
4) Faça $k = 1$;
5) Calcule os valores da carga para cada máquina q (C_{jqk}) segundo a Eq. (11);
6) Calcule os valores de R para as m_k máquinas: Eq. (13);
7) Baseado nos valores de R do passo 6 escolha a máquina de menor valor q^*;
8) Guarde o índice da tarefa que será executada na máquina q^* para a programação das próximas tarefas em a_{kq}.
9) Se $k = 1$, calcule o valor da data de início como: $b_{jq^*1} = R_{jq^*k}$. Se k>1, calcule a da de início de processamento como: $b_{jq^*k} = b_{jq^*k-1}+p_{jk-1}$.
10) Calcule a data de liberação como: $L_{jq^*k-1} = b_{jq^*k}$;
11) Viabilização: Se $b_{jq^*k} \geq R_{jq^*k}$ vá para o passo 13. Senão vá para o passo 12;
12) Utilizando a máquina q^* do estágio 1 faça: $b_{jq^*1} = b_{jq^*1}+1$. Faça $k = 2$ e volte para o passo 5;
13) Se $k<g-1$ faça: $k = k+1$ e retorne ao passo 5. Caso contrário execute o passo 14;
14) Se $u<n$ faça: $u = u+1$ e retorne ao passo 3. Caso contrário execute o passo 15;
15) Calcule o *Makespam* como: $M = \max_q C_{qg}$.

CAPÍTULO 05 – RESULTADOS E DISCUSSÃO

Neste capítulo serão apresentados os resultados obtidos neste projeto de pesquisa. O capítulo iniciará com a demonstração do *software* desenvolvido, uma vez que este também é um resultado desse trabalho. Em seguida será feita uma análise de performance das regras de prioridade empregadas no FSH de tempos de *setup* explícitos e independentes. Por fim, a mesma análise será realizada para *setup* dependentes.

5.1 Interface gráfica do *software* para setup independente

O software foi escrito em Pascal, utilizando o ambiente de desenvolvimento integrado (IDE) Delphi®. O programa foi escrito nessa interface para proporcionar melhor iteração *software*-usuário. A interface de abertura do programa é apresentada na Figura 2.

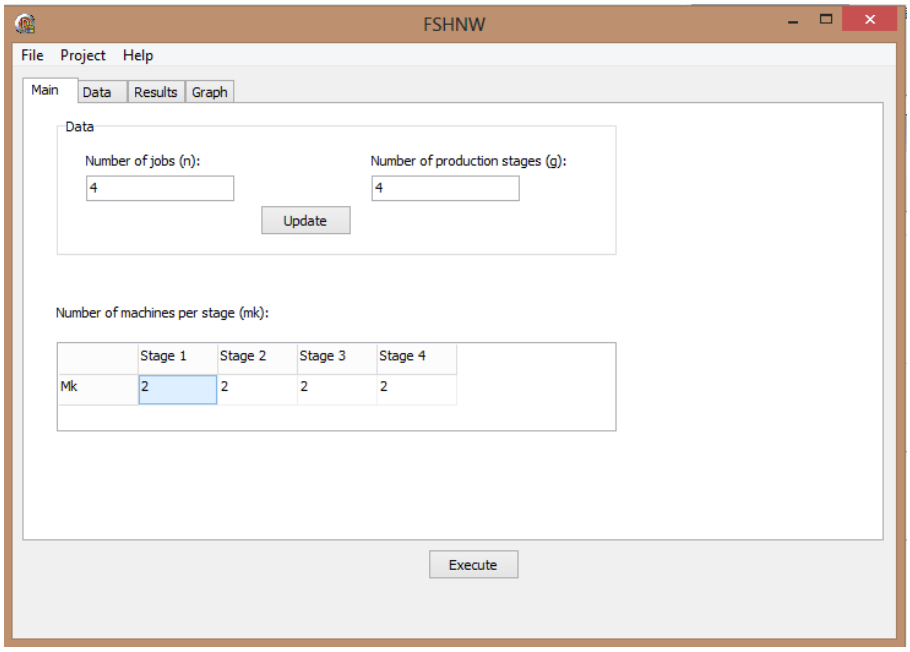

Figura 2 – Tela de abertura do programa.

Observe pela Figura 2 que o programa conta com menus, abas, e inúmeras outras características para auxiliar o seu uso. Também foi incluido a função de salvar e abrir o projeto. Essa funcionalidade foi incluida pois será de extrema importância.

Na Figura 3 o usuário pode fazer a inserção manual dos tempos de processamento das tarefas e tempos de preparação das máquinas e/ou carregar um arquivo de texto previamente salvo com estes valores.

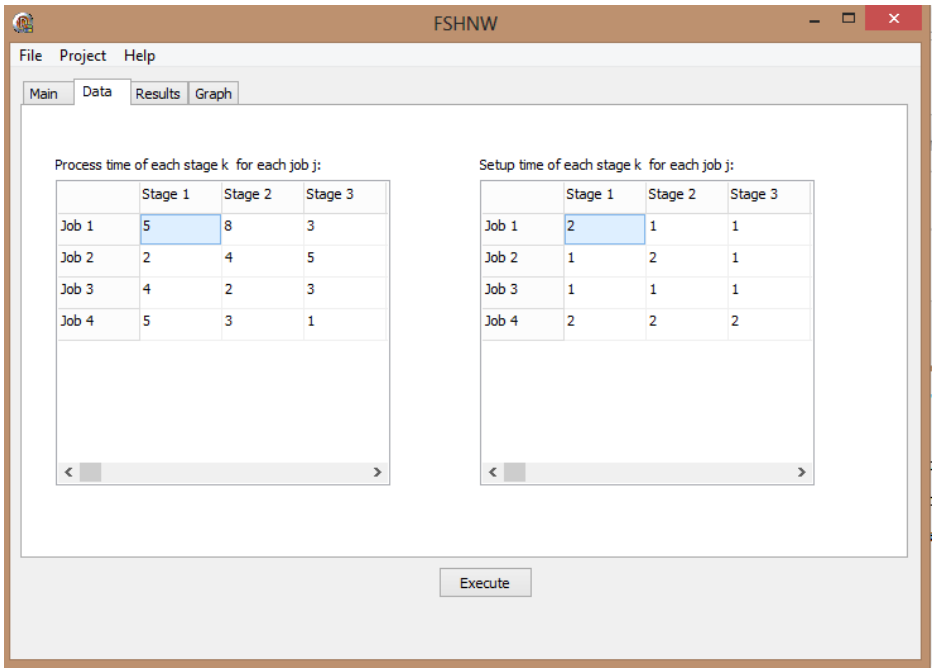

Figura 3 – Tela com os tempos de processamento das tarefas e *setup* das máquinas para o programa de *setup* independente.

Após o preenchimento da tela apresentada na Figura 3, o usuário do software deverá clicar em executar. Passando para a aba resultados, representada na Figura 4 pode-se observar que o *software* fornece a ordenação por regra de prioridade, assim como apresenta o *makespan* também por regra de prioridade.

Desta forma, permite de uma maneira rápida para a resolução de um problema-teste, por exemplo, identificar qual regra de prioridade apresentaria o melhor resultado.

Como já esperado, toda vez que o usuário clicar em executar, o resultado do *makespan* e ordenação das tarefas na regra de prioridade RAND, que é a regra de prioridade aleatória, irá mudar.

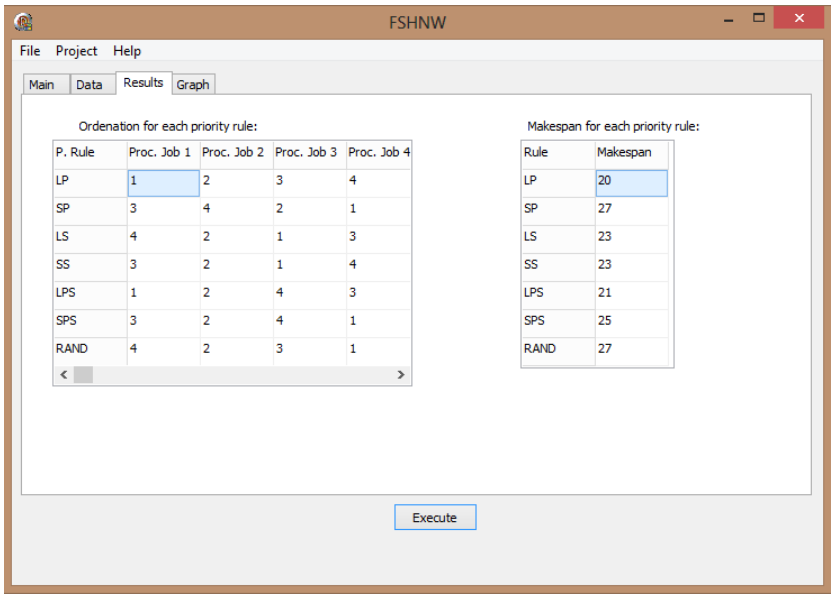

Figura 4 – Tela com os ordenações e *makespans* por regra de prioridade para o programa de *setup* independente.

No caso do problema-teste utilizado para explanação do *software* desenvolvido para o estudo desta dissertação, observa-se que a melhor regra de prioridade foi a LP (regra de prioridade que ordena as tarefas de acordo com a ordem não crescente dos tempos totais de processamento). Passando para a próxima aba do *software*, representada pela Figura 5 abaixo, pode-se visualizar o gráfico para a regra LP, bem como para as demais regras de prioridades.

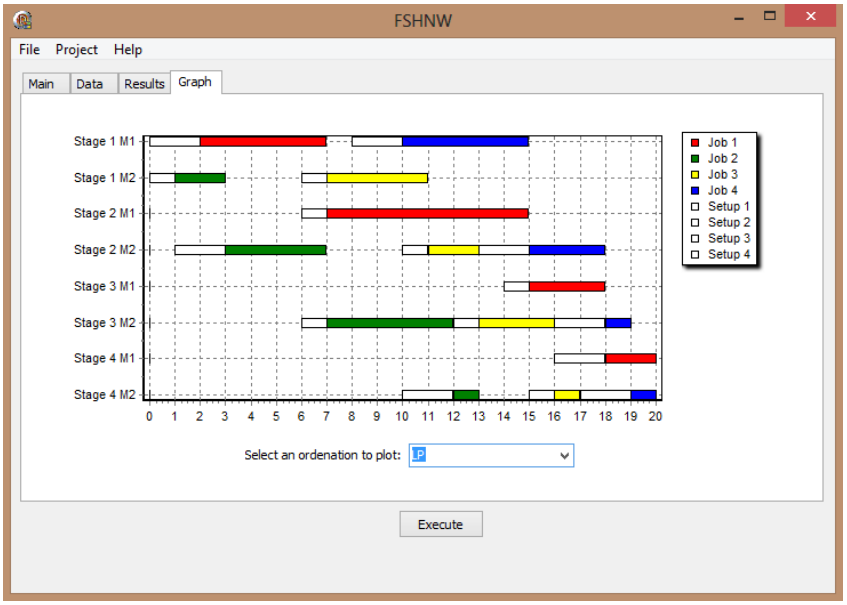

Figura 5 – Tela com Gráfico de *Gantt* para regra de prioridade LP.

No menu de ajuda aparece a tela representada pela Figura 6 com maiores informações sobre o programa desenvolvido.

Figura 6 – Tela com maiores informações do programa.

Conforme pode ser verificado, o *layout* deste *software* está bastante intuitivo para facilitar a utilização de qualquer usuário.

5.2 Interface gráfica do *software* para setup dependente

Por razões de simplicidade o código para tratar tempos de *setup* dependentes foi separado em um outro pacote. Esta versão do *software* é similar a desenvolvida para o *setup* independente tendo algumas diferenças na sua interface gráfica e no algoritmo de solução. Duas diferenças foram incluidas quanto a interface gráfica: entrada de dados de tempos de *setup* e processamento, e a tabela de resultados das heurísticas.

A Figura 7 apresenta a tela de entrada de dados para para tempos de processamento e *setup*. Como pode ser observado da figura, a entrada de dados para os tempos de processamento não foi alterada. A entrada de dados para tempos de processamento, por outro lado, foi alterada para admitir a entrada de tabelas de tempo para cada tarefa. Cada tabela representa os tempos de *setup* em relação a cada tarefa tendo precedida-a.

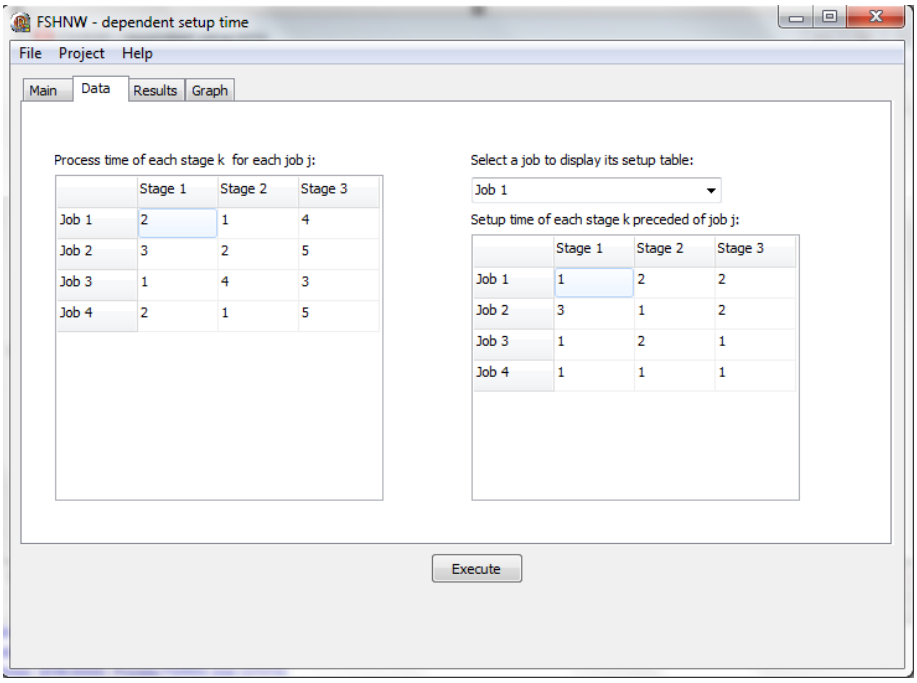

Figura 7 – Tela com os tempos de processamento das tarefas e *setup* das máquinas para o programa de *setup* dependente.

A Figura 8 apresenta a tela de apresentação das ordenações e *makespan's* obtidos para cada respectiva tarefa. No caso de setup dependentes, as tabelas apenas foram alteradas para conter as 11 regras de prioridade.

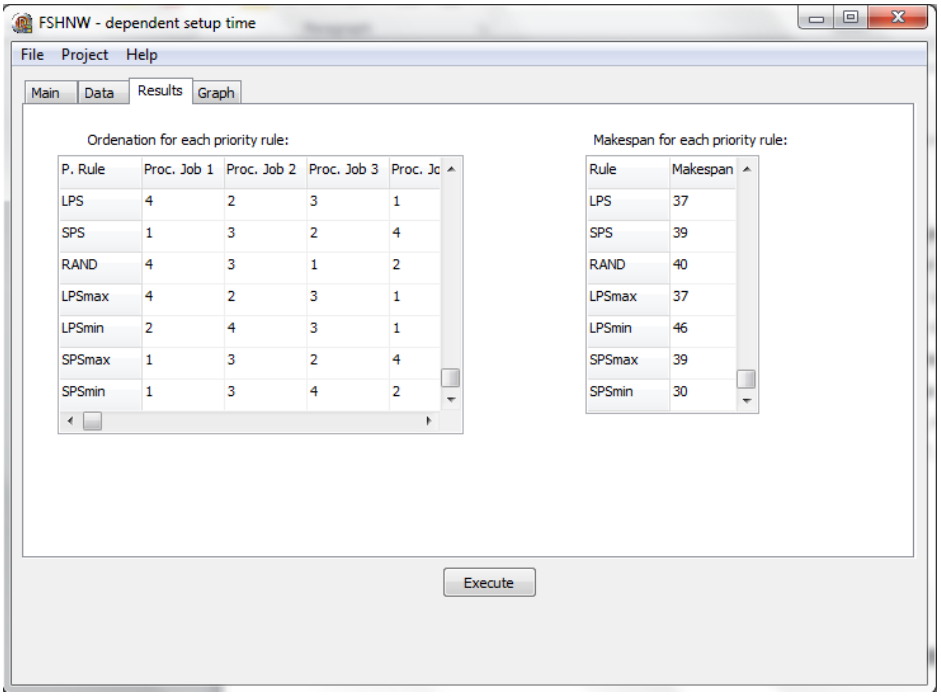

Figura 8 – Tela com os ordenações e *makespans* por regra de prioridade para o programa de *setup* dependente.

5.3 Análise de performance das regras de prioridade para o caso de *setup* independente

A performance das regras de prioridade será avaliada através da porcentagem de sucesso destas para problemas de diversas classes e tamanhos. Para tal análise foram gerados 32.000 casos onde tarefas devem ser programadas. Os casos são gerados aleatoriamente obedecendo alguns critérios.

Os primeiros critérios são quanto aos valores do número de tarefas, estágios e quanto a flexibilidade. A Tabela 3 apresenta os níveis em esses valores podem variar. Tais níveis significam que as variáveis podem assumir apenas os valores do próprio nível. Por exemplo, o número de tarefas poderá ser apenas 10, 20, 30, 40, 50, 60, 70, 80, 90 ou 100. O valor do número de estágios poderá ser apenas 3, 5 ou 7. Finalmente, a flexibilidade poderá ser apenas baixa (1), média (2) ou alta (3).

Tabela 3 – Parâmetros da experimentação computacional.

Parâmetro	**Símbolo**	**Número de níveis**	**Valores**
Número de tarefas	*N*	10	10, 20, 30, ..., 100
Número de estágios	*G*	3	3, 5, 7
Flexibilidade	*F*	3	Baixa, média, alta

Os níveis de flexibilidade definem o número de estágios que irão conter máquinas paralelas. Os três níveis são: baixa, em que 1/3 dos estágios conterão máquinas paralelas; média, com 2/3 dos estágios contendo máquinas paralelas; e alta, com todos os estágios contendo máquinas paralelas.

O número de estágios será o inteiro mais próximo do produto obtido entre os valores definidos e o número de estágios do problema. Para os casos aleatórios, os estágios que conterão máquinas paralelas são escolhidos aleatoreamente. Em tais estágios, o número de máquinas paralelas também irá variar aleatoreamente de 2 a 4.

Para os tempos de processamento e de *setup* são definidos dois grupos de problema. Esses grupos se diferem pelos valores que *p* e *s* podem assumir. O grupo I é apresentada na Tabela 4 e o grupo II na Tabela 5.

Tabela 4 – Variáveis da experimentação computacional – Grupo I.

Parâmetro	**Símbolo**	**Número de níveis**	**Valores**
Tempo de processamento de tarefas	*p*	1	U[01, 50]
Tempo de preparação das máquinas (*setup*)	*S*	2	U[01, 35], U[35, 70]

Tabela 5 – Variáveis da experimentação computacional – Grupo II.

Parâmetro	**Símbolo**	**Número de níveis**	**Valores**
Tempo de processamento de tarefas	*p*	1	U[01, 99]
Tempo de preparação das máquinas (*setup*)	*S*	2	U[01, 60], U[60, 120]

Um código específico foi escrito para gerar todos os casos aleatoriamente, sendo que 16.000 foram gerados de acordo com o grupo I e 16.000 com o grupo II. O resolvedor desenvolvido foi então adaptado para resolver cada grupo de problemas de uma só vez.

Para fazer a análise de performance das regras de prioridade foi feito uso de algumas estatísticas obtidas da experimentação. Os resultados são então analisados através da porcentagem de sucesso, desvio relativo médio e desvio-padrão do desvio relativo para todos os métodos de solução a serem comparados.

A porcentagem de sucesso referente a uma classe do problema é calculada pelo número de vezes que o método forneceu a melhor solução (empatando ou não), dividido pelo número de problemas da classe.

O desvio relativo (DR) é o percentual da 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 *makespan*, ou seja, o algoritmo apresentou a melhor programação. Entretanto, mais de um método pode fornecer a melhor programação.

Desta forma, o melhor algoritmo é aquele que apresenta o menor valor de desvio relativo médio (a média aritmética dos desvios relativos) para uma determinada classe de problemas.

O desvio relativo (DR_h) de um método (regra de prioridade) h para um determinado problema é assim calculado,

$$DR_h = \frac{D_h - D^*}{D^*} \tag{11}$$

onde D_h é o *makespan* fornecido pelo método h e D^* é o melhor makespan fornecido pelos métodos.

O desvio-padrão de uma amostra mede o grau de dispersão dos elementos em torno da média. O desvio-padrão do desvio relativo é o valor da variação dos desvios relativos de uma classe de problemas em torno do desvio relativo médio. Quanto menor for o valor do desvio-padrão, mais estável é o desempenho do método.

O desvio-padrão do desvio relativo (S_h) de um método h é calculado da seguinte forma:

$$S_h = \sqrt{\frac{\sum_{i=1}^{L}\left(DR_{h_i} - DRM\right)}{L-1}} \tag{11}$$

onde L é o número de problemas da classe, DR_{hi} é o desvio relativo da solução do problema i e DRM é o desvio relativo médio da classe de problemas.

A primeira parte desta seção de resultados é dedicada à análise de performance das regras de prioridade para o ambientes *flow shop* híbrido com tempos de *setup* explícitos independentes (FSHSEi).

A Figura 9 apresenta os percentuais de sucesso de cada regra de prioridade em função das dimensões do problema.Como pode ser observado, a regra de prioridade que mostrou melhor performance para os problemas do grupo I foi a SP, seguido das regras LP, SPS e LPS.Também pode ser observado que a regra RAND apresentou um dos piores resultados

independente do tamanho do problema. Além disso, os percentuais de sucesso das regras SP e LP aumentam à medida que as dimensões do problema aumentam.

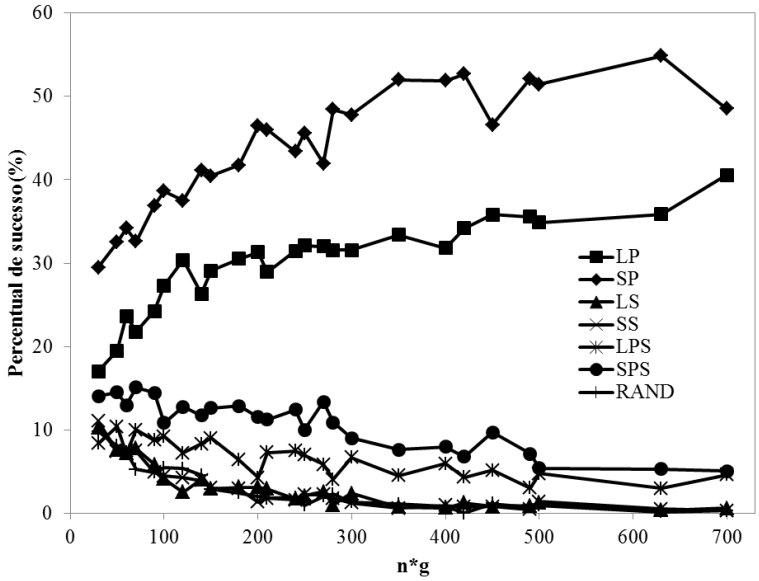

Figura 9 – Porcentagem de sucesso para cada regra de prioridade em função do tamanho do problema ($n_x g$) para o grupo I - FSHSEi.

As quatro melhores regras de prioridade são reagrupadas na Figura 10 junto com os erros relativos, onde pode ser observado que a regra de prioridade SP apresentou os menores desvios relativos além de a melhor porcentagem de sucesso, possuindo desvios relativos irrelevantes para problemas de grandes dimensões.

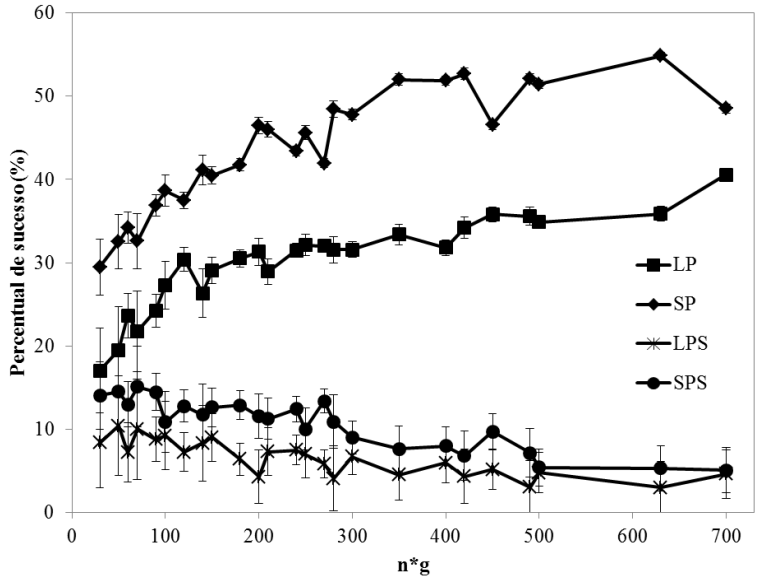

Figura 10 – Porcentagem de sucesso para as melhores regras de prioridade em função do tamanho do problema ($n_x g$) para o grupo I considerando os erros relativos - FSHSEi.

Os dados estatísticos obtidos considerando todos os problemas do grupo I são apresentados na Tabela 6, onde pode ser notado mais uma vez que a regra SP obteve o melhor resultado com o menor desvio relativo e desvio padrão relativo. Destaca-se também os valores obtidos para a regra de prioridade RAND, os quais são em geral inferiores a maioria das regras de prioridade, contendo um dos maiores desvios. Essa informação é importante pois mostra que as regras de prioridade são capazes de obter respostas de maneira estruturada, ou seja, melhores respostas serão obtidas por estas regras de uma forma ordenada e não aleatória.

Tabela 6 - Resultados obtidos para o Grupo I de problemas - FSHSEi.

Regra	Porcentagem de sucesso	Desviorelativo (%)	Desviopadrãorelativo
LP	30.01667	1.775842	0.025379
SP	44.11111	1.134975	0.02024
LS	2.961111	5.505801	0.039725
SS	2.827778	5.538767	0.03981
LPS	6.594444	3.166424	0.027254
SPS	10.51667	2.692012	0.025439
RAND	2.972222	5.530271	0.039775

O mesmo estudo pode ser feito para os problemas do grupo II, onde resultados semelhantes foram obtidos. A partir da Figura 11, assim como obtido foi observado da Figura 9, a regra SP apresentou melhores resultados seguido das regras LP, SPS e LPS. A Figura 12 mostra que a regra SP também apresentou os menores desvios nos resultados.

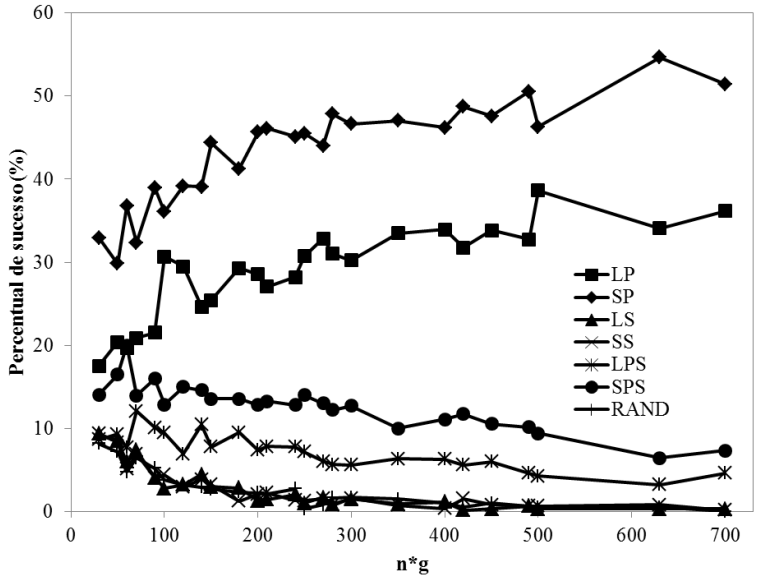

Figura 11 – Porcentagem de sucesso para cada regra de prioridade em função do tamanho do problema ($n_x g$) para o grupo II.

A Figura 12 mostra um estudo semenlhante ao apresentado na Figura 10 para o grupo II. Através dessa figura pode-se concluir que, novamente, a regra SP apresentou menores desvios que se reduzem com o tamanho do problema.

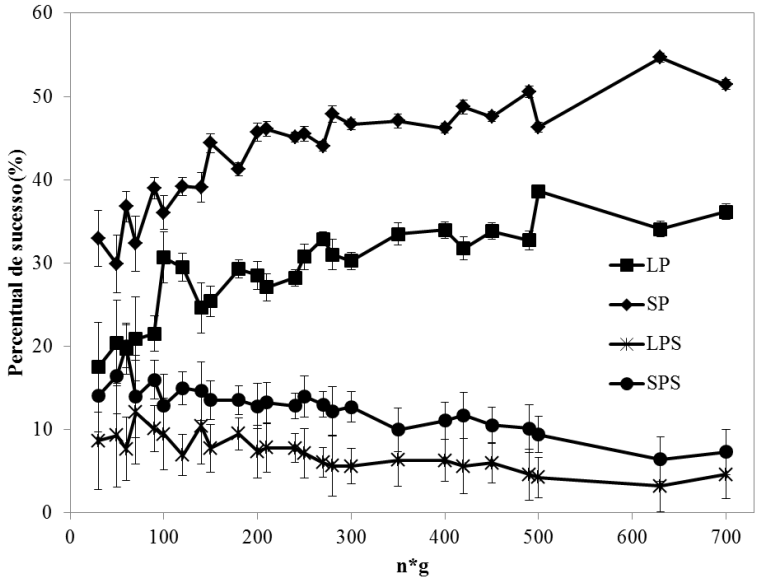

Figura 12 – Porcentagem de sucesso para as melhores regras de prioridade em função do tamanho do problema ($n_x g$) para o grupo II considerando os erros relativos - FSHSEi.

Os resultados considerando todos os problemas do grupo II são apresentados na Tabela 7, onde pode-se observar que resultados semelhantes aqueles obtidos na Tabela 6.

Tabela 7 - Resultados obtidos para o Grupo II de problemas - FSHSEi.

Regra	Porcentagem de sucesso	Desviorelativo (%)	Desviopadrãorelativo
LP	28.82222	1.912784	0.026928
SP	43.65556	1.19536	0.021344
LS	2.427778	6.10968	0.042899
SS	2.5	6.185902	0.043121
LPS	7.311111	3.212788	0.028642
SPS	12.62778	2.653874	0.025903
RAND	2.655556	6.127868	0.043291

O estudo realizado para os dois grupos de problemas mostra que a melhor regra de prioridade foi a SP, tendo pequenos desvios e resultados obtidos de forma estruturada quando considerando tempos de *setup* explicitos e independentes da tarefa precedente.

A próxima seção de resultados contemplará os resultados de validação para regras de prioridade para problemas de *flow shop* híbrido com tempos de *setup* explícitos e dependentes da tarefa anterior em cada estágio de produção (FSHSEd).

5.4 Análise de performance das regras de prioridade para o caso de *setup* dependente

Para a análise de performance do problema de FSHSEd um método semelhante ao aplicado para o FSHSEi. Dois grupos foram gerados (I e II) com 16000 problemas cada, usando as Tabelas 3, 4 e 5. A diferença aqui é que a matriz de tempos de setup é agora uma matriz de blocos, onde cada bloco é tempos de setup é gerado de acordo com a Tabela 4 ou 5.

A Figura 13 apresenta um estudo de performance de cada regra de prioridade o grupo I.É possivel observar pela Figura 13 que várias regras de prioridade apresentaram performance similar. No entanto, aquela que apresentou melhor performance nesse tipo de problema foi a SPS, seguida da SPSmin, SP e SPSmax. Essas quatro regras de prioridade são reagrupadas na Figura 14 onde os erros relativos são também apresentados, onde é possível observar que as regras de prioridade em questão apresentaram desvios relativos semelhantes.

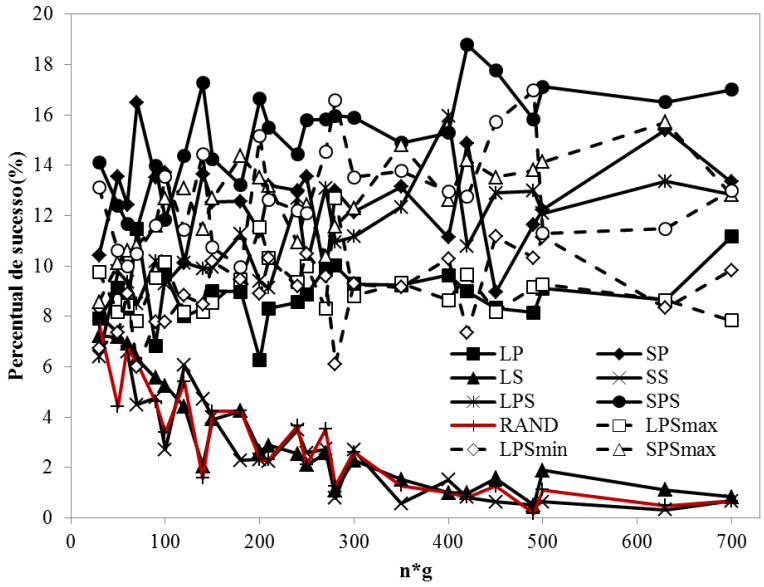

Figura 13 – Porcentagem de sucesso para cada regra de prioridade em função do tamanho do problema ($n_x g$) para o grupo I - FSHSEd.

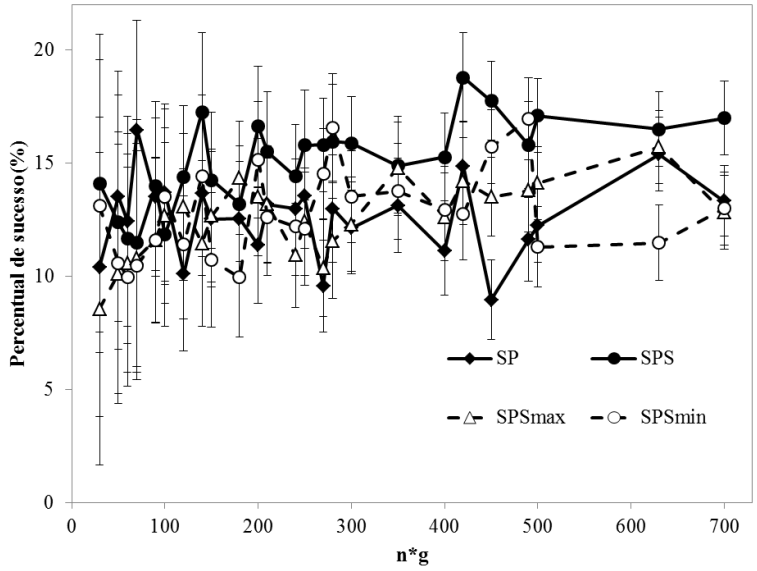

Figura 14 – Porcentagem de sucesso para as melhores regras de prioridade em função do tamanho do problema ($n_x g$) para o grupo I considerando os erros relativos - FSHSEd.

Os dados considerando todos os casos são apresenatados na Tabela 8, onde pode-se observar a superioridade das regras de prioridade apresentadas na Figura 14.

Tabela 8 - Resultados obtidos para o Grupo I de problemas - FSHSEd.

Regra	Porcentagem de sucesso	Desviorelativo (%)	Desviopadrãorelativo
LP	8.89	3.43	0.0347
SP	12.60	2.82	0.0307
LS	3.07	7.22	0.0467
SS	2.86	7.27	0.0470
LPS	10.90	3.46	0.0352
SPS	15.10	2.88	0.0312
RAND	2.87	7.29	0.0470
LPSmax	9.20	3.43	0.0352
LPSmin	9.04	3.44	0.0348
SPSmax	12.60	2.85	0.0307
SPSmin	12.80	2.85	0.0308

Para o grupo II uma análise semelhante é realizada. Primeiro plotamos todos os percentuais de sucesso juntos (Figura 15), e depois plotamos os melhores na Figura 16 com seus desvios relativos. As melhores regras de prioridade foram as mesmas obtidas para o grupo I.

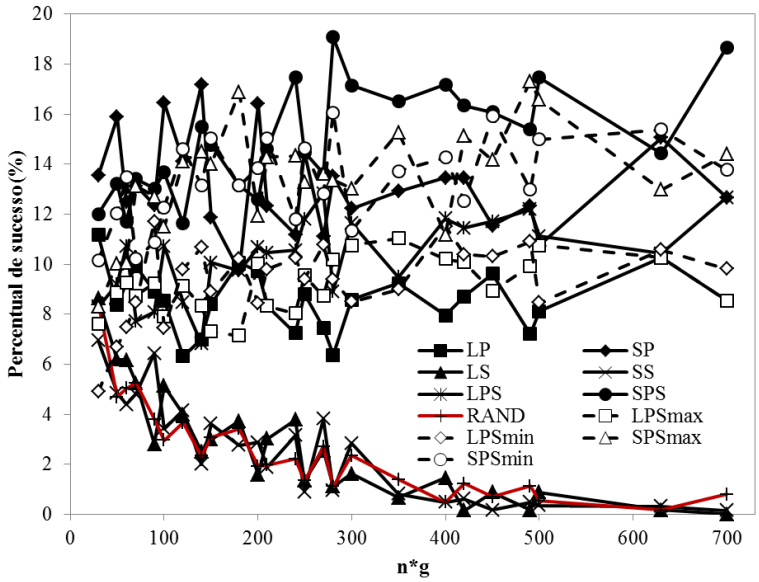

Figura 15 – Porcentagem de sucesso para cada regra de prioridade em função do tamanho do problema ($n_x g$) para o grupo II - FSHSEd.

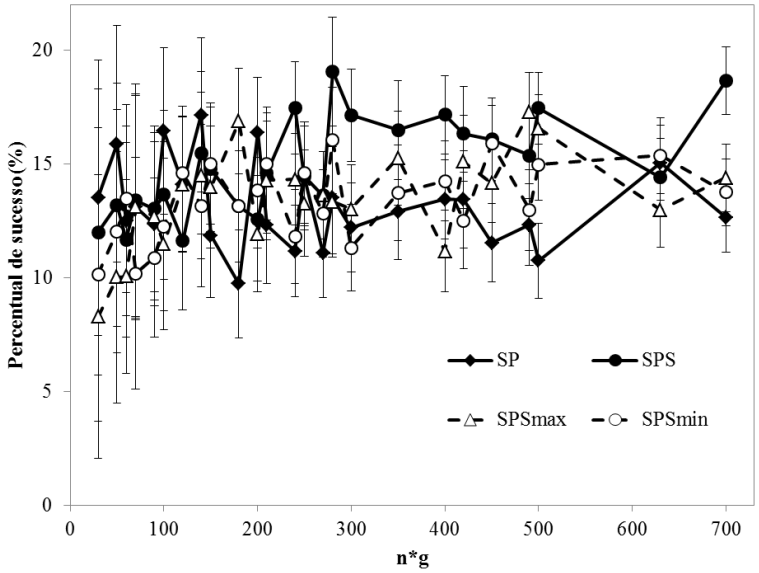

Figura 16 – Porcentagem de sucesso para as melhores regras de prioridade em função do tamanho do problema ($n_x g$) para o grupo II considerando os erros relativos - FSHSEd.

Os dados considerando todos os casos do grupo II são apresentados na Tabela 9. Onde foram obtidos resultados semelhantes aqueles para o grupo I.

Tabela 9 - Resultados obtidos para o Grupo II de problemas - FSHSEd.

Regra	Porcentagem de sucesso	Desviorelativo (%)	Desviopadrãorelativo
LP	8.57	3.37	0.0331
SP	13.10	2.71	0.0293
LS	2.58	7.65	0.0488
SS	2.49	7.64	0.0482
LPS	10.30	3.43	0.0338
SPS	15.00	2.77	0.0302
RAND	2.46	7.67	0.0485
LPSmax	9.18	3.38	0.0336
LPSmin	9.31	3.38	0.0334
SPSmax	13.50	2.73	0.0297
SPSmin	13.40	2.71	0.0298

O estudo de validação apresentado para as regras de prioridade consideradas mostra que a melhor regra de prioridade para o caso de *flow shop* híbrido com tempos de *setup* explícitos dependentes foi a regra SPS. As mesmas respostas foram obtidos para ambos os grupos de problemas.

Para melhor ilustrar os resultados que podem ser obtidos pelo algoritmo implementado neste trabalho, as duas próximas seções tratarão de dois casos teste, um para FSHSEi e outro para FSHSEd, isolados, onde serão apresentados os valores de *makespan* e seus diagramas de Gantt.

5.5 Caso 1: FSHSEi

Para o Caso 1 foi desenvolvido um problema de pequeno porte com o objetivo de proporcionar uma melhor visualização da ferramenta computacional desenvolvida. Os dados referentes as dimensões do problema são apresentados na Tabela 10.

Tabela 10 – Dimensões do Caso 1.

Parâmetro	**Valor**
Número de tarefas	5
Número de estágios	3
Quantidade de máquinas no estágio 1	1
Quantidade de máquinas no estágio 2	2
Quantidade de máquinas no estágio 3	2

Os tempos de processamento para o Caso 1 são apresentados na Tabela 11 e os tempos de setup são apresentados na Tabela 12.

Tabela 11 – Tempos de processamento do Caso 1.

	Estágio 1	**Estágio 2**	**Estágio 3**
Tarefa 1	2	16	4
Tarefa 2	6	4	5
Tarefa 3	8	4	6
Tarefa 4	2	10	5
Tafera 5	6	4	5

Tabela 12 – Tempos de processamento do Caso 1.

	Estágio 1	**Estágio 2**	**Estágio 3**
Tarefa 1	1	4	2
Tarefa 2	3	1	2
Tarefa 3	1	2	4
Tarefa 4	2	1	3
Tarefa 5	5	2	3

As ordenações obtidas pelo código para cada regra de prioridade são escritas na Tabela 13.

Tabela 13 – Ordenações do Caso 1.

RP	J_1	J_2	J_3	J_4	J_5
LP	1	3	4	2	5
SP	2	5	4	3	1
LS	5	3	1	4	2
SS	2	4	3	1	5
LPS	1	3	5	4	2
SPS	2	4	3	5	1
RAND	5	1	3	2	4

Os *makespans* obtidos são apresentaos na Tabela 14, onde pode-se observar que ao contrário do esperado a melhor regra de prioridade foi a LPS, enquando que a regra SP, obtida como melhor na análise de performance, foi a pior regra. Uma forma de analisar este resultado é observar que as dimensões desse problema são de (*nxg)*= 15. Esse resultado sugere que esse problema está entre os problemas com maior desvio relativo, e portanto, prêve qual regra de prioridade será mais bem sucedida não é tão simples.

Tabela 14 – Makespans do Caso 1.

RP	**Makespan**
LP	47
SP	58
LS	49
SS	47
LPS	46
SPS	56
RAND	51

Por fim, os diagramas de Gantt para cada regra de prioridade são apresentados a seguir:

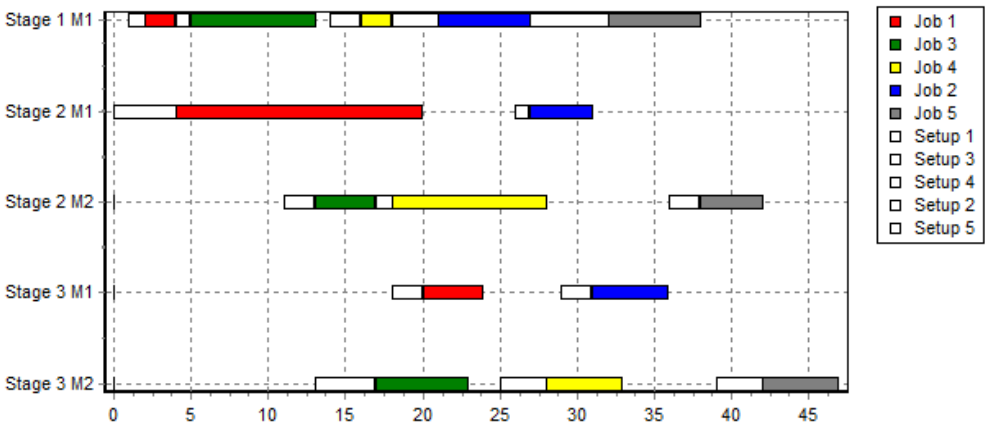

Figura 17 – Diagrama para regra LP – Caso 1.

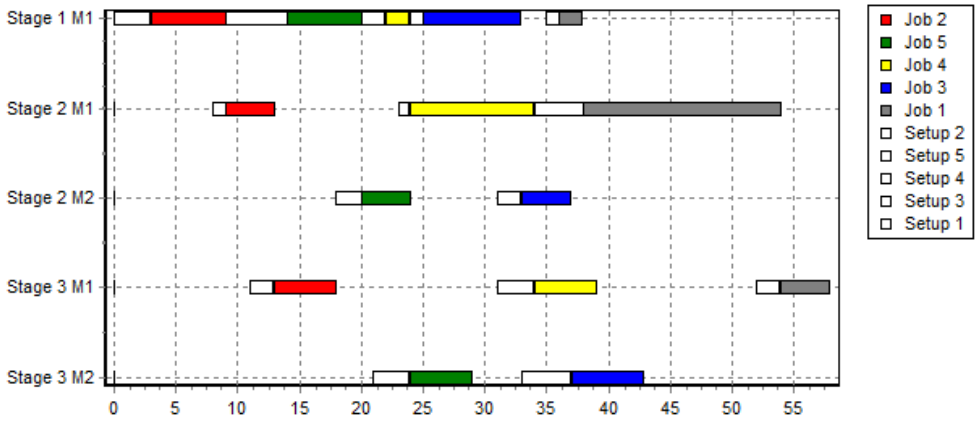

Figura 18 – Diagrama para regra SP – Caso 1.

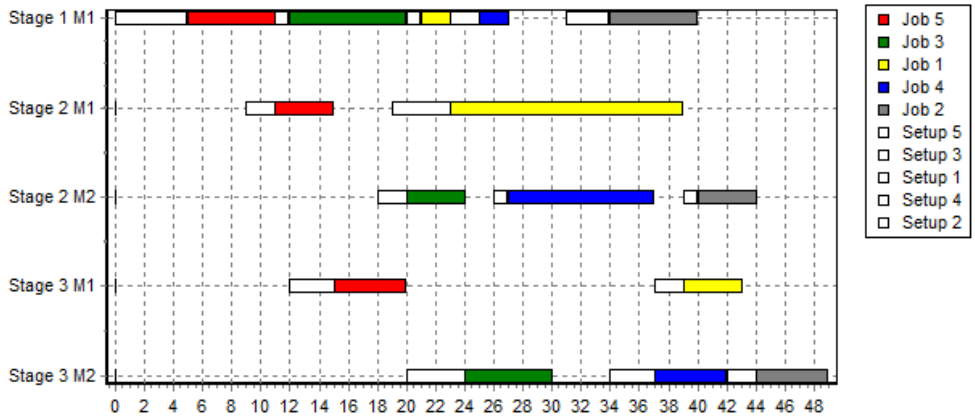

Figura 19 – Diagrama para regra LS – Caso 1.

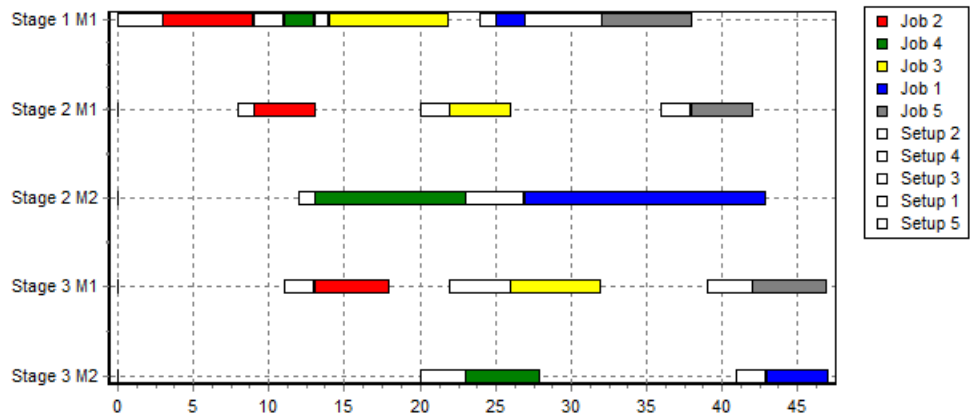

Figura 20 – Diagrama para regra SS – Caso 1.

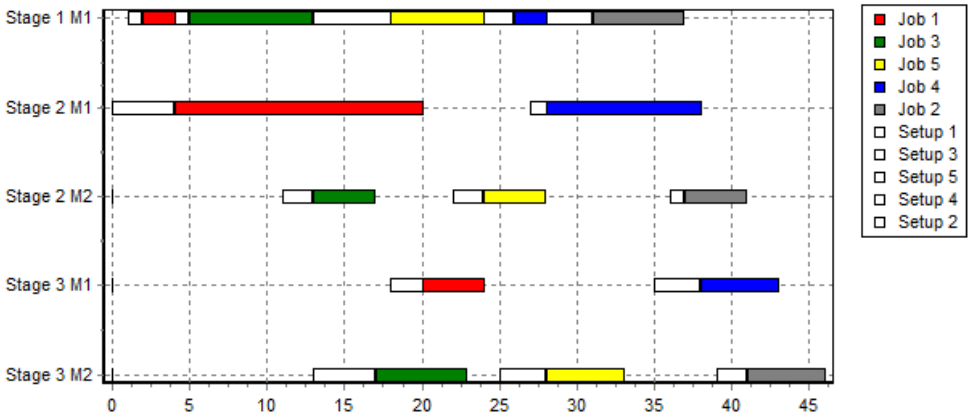

Figura 21 – Diagrama para regra LPS – Caso 1.

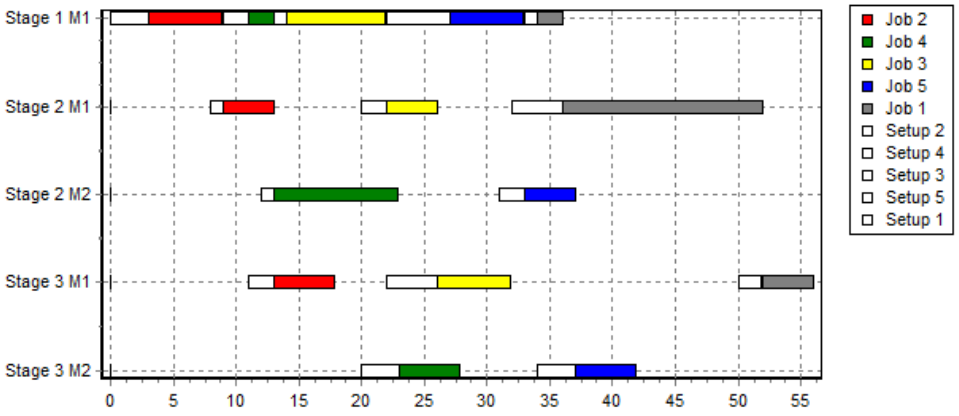

Figura 22 – Diagrama para regra SPS – Caso 1.

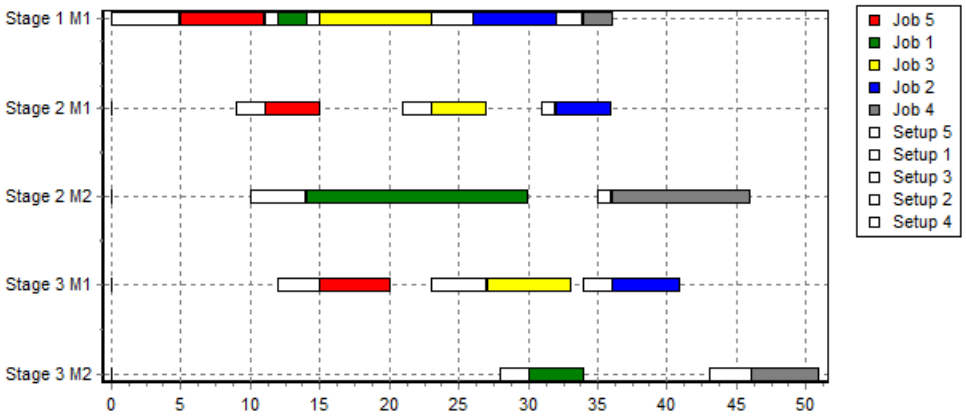

Figura 23 – Diagrama para regra RAND – Caso 1.

5.6 Caso 2: FSHSEd

O último estudo conduzido neste trabalho consiste de um caso teste para o problema FSHSEd. Os parâmetros relacionados as dimensões do problema são apresentados na Tabela 15.

Tabela 15 – Dimensões do Caso 2.

Parâmetro	**Valor**
Número de tarefas	5
Número de estágios	3
Quantidade de máquinas no estágio 1	1
Quantidade de máquinas no estágio 2	2
Quantidade de máquinas no estágio 3	2

Os tempos de processamento para cada tarefa em cada estágio são apresentados na Tabela 16.

Tabela 16 – Tempos de processamento do Caso 2.

	Estágio 1	**Estágio 2**	**Estágio 3**
Tarefa 1	14	2	11
Tarefa 2	6	7	18
Tarefa 3	15	18	6
Tarefa 4	8	16	18
Tarefa 5	13	3	8

Os tempos de setup para a tarefa 1, considerando todas as possibilidades de tarefa prescedente em cada estágio são apresentados na Tabela 17.

Tabela 17 – Tempos de setup para a tarefa 1 do Caso 2.

	Estágio 1	**Estágio 2**	**Estágio 3**
Primeira tarefa	6	6	1
Após tarefa 2	2	4	3
Após tarefa 3	2	3	9
Após tarefa 4	3	4	8
Após tarefa 5	4	3	2

Os tempos de setup para as demais tarefas são apresentadas nas Tabelas 18, 19, 20 e 21 para as tarefas 2, 3, 4 e 5, respectivamente.

Tabela 18 – Tempos de setup para a tarefa 2 do Caso 2.

	Estágio 1	**Estágio 2**	**Estágio 3**
Após tarefa 1	4	9	3
Primeira tarefa	1	3	2
Após tarefa 3	6	9	1
Após tarefa 4	4	7	2
Após tarefa 5	1	8	8

Tabela 19 – Tempos de setup para a tarefa 3 do Caso 2.

	Estágio 1	**Estágio 2**	**Estágio 3**
Após tarefa 1	6	2	5
Após tarefa 2	5	4	1
Primeira tarefa	6	7	9
Após tarefa 4	8	7	9
Após tarefa 5	9	5	3

Tabela 20 – Tempos de setup para a tarefa 4 do Caso 2.

	Estágio 1	**Estágio 2**	**Estágio 3**
Após tarefa 1	2	6	5
Após tarefa 2	2	6	3
Após tarefa 3	1	7	3
Primeira tarefa	4	2	9
Após tarefa 5	3	2	1

Tabela 21 – Tempos de setup para a tarefa 5 do Caso 2.

	Estágio 1	**Estágio 2**	**Estágio 3**
Após tarefa 1	5	1	7
Após tarefa 2	6	3	1
Após tarefa 3	4	1	4
Após tarefa 4	9	4	7
Primeira tarefa	1	3	7

Os resultados obtidos através do código são apresentados na Tabela 22.

Tabela 22 – Ordenações do Caso 2.

RP	**J_1**	**J_2**	**J_3**	**J_4**	**J_5**
LP	4	3	2	1	5
SP	5	1	2	3	4
LS	3	2	5	1	4
SS	4	1	5	2	3
LPS	3	4	2	1	5
SPS	5	1	2	4	3
RAND	5	1	2	3	4
LPS	3	4	2	1	5
LPs	3	4	2	1	5
SPS	5	1	2	4	3
SPs	5	1	2	4	3

Os valores de makespan obtidos são apresentados na Tabela 23, onde pode ser observado que as regras LPS, LPS e LPs apresentaram os melhores resultados.

Tabela 23 – Ordenações do Caso 2.

RP	**Makespan**
LP	91
SP	102
LS	116
SS	97
LPS	95
SPS	96
RAND	102
LPS	95
LPs	95
SPS	96
SPs	96

Os diagramas de Gantt para cada regra de prioridade são apresentados a seguir.

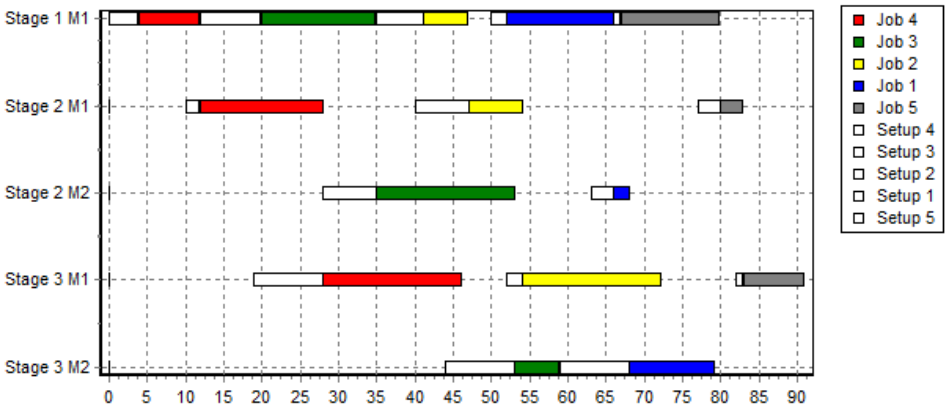

Figura 24 – Diagrama para regra LP – Caso 2.

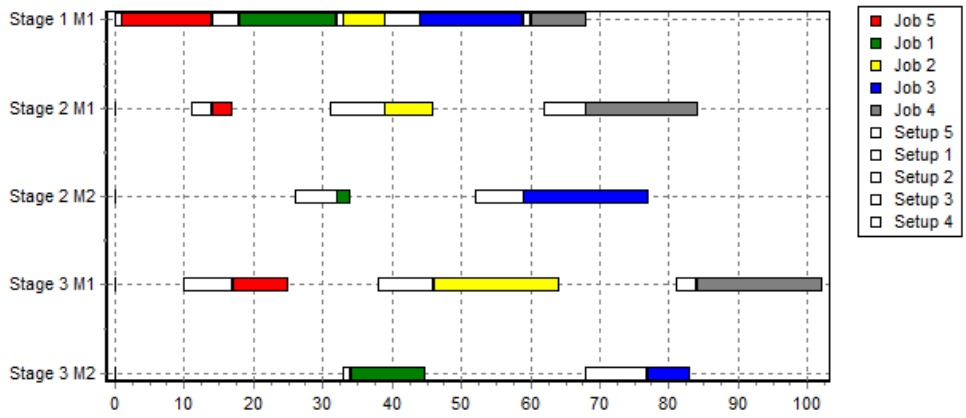

Figura 25 – Diagrama para regra SP – Caso 2.

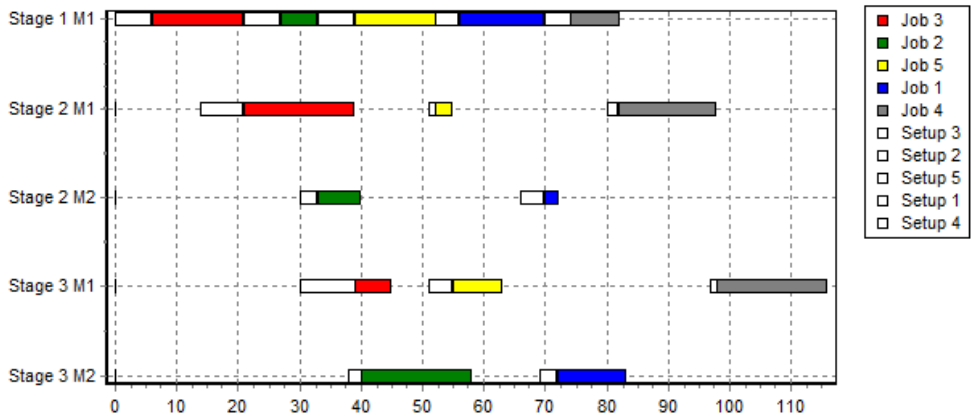

Figura 26 – Diagrama para regra LS – Caso 2.

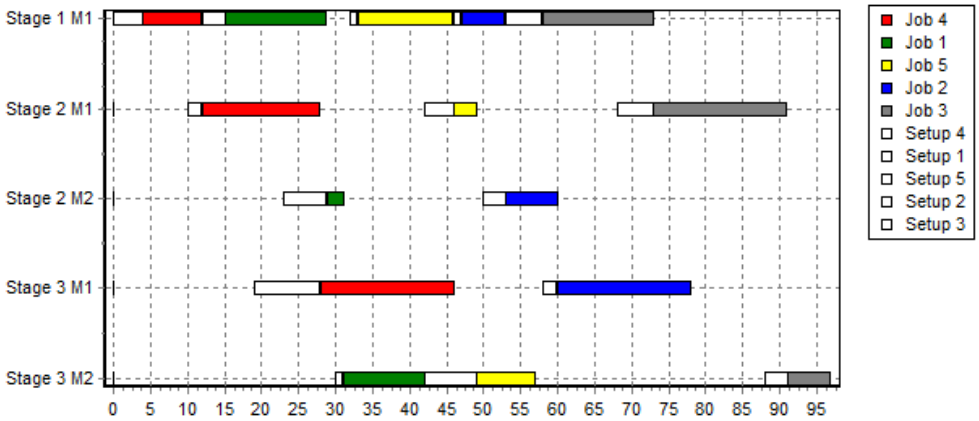

Figura 27 – Diagrama para regra SS – Caso 2.

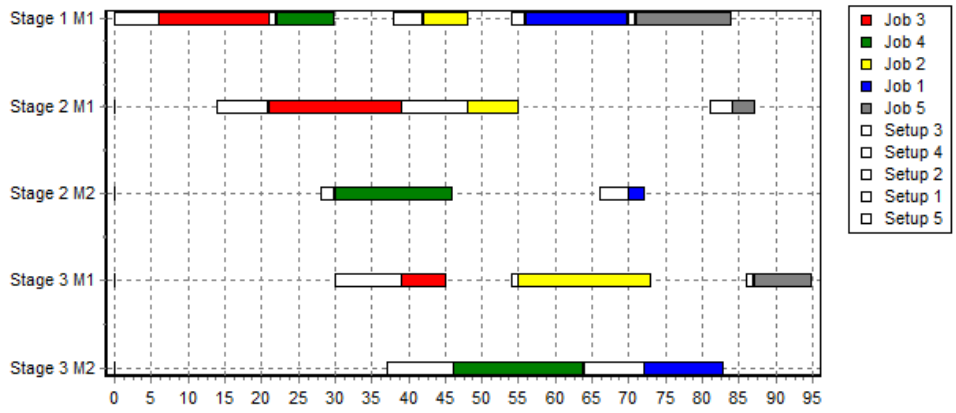

Figura 28 – Diagrama para regra LPS – Caso 2.

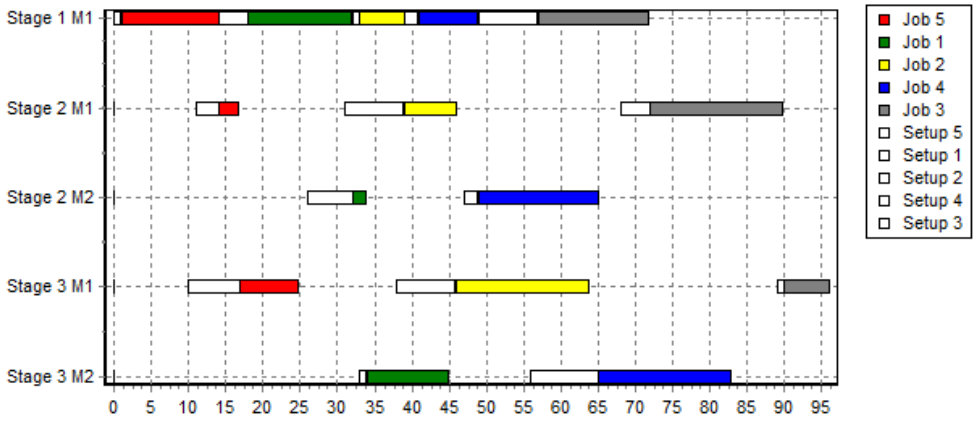

Figura 29 – Diagrama para regra SPS – Caso 2.

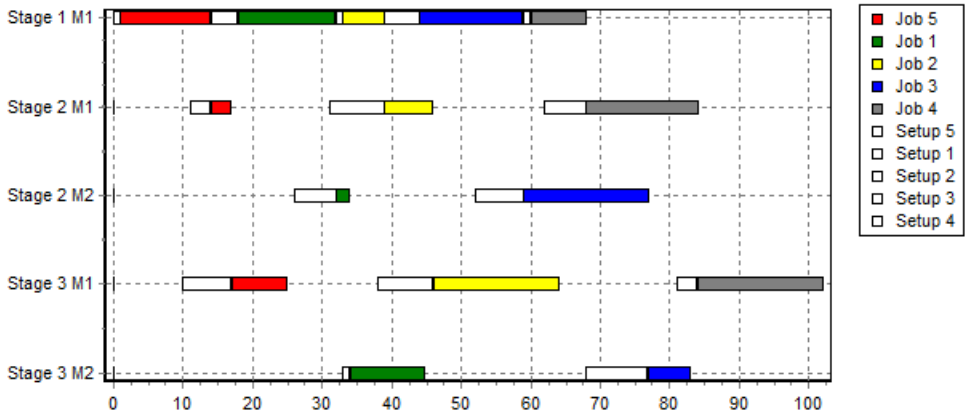

Figura 30 – Diagrama para regra RAND – Caso 2.

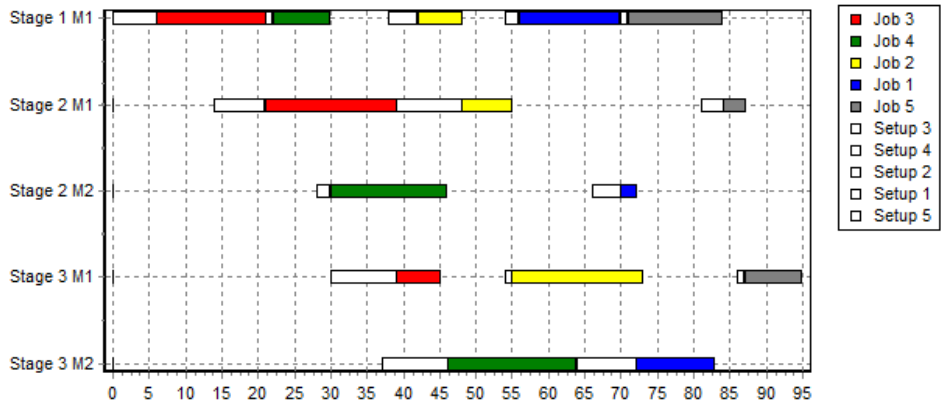

Figura 31 – Diagrama para regra LPS – Caso 2.

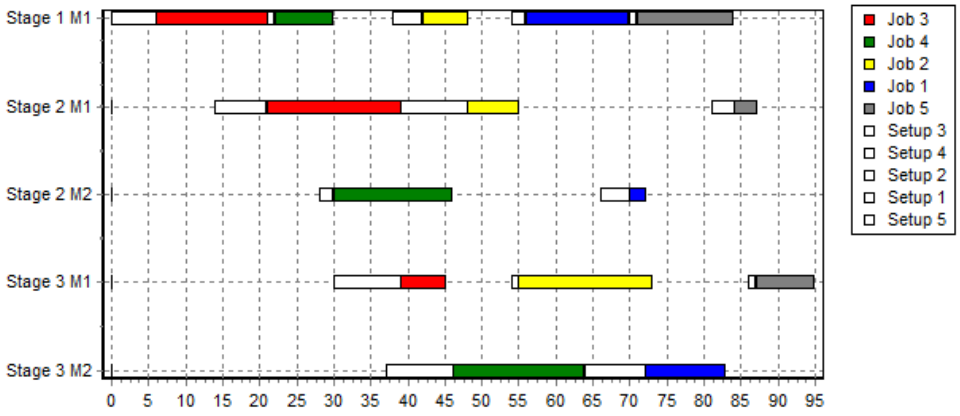

Figura 32 – Diagrama para regra LPs – Caso 2.

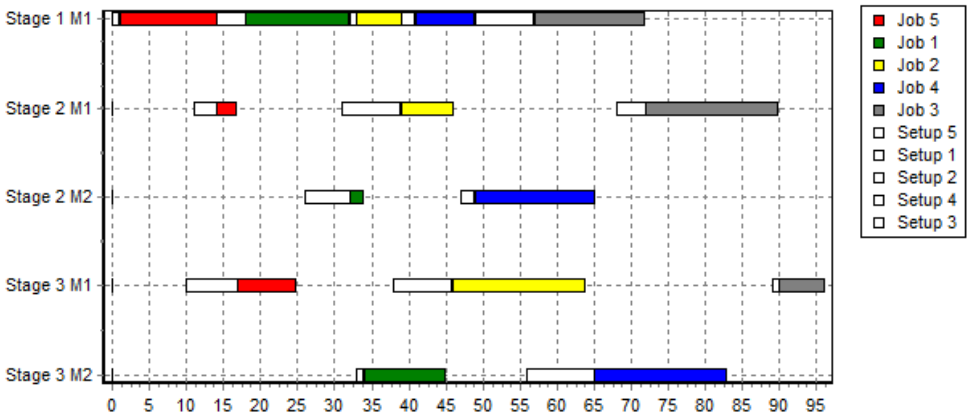

Figura 33 – Diagrama para regra SPS – Caso 2.

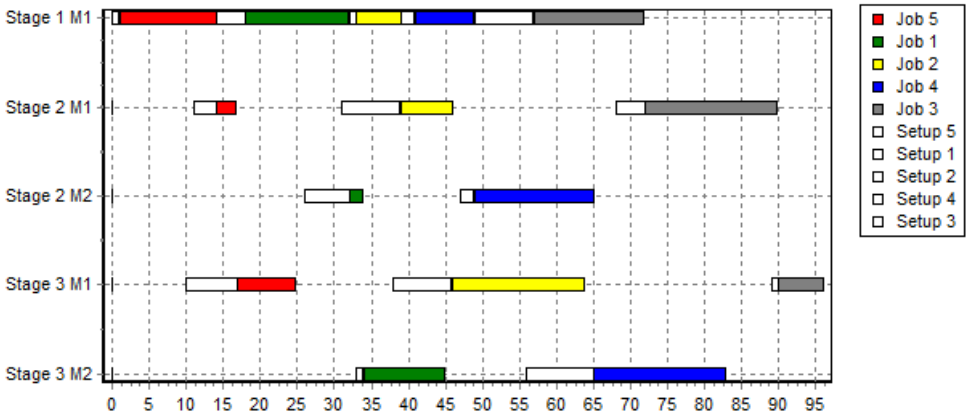

Figura 34 – Diagrama para regra SPs – Caso 2.

Como pode ser observado das figuras as regras LPS e LPs apresentaram a mesma ordenação, bem como as regras SPS e SPs.

CAPÍTULO 06 – CONSIDERAÇÕES FINAIS E SUGESTÕES PARA TRABALHOS FUTUROS

No presente trabalho foram implementadas regras de prioridade para os problemas de flow-shop híbrido com tempos de setup explicitos independentes e dependentes bem como os algorítmos para solução de ambos os problemas. Uma interface gráfica para facilitar o uso, extração de dados e salvamento de projetos também foi desenvolvida.

As regras de prioridade obtidas forneceram, em geral, resultados estruturados confiáveis sendo bem mais úteis do que testar ordenações aleatoreamente. Algumas regras de prioridade se mostraram mais eficiente do que outras, como foi o caso da regra SP para o caso de setup independente e das regras SP, SPS, SPS e SPs para o caso de setup dependente.

Dois casos extras foram testados para mostrar os resultados em termos de diagramas para cada regra de prioridade.

Os tempos de processamento e setup nesse trabalho foram considerados únicos e iguais para todas as máquinas pertencentes a um mesmo estágio. Essa ainda é uma situação ideal. Um bom tema para trabalhos futuros poderia ser considerar máquinas de diferentes capacidades em um mesmo estágio. Tal consideração aumentaria em muito a complexidade do problema e abriria espaço para a aplicação de novas regras de prioridade.

Um outro tópico importante a ser considerado como tema de trabalhos futuros é a comparação das regras de prioridade com heurísticas de outra natureza. As regras de prioridade apresentaram bons resultados e um baixissimo custo operacional, visto que é necessário calcular um número de amostras igual ao número de regras para um determinado problema. Heurísticas como algoritmo genético precisariam calcular um número elevado de amostras para obter algum resultado satisfatório. A questão é se essas heurísticas apresentariam ou não resultados superiores às regras de prioridade estudas, e se sim, em que magnitude.

REFERÊNCIAS BIBLIOGRÁFICAS

ALDOWAISAN, T.; ALLAHVERDI, A. (2004).New heuristics for m-machine no-wait flowshop to minimize total completion time.OMEGA, v.32, p.345-352.

ALLAHVERDI, A.; ALDOWAISAN, T. (2000).No-wait and separate setup three-machine flowshop with total completion time criterion. International Transactions in Operational Research, v.7, p.245– 264.

ALLAHVERDI, A.; ALDOWAISAN, T. (2001).Minimizing total completion time in a no-wait flowshop with sequence-dependent additive changeover times.Journal of the Operational Research Society, v.52, p.449–462.

BERTOLISSI, E. (2000). Heuristic algorithm for scheduling in the no-wait flow-shop.Journal of Materials Processing Technology, v.107, p.459-465.

BROWN, S.I.; McGARVEY, R.G.; VENTURA, J.A. (2004). Total flowtime and makespan for a no-wait *m*-machine flowshop with set-up times separated. Journal of the Operational Research Society, v.55, p.614-621.

CHEN, C.L.; NEPPALLI, R.V.; ALJABER, N. (1996). Genetic algorithms applied to the continuous flow shop problem. Computers & Industrial Engineering, v.30, p.919–29.

CHOONG, F.; PHON-AMNUAISUK, S.; ALIAS, M.Y. (2011). Metaheuristic methods in hybrid flow shop scheduling problem. Expert Systems with Applications, v.38, p.10787–10793.

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 seqüência. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.

FUCHIGAMI, H.Y.; MOCCELLIN, J.V. (2006).Estudo da influência da programação do primeiro estágio em sistemas *flow shop* híbridos com tempos de *setup* independentes da seqüência de processamento das tarefas. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 38., 2006, Goiânia. Anais... Rio de Janeiro: Sociedade Brasileira de Pesquisa Operacional. CD-ROM.

FUCHIGAMI, H.Y.; MOCCELLIN, J.V. (2007).Análise de desempenho de regras de prioridade para programação em *flow shop* com múltiplas máquinas e tempos de *setup* independentes da seqüência. In: ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, 27., 2007, Foz do Iguaçu. Anais... Rio de Janeiro: Associação Brasileira de Engenharia de Produção. CD-ROM.

FUCHIGAMI, H.Y.; MOCCELLIN, J.V.; RUIZ, R. (2007).Análise comparativa do desempenho de regras de prioridade em sistemas *flexible flow line* com múltiplas máquinas e tempos de *setup* independentes da seqüência. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 39., 2007, Fortaleza. Anais... Rio de Janeiro: Sociedade Brasileira de Pesquisa Operacional. CD-ROM.

GUINET, A.; ECHALIER, F.; DUSSAUCHOY, A. (1992).Scheduling jobs on parallel machines: a survey.EURO 12, TIMS XXXI, Helsinki, Finland.

HUANG, W.; LI, S. (1998). A two-stage hybrid flowshop with uniform machines and setup times.Mathematical and Computer Modeling, v.27, p.27-45.

HUNSUCKER, J.L.; BRAH, S.A.; SANTOS, D.L. (1989).Simulation study of a flow shop with multiple processors scheduling.TIMS/ORSA joint National Meeting, New York, p.16-18.

HUNSUCKER, J.L.; SHAH, J.R. (1994). Comparative performance analysis of priority rules in a constrained flow shop with multiple processor environment. European Journal of Operational Research, v.72, p.102-104.

JUNGWATTANAKIT, J.; REODECHA, M.; CHAOVALITWONGSE, P.; WERNER, F. (2009).A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria.Computers & Operations Research, v.36, p.358–378.

KENNEDY, J.; EBERHART, R. (1995).Particle swarm optimization. In Proceedings of IEEE international conference on neural network (p.1942–1948)

KUMAR, A.; PRAKASH, A.; SHANKAR, R.; TIWARI, M.K. (2006). Psycho-Clonal algorithm based approach to solve continuous flow shop scheduling problem. Expert Systems with Applications, v.31, p.504-514.

LOGENDRAN, R.; CARSON, S.; HANSON, E. (2005). Group scheduling in flexible flow shops. International Journal of Production Economics, v.96, p.143-155.

NAGANO, M.S., MOCCELIN, J.V. e LORENA, L.A.N. intitulado como: Programação da Produção Flow Shop Permutacional com minimização do tempo médio de fluxo. Disponível em: < http://www.lac.inpe.br/~lorena/nagano/sbpo-nagano-mocellin-lorena.pdf>. Acesso em: 20 de Junho de 2013.

OĞUZ, C.; ERCAN, M.F.; CHENG, T.C.E.; FUNG, Y.F. (2003). Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow-shop. European Journal of Operational Research, v.149, p.390-403.

OĞUZ, C.; ZINDER, Y.; DO, V.H.; JANIAK, A. LICHTENSTEIN, M. (2004). Hybrid flow-shop scheduling problems with multiprocessor task systems. European Journal of Operational Research, v.152, p.115-131.

PAN, Q-K.; TASGETIREN, M.F.; LIANG Y-C.(2007). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem.Computers & Operations Research.

POCHET, Y.; MOURSLI, O. (2000).A branch-and-bound algorithm for the hybrid flowshop.International Journal of Production Economics, v.64, p.113-125.

PORTMANN, M-C; VIGNIER, A.; DARDILHAC, D.; DEZALAY, D. (1996).Some hybrid flowshop scheduling by crossing branch and bound and genetic algorithms.5th International Workshop on Project Management and Scheduling (PMS'96), Poznan (Poland), p.186-189.

QIAN, B.; WANG, L.; HU, R.; HUANG, D.X.; WANG, X. (2009).A DE-based approach to no-wait flow-shop scheduling.Computers & Industrial Engineering, v.57, p.787–805.

RAJENDRAN, C.; CHAUDHURI, D. (1990).Heuristic algorithms for continuous flow-shop problem. Naval Research Logistics, v.37, p.695–705.

RUIZ, R.; ALLAHVERDI, A. (2007). Some effective heuristics for no-wait flowshops with setup times to minimize total completion time. Annals of Operations Research, v.156, p.143-171.

RUIZ, R.; MAROTO, C. A. (2006).A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. European Journal of Operational Research, v.169, p.781-800.

RUIZ, R.; ŞERIFOĞLU, F.S.;URLINGS, T. (2008).Modeling realistic hybrid flexible flowshop scheduling problems.Computers & Operations Research, v.35, p.1151-1175.

SAHNI, S.; GONZALES, T. (1976).P-complete approximation problems.Journal of the ACM, v.23, p.555-565.

SHEN, V.Y.; CHEN, Y.E. (1972).A scheduling strategy for the flow-shop problem in a system with two classes of processors.Proceedings of the Conference on Information and System Science, p.645-649.

SHYU, S.J.; LIN, B.M.T.; YIN, P.Y. (2004).Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time.Computers & Industrial Engineering, v.47, p.181-193.

SIMONS JR., J.V. (1992). Heuristics in flow shop scheduling with sequence dependent setup times. OMEGA, v.20, p.215-225.

SRISKANDARAJAH, C. (1993). Performance of scheduling algorithms for no-wait flowshops with parallel machines.European Journal of Operational research, v.70.p.365-378.

TANG, L.X.; ZHANG, Y.Y. (2005). Heuristic combined artificial neural networks to schedule hybrid flow shop with sequence dependent setup times. Lecture Notes in Computer Science, v.3496, p.788-793.

TSENG, L-Y.; LIN, Y-T.(2010). A hybrid genetic algorithm for no-wait flowshop scheduling problem. International Journal of Production Economics, v.128, p.144-152.

VAN DEMAN, J.M.; BAKER, K.R. (1974). Minimizing flowtime in the flow shop with no intermediate queues. AIIE Transactions, v.6, p.28-34.

VIGNIER, A.; BILLAUT, J-C.; PROUST, C. (1995). Les problèmes de type flow shop hybride: état de l'art. Journée d'études: Affectation et Ordonnancement, CNRS / GdR Automatique/ Pôle SED /GT3, Tours, France, p.7-47.

WENG, M.X.; LU, J.; REN, H. (2001). Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. International Journal of Production Economics, v.70, p.215-226.

XIE, J.; XING, W.; LIU, Z.; DONG, J. (2004).Minimum deviation algorithm for two-stage no-wait flowshops with parallel machines.Computers and Mathematics with Applications, v.47, p.1857-1863.

YING, K-c. (2009). An iterated greedy heuristic for multistage hybrid flowshop scheduling problems with multiprocessor tasks. The Journal of the Operational Research Society, v.60, p.810-817.