Universidade de São Paulo

Escola de Engenharia de São Carlos

Departamento de Engenharia de Produção

Felipe Marcus Carneiro

Avaliação de Métodos Heurísticos para a Solução do Problema de Programação

***Flowshop* com Tempos de *Setup* Assimétricos e Dependentes da Sequência**

Orientador: Prof. Dr. Marcelo Seido Nagano

São Carlos

2010

Felipe Marcus Carneiro

**Avaliação de Métodos Heurísticos para a Solução do Problema de Programação *Flowshop*
com Tempos de *Setup* Assimétricos e Dependentes da Sequência**

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo para a obtenção do título de Mestre em Engenharia de Produção.

Área de Concentração: Pesquisa Operacional

Orientador: Prof. Dr. Marcelo Seido Nagano

São Carlos

2010

FOLHA DE JULGAMENTO

Candidato(a): Engenheiro Mecânico FELIPE MARCUS CARNEIRO.

Dissertação defendida e julgada em 23.02.2011 perante a Comissão Julgadora:

Prof. Dr. **MARCELO SEIDO NAGANO** - (Orientador)
(Escola de Engenharia de São Carlos/USP)

APROVADO

Prof. Titular **JOÃO VITOR MOCCELLIN**
(Escola de Engenharia de São Carlos/USP)

APROVADO

Prof. Associado **MARCIO MATTOS BORGES DE OLIVEIRA**
(Faculdade de Economia, Administração e Contabilidade de Ribeirão Preto/FEARP)

APROVADO

Prof. Associado **AQUILES ELIE GUIMARÃES KALATZIS**
Coordenador do Programa de Pós-Graduação em
Engenharia de Produção

Prof. Associado **PAULO CÉSAR LIMA SEGANTINE**
Presidente da Comissão de Pós-Graduação da EESC

“Pedras no caminho? Guardo todas, um dia vou construir um castelo...”

Fernando Pessoa

À minha esposa Rosiane.

AGRADECIMENTOS

Em primeiro lugar, a Deus.

Agradeço também a todos aqueles que me apoiaram nestes quase três anos de estudo: a meus pais Gabriel e Rosalba, que me suportaram no início, quando só podia contar com eles.

A meus companheiros de trabalho por seu desdobramento durante minhas ausências.

Ao professor Nagano por sua dedicação e, principalmente, por ter me ajudado a enxergar a “luz”; ao professor Moccasin por sua paciência e colaboração.

Por fim, e talvez mais devotadamente agradeço à minha esposa Rosiane, pois sem seu apoio e compreensão, acredito, este trabalho não seria possível.

RESUMO

CARNEIRO, F. M. **Avaliação de Métodos Heurísticos para a Solução do Problema de Programação *Flowshop* com Tempos de *Setup* Assimétricos e Dependentes da Sequência**. 2010. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

Este trabalho é dedicado ao problema de programação em *Flowshop* Permutacional com tempos de preparação (*setup*) assimétricos e separados dos tempos de processamento e dependentes da sequência de execução das tarefas e tem o objetivo de minimização da duração total da programação (*Makespan*). Através da investigação das propriedades estruturais do problema, são desenvolvidos os parâmetros *XR* e *QR* de uma programação, que indicam ociosidade das máquinas (para valores positivos) e bloqueio das tarefas (para valores negativos). Os novos parâmetros são utilizados para propor uma melhoria no cálculo eficiente de *Makespan* proposto por Taillard (1990). Esta melhoria é então utilizada no desenvolvimento de uma nova heurística construtiva baseada no método NEHT-RB de Ríos-Mercado e Bard (1998b) denominada CNIT, que é comparada durante a experimentação computacional com os métodos SETUP e TOTAL, de Simons (1992) com pequenas melhorias; com a proposta da utilização da propriedade UBX de Moccellini e Nagano (2007); e com o método NEHT-RB. Os métodos são então submetidos a uma busca local descendente como proposta em Ruiz e Stützle (2008) e seus desempenhos como soluções iniciais para este procedimento de busca local são avaliados. Em seguida, um método melhorativo derivado do novo método construtivo e baseado na meta-heurística IG de Ruiz e Stützle (2008) é proposto e denominado CNIT-IG. O método é comparado com a heurística IG original submetida às diferentes soluções iniciais estudadas durante a avaliação da nova heurística construtiva. As comparações são realizadas utilizando-se o banco de dados de Taillard (1990) para o *flowshop* permutacional adaptado para o problema de *flowshop* com tempos de *setup* assimétricos e dependentes da sequência. Os resultados da experimentação computacional são analisados em termos da porcentagem média de sucesso, do desvio relativo médio e em relação ao tempo médio computacional e mostram a superioridade dos resultados da nova heurística construtiva CNIT e seu alto custo computacional, de complexidade mn^3 . Os resultados mostram ainda a superioridade da meta-heurística CNIT-IG sobre o método IG.

Palavras chave: Programação da produção, *Flowshop* Permutacional, *Setup* Dependente, Métodos Heurísticos, Métodos Meta-Heurísticos.

ABSTRACT

CARNEIRO, F. M. *Heuristic Methods Evaluation for Solution of Flowshop Scheduling Problems with Asymmetric Sequence Dependent Setup Times*. 2010. Dissertation (Mastership) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

This work addresses the Permutation Flowshop scheduling problem with separated sequence-dependent setup times with the objective of minimizing Makespan. Through the investigation of the problem structural properties, two scheduling parameters XR e QR are developed, they indicate the machine idleness (for positive values) and task blocking (for negative values). These new parameters are used to propose an improvement in the efficient makespan calculation as stated by Taillard (1990). This improvement is then used for development of a new constructive heuristic based on Ríos-Mercado and Bard (1998b) method NEHT-RB nominated CNIT, and it is compared during computational experimentation with the methods SETUP and TOTAL of Simons (1992), with slight improvements; with the proposal of property UBX from Moccellini and Nagano (2007) and with NEHT-RB method. The methods are then submitted to descent local search as proposed in Ruiz and Stützle (2008) and its performance as initial solutions for this local search procedure is evaluated. Next, an improvement method derivate from the new constructive method and based on metaheuristic IG from Ruiz and Stützle (2008) is proposed and nominated CNIT-IG. This method is compared with original IG submitted to different initial solutions studied during constructive heuristic evaluation. Comparisons are done using Taillard's instances (1990) for standard flowshop and adapted to the flowshop with sequence-dependent setup times problem. The results of computation experimentation are analyzed in terms of average percentage of success, average relative percentage deviation and average computational time and show superiority of new constructive heuristic CNIT-IG and its high computational cost, with complexity mn^3 . The results also show superiority of metaheuristic CNIT-IG over IG method.

Keywords: Production schedule, Permutational Flowshop, sequence dependent setup times, Heuristics, Metaheuristics.

Lista de Símbolos

σ	Seqüência de tarefas programadas
$\sum_{j=1}^n C_j$	Somatório das datas de entrega das tarefas
C_{max}	Duração total da programação (<i>Makespan</i>)
d	Parâmetro das etapas de destruição e construção dos métodos IG e CNIT-IG
$F2$	<i>Flowshop</i> com 2 máquinas
FJc	<i>Jobshop</i> flexível ou híbrido
FFc	<i>Flowshop</i> flexível ou híbrido
Fm	<i>Flowshop</i> com m máquinas
J	Conjunto de tarefas
J_j	Tarefa do conjunto de tarefas
Jm	<i>Jobshop</i> com m máquinas
m	Número de máquinas em um problema de programação
M	Conjunto de máquinas
M_i	Máquina do conjunto de máquinas
n	Número de tarefas em um problema de programação
NP	Classificação de problemas resolvidos em tempo não polinomial
Om	<i>Open shop</i> com m máquinas
op_{ij}	Operação da tarefa j na máquina i
p_{ij}	Tempo de processamento da tarefa j na máquina i
P	Classificação de problemas resolvidos em tempo polinomial
Pm	m máquinas paralelas
Qm	m máquinas paralelas não idênticas (dependente de tarefas)
r_{jk}^i	Fator residual entre as tarefas j e k na máquina i
Rm	m máquinas paralelas não idênticas (dependente de máquinas)

S_{DS}, S_{SD}	Setup dependente da seqüência
s_{jk}^i	Tempo de <i>setup</i> entre as tarefas j e k na máquina i
T	Parâmetro de temperatura constante dos métodos IG e CNIT-IG
$\bar{T}$	Atraso (<i>Tardiness</i>) médio
U	Conjunto de tarefas não programadas
UBX_{jk}^i	Limitante superior da ociosidade da máquina i entre as tarefas j e k
$\sum w_j f_j$	Soma ponderada dos tempos de fluxo
$\sum w_j T_j$	Soma ponderada dos atrasos

Lista de Siglas

<i>ACO</i>	<i>Ant Colony Optimization</i>
<i>ATSP</i>	<i>Asymmetric Travelling Salesman Problem</i>
<i>B&B</i>	<i>Branch & Bound</i>
<i>CM</i>	<i>Cellular Manufacturing</i>
<i>DOE</i>	<i>Design of Experiments</i>
<i>FIFO</i>	<i>First In First Out</i>
<i>GRASP</i>	<i>Greedy Adaptive Search Procedure</i>
<i>GT</i>	<i>Group Technology</i>
<i>IA</i>	<i>Immune Algorithm</i>
<i>IG</i>	<i>Iterated Greedy</i>
<i>JIT</i>	<i>Just in Time</i>
<i>LPT</i>	<i>Longest Processing Time</i>
<i>MILP</i>	<i>Mixed Integer Linear Programming</i>
<i>MIP</i>	<i>Mixed Integer Programming</i>
<i>nwt</i>	<i>No wait</i>
<i>OPT</i>	<i>Optimized Production Technology</i>
<i>prmu</i>	<i>Permutacional</i>
<i>rcrc</i>	<i>Recirculação</i>
<i>RKGA</i>	<i>Random Key Genetic Algorithm</i>
<i>SDST</i>	<i>Sequence-dependent Setup Times</i>
<i>SI</i>	<i>Savings Index</i>
<i>TSP</i>	<i>Travelling Salesman Problem</i>
<i>VAM</i>	<i>Vogel's Approximation Method</i>
<i>WWW</i>	<i>World Wide Web</i>

Lista de Figuras

Figura 1.1 – Relação entre as classes de problemas. Adaptado de Moccelin e Nagano (2003)	22
Figura 3.1 - Exemplo de ociosidade na máquina $i+1$, representado na figura por $X_{uv}^{i+1} > 0$	43
Figura 3.2 – Exemplo de bloqueio da tarefa J_v entre as máquinas i e $i+1$, representado na figura por $Y_{uv}^i > 0$	43
Figura 3.3 – Exemplo de relação entre duas tarefas j e k com $r_{jk}^i > 0$	45
Figura 3.4 – Exemplo de relação entre duas tarefas j e k com $r_{jk}^i < 0$	46
Figura 3.5 – Exemplo de relação entre duas tarefas j e k com $r_{jk}^i = 0$	46
Figura 3.6 – Exemplo de uma programação de 3 tarefas em 2 máquinas	49
Figura 5.1 – Porcentagem de sucesso das heurísticas construtivas	71
Figura 5.2 – Desvio relativo médio das heurísticas em relação ao BM (médias e por classe de problemas)	77
Figura 5.3 – Tempos médios de computação das heurísticas construtivas	77
Figura 5.4 – Porcentagem de sucesso para as soluções com busca local, considerando as médias (barras) e mínimos (linhas)	84
Figura 5.5 – Desvio relativo médio em relação ao BM para a busca local (médias, mínimos e por classe de problema)	92
Figura 5.6 – Tempos médios computacionais para a busca local	93
Figura 5.7 – Porcentagem de sucesso para as diferentes soluções iniciais na meta-heurística IG com $t = 30$, considerando médias e mínimos	100
Figura 5.8 – Desvio relativo médio em relação ao BM das soluções IG com $t = 30$, considerando médias, mínimos e por classe de problemas	105
Figura 5.9 – Porcentagem de sucesso para a solução IG com $t = 60$ considerado as médias e os mínimos	110
Figura 5.10 – Desvio relativo médio em relação ao BM das soluções IG com $t = 60$ considerando médias, mínimos e por classe de problemas	115
Figura 5.11: Desvio relativo médio em relação ao BM das soluções CNIT-IG considerando as médias, mínimos e por classe de problemas ..	121

Figura 5.12: Desvio relativo médio em relação ao BM das soluções selecionadas IG e CNIT-IG com $t = 30$, considerando as médias, mínimos e por classe de problemas 127

Lista de Tabelas

Tabela 3.1 – Exemplo numérico de um problema $F2/S_{DS}, prmu/C_{max}$	56
Tabela 3.2 – Fatores residuais para o problema da tabela 3.1	57
Tabela 3.3 – Valores de <i>Makespan atual</i> da primeira tarefa a ser inserida do problema da tabela 3.1	57
Tabela 3.4 – Valores de <i>Makespan atual</i> da segunda tarefa a ser inserida do problema da tabela 3.1	58
Tabela 3.5 – Valores de <i>Makespan atual</i> da terceira tarefa a ser inserida do problema da tabela 3.1	59
Tabela 3.6 – Valores de <i>Makespan atual</i> da quarta tarefa a ser inserida do problema da tabela 3.1	59
Tabela 5.1 – Porcentagem de sucesso das heurísticas construtivas para o grupo SSD_10	68
Tabela 5.2 – Porcentagem de sucesso das heurísticas construtivas para o grupo SSD_50	69
Tabela 5.3 – Porcentagem de sucesso das heurísticas construtivas para o grupo SSD_100	70
Tabela 5.4 – Porcentagem de sucesso das heurísticas construtivas para o grupo SSD_125	71
Tabela 5.5 – Desvio relativo médio das heurísticas construtivas em relação ao <i>BM</i> para o grupo SSD_10	73
Tabela 5.6 – Tempo médio de computação das heurísticas construtivas para o grupo SSD_10	73
Tabela 5.7 – Desvio relativo médio das heurísticas construtivas em relação ao <i>BM</i> para o grupo SSD_50	74
Tabela 5.8 – Tempo médio de computação das heurísticas construtivas para o grupo SSD_50	74
Tabela 5.9 – Desvio relativo médio das heurísticas construtivas em relação ao <i>BM</i> para o grupo SSD_100	75
Tabela 5.10 – Tempo médio de computação das heurísticas construtivas para o grupo SSD_100	75

Tabela 5.11 – Desvio relativo médio das heurísticas construtivas em relação ao <i>BM</i> para o grupo SSD_125	76
Tabela 5.12 – Tempo médio de computação das heurísticas construtivas para o grupo SSD_125	76
Tabela 5.13 – Porcentagem de sucesso para a solução com busca local do grupo SSD_10, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)	80
Tabela 5.14 – Porcentagem de sucesso para a solução com busca local do grupo SSD_50, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)	81
Tabela 5.15 – Porcentagem de sucesso para a solução com busca local do grupo SSD_100, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)	82
Tabela 5.16 – Porcentagem de sucesso para a solução com busca local do grupo SSD_125, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)	83
Tabela 5.17 – Desvio relativo médio em relação ao <i>BM</i> (média na primeira linha, em negrito e mínimo na segunda) para a busca local do grupo SSD_10	85
Tabela 5.18 – Tempo médio computacional para a busca local do grupo SSD_10	86
Tabela 5.19 – Desvio relativo médio em relação ao <i>BM</i> (média na primeira linha, em negrito e mínimo na segunda) para a busca local do grupo SSD_50	87
Tabela 5.20 – Tempo médio computacional para a busca local do grupo SSD_50	88
Tabela 5.21 – Desvio relativo médio em relação ao <i>BM</i> (média na primeira linha, em negrito e mínimo na segunda) para a busca local do grupo SSD_100	89
Tabela 5.22 – Tempo médio computacional para a busca local do grupo SSD_100	90
Tabela 5.23 – Desvio relativo médio em relação ao <i>BM</i> (média na primeira linha, em negrito e mínimo na segunda) para a busca local do grupo SSD_125	91
Tabela 5.24 – Tempo médio computacional para a busca local do grupo SSD_125	92
Tabela 5.25 – Porcentagem de sucesso para as diferentes soluções iniciais na meta-heurística IG com $t = 30$ para o grupo SSD_10, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)	96

Tabela 5.26 – Porcentagem de sucesso para as diferentes soluções iniciais na meta-heurística IG com $t = 30$ para o grupo SSD_50, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)	97
Tabela 5.27 – Porcentagem de sucesso para as diferentes soluções iniciais na meta-heurística IG com $t = 30$ para o grupo SSD_100, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)	98
Tabela 5.28 – Porcentagem de sucesso para as diferentes soluções iniciais na meta-heurística IG com $t = 30$ para o grupo SSD_125, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)	99
Tabela 5.29 – Desvio relativo médio em relação ao <i>BM</i> das soluções IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_10	101
Tabela 5.30 – Desvio relativo médio em relação ao <i>BM</i> das soluções IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_50	102
Tabela 5.31 – Desvio relativo médio em relação ao <i>BM</i> das soluções IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_100	103
Tabela 5.32 – Desvio relativo médio em relação ao <i>BM</i> das soluções IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_125	104
Tabela 5.33 – Porcentagem de sucesso para a solução IG com $t = 60$ do grupo SSD_10, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)	106
Tabela 5.34 – Porcentagem de sucesso para a solução IG com $t = 60$ do grupo SSD_50, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)	107
Tabela 5.35 – Porcentagem de sucesso para a solução IG com $t = 60$ do grupo SSD_100, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)	108
Tabela 5.36 – Porcentagem de sucesso para a solução IG com $t = 60$ do grupo SSD_125, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)	109
Tabela 5.37 – Desvio relativo médio em relação ao <i>BM</i> das soluções IG com $t = 60$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_10	111

Tabela 5.38 – Desvio relativo médio em relação ao <i>BM</i> das soluções IG com $t = 60$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_50	112
Tabela 5.39 – Desvio relativo médio em relação ao <i>BM</i> das soluções IG com $t = 60$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_100	113
Tabela 5.40 – Desvio relativo médio em relação ao <i>BM</i> das soluções IG com $t = 60$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_125	114
Tabela 5.41. Desvio relativo médio em relação ao <i>BM</i> das soluções CNIT-IG (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_10	117
Tabela 5.42. Desvio relativo médio em relação ao <i>BM</i> das soluções CNIT-IG (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_50	118
Tabela 5.43. Desvio relativo médio em relação ao <i>BM</i> das soluções CNIT-IG (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_100	119
Tabela 5.44. Desvio relativo médio em relação ao <i>BM</i> das soluções CNIT-IG (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_125	120
Tabela 5.45. Desvio relativo médio em relação ao <i>BM</i> das soluções selecionadas IG e CNIT-IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_10	123
Tabela 5.46. Desvio relativo médio em relação ao <i>BM</i> das soluções selecionadas IG e CNIT-IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_50	124
Tabela 5.47. Desvio relativo médio em relação ao <i>BM</i> das soluções selecionadas IG e CNIT-IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_100	125
Tabela 5.48. Desvio relativo médio em relação ao <i>BM</i> das soluções selecionadas IG e CNIT-IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_125	126

Sumário

1. Introdução	19
1.1.Programação de Operações	20
1.2.Métodos de Solução	24
1.3.Objetivos e Estrutura deste Trabalho	25
2. <i>Floswhop</i> com tempos de <i>setup</i> dependentes da seqüência	27
2.1.Métodos de Solução Exata	30
2.2.Métodos Heurísticos	33
2.3.Métodos Meta-heurísticos	37
2.4.Considerações	39
3. Proposição do método heurístico e do método meta-heurístico	41
3.1.Notação	41
3.2.Os conceitos de Ociosidade de máquina e Bloqueio de tarefas	42
3.3.Investigação do fator residual e da propriedade UBX	44
3.4.Desenvolvimento do parâmetro <i>XR</i>	47
3.5.O cálculo eficiente de Makespan e o parâmetro <i>QR</i>	51
3.6.A nova heurística CNIT	53
3.7.Exemplo numérico	55
3.8.A nova meta-heurística CNIT-IG	60
4. Experimentação computacional, resultados e análise	63
4.1.Delineamento do experimento	63
4.1.1. Banco de dados	64
4.1.2. Processo de análise dos resultados	65
4.1.2.1.Porcentagem de sucesso	65

4.1.2.2.Desvio relativo médio	66
4.1.2.3.Tempo médio de computação	67
4.2.Análise dos resultados	67
4.2.1. Avaliação das heurísticas construtivas	68
4.2.2. Avaliação da Busca Local Descendente sobre as soluções das heurísticas construtivas	78
4.2.3. A avaliação do efeito da solução inicial para a meta-heurística IG e sua comparação com a nova meta-heurística CNIT-IG	94
4.2.3.1.Análise dos resultados da nova meta-heurística CNIT-IG	115
5. Conclusões	129
REFERÊNCIAS	133
APÊNDICE – Código Fonte do programa computacional	143

1. Introdução

Um sistema de produção tem por objetivo transformar insumos nos produtos e serviços de uma organização. Assim, de acordo com Gaither e Frazier (2005), um sistema de produção transforma insumos – matérias primas, pessoal, máquinas, prédios, tecnologia, dinheiro, informação e outros recursos – em saídas – produtos e serviços. Segundo Slack et al. (1999), a produção deve ser o foco central para uma organização, porque é responsável por produzir bens e serviços que são a razão de sua existência.

Os Sistemas de Produção podem ser classificados, segundo Johnson e Montgomery (1974), em:

- a) Sistema Contínuo, onde poucos tipos de produtos similares são fabricados em grande volume;
- b) Sistema Intermitente, onde ocorrem freqüentes mudanças de um produto para outro nos estágios produtivos, como consequência de uma grande variedade de produtos fabricados. Nesta classe distinguem-se dois tipos de sistemas:
 - Intermitente *Flowshop*, onde os itens fabricados em uma linha ou célula de manufatura têm a mesma seqüência de operações nas diversas máquinas;
 - Intermitente *Jobshop*, onde os itens fabricados em um setor produtivo não têm o mesmo roteiro de fabricação;
- c) Sistema Grande Projeto, onde são produzidos itens complexos e especiais, muitas vezes únicos.

Dentro do Sistema Intermitente, um importante nível de decisão é a programação de operações, que oferece a abertura a grandes ganhos em tempo e aproveitamento da capacidade instalada.

O enfoque hierárquico clássico que consiste de múltiplos níveis (usualmente três: estratégico, tático e operacional), possui as seguintes características, de acordo com Dauzère-Pérès e Lasserre (2002):

- Quanto mais alto na hierarquia, mais estratégicas são as decisões;
- Quanto mais alto na hierarquia, mais complexos são os modelos e maiores os horizontes de planejamento;
- As decisões em algum nível tornam-se uma restrição ou um objetivo a ser satisfeito em níveis mais baixos;
- Cada nível de decisão tem seus próprios modelos de decisão e procedimentos de solução.

A teoria associada às decisões de nível operacional relacionadas à alocação de recursos às tarefas em intervalos de tempo definidos e seu objetivo de otimizar uma ou mais funções de desempenho é chamada de Programação de Operações, ou seu termo em inglês *Scheduling*. Além da alocação de recursos produtivos, como máquinas, mão-de-obra e matérias primas, este processo de decisão pode ainda ser estendido para outros sistemas, tais como corredores em aeroportos, equipes de construção, unidades de processamento computacional. As tarefas, além das já descritas operações em sistemas produtivos, podem ser pousos e decolagens em aeroportos, estágios em uma construção, execução de programas de computador (PINEDO, 2008). Por fim, os objetivos podem ser diversos, porém em geral estão associados à melhor utilização dos recursos (eficiência), o rápido atendimento às demandas e conformidade com os prazos estabelecidos (BAKER, 1974).

1.1.Programação de Operações

De acordo com Pinedo (2008), a programação refere-se à ordenação de tarefas a serem executadas, em uma ou diversas máquinas, considerando-se uma base de tempo, ou seja, determinando-se, principalmente, as datas de início e fim de cada tarefa.

Tal problema, representado através da notação descrita por Pinedo (2008), utiliza o padrão de três campos $\alpha/\beta/\gamma$, em que α representa o ambiente de máquinas, β fornece os detalhes das características de processamento e restrições e pode estar vazio ou conter um ou múltiplos parâmetros; o terceiro campo, γ , contém a função objetivo.

Os problemas de programação de operações em máquinas podem ser complexos, não apenas pela natureza combinatória do problema, mas também pelas restrições dadas pelo sistema. Nesses problemas, as restrições tecnológicas e a medida de desempenho da programação devem ser especificadas e estas restrições são determinadas principalmente pelo fluxo das tarefas nas máquinas. Neste contexto, Maccarthy e Liu (1993) classificaram os problemas de programação de operações da seguinte forma:

- Máquina simples ou única (1): o ambiente mais simples, em que há apenas uma máquina para processar todas as tarefas;
- Idênticas máquinas em paralelo (Pm): nesta situação existem m máquinas idênticas em paralelo. A tarefa j precisa de uma única operação e pode ser processada em qualquer uma das m máquinas. O cenário em que as máquinas paralelas não são idênticas, ou seja, os tempos de processamento são diferentes, é representado por Qm se os tempos dependem das tarefas e das máquinas e por Rm se dependem apenas das máquinas. Por fim, se um subgrupo de tarefas exigirem seu processamento em máquinas específicas, esta informação estará no campo β .
- *Flowshop* (Fm): Existem m máquinas em série. Cada tarefa deve ser processada em cada uma das m máquinas seguindo a mesma rota, ou seja, sendo processada primeiro na máquina 1, em seguida na máquina 2 e assim por diante. Quando a sequência “primeiro a entrar, primeiro a sair” (*FIFO* – do inglês *first in, first out*) é observada em todas as máquinas, o *flowshop* é dito permutacional e esta informação pode aparecer no campo β como *prmu*. Quando existem máquinas paralelas em um ou mais estágios do *flowshop* ele é dito flexível ou híbrido (*FFc*).
- *Jobshop* (Jm): Existem m máquinas e cada tarefa segue sua própria sequência pré-determinada de processos. Se as tarefas devem retornar a uma máquina já visitada anteriormente, o campo β recebe *rcrc* de recirculação. Como no caso anterior, se existem mais máquinas em um ou mais estágios, ele é dito flexível ou híbrido (*FJc*).
- *Openshop* (Om): Existem m máquinas e todas as tarefas são processadas em todas as m máquinas, porém cada uma das tarefas é processada em qualquer ordem e o tempo de processamento pode ser zero.

A figura a seguir ilustra a relação entre os tipos de programação:

Figura 1.1: Relação entre as classes de problemas. Adaptado de Moccelin e Nagano (2003)

De acordo com Taillard (1993), o problema de *flowshop* pode ser entendido como um conjunto de n tarefas $J = \{J_1, J_2, \dots, J_n\}$ a serem processadas em um conjunto de m máquinas $M = \{M_1, M_2, \dots, M_m\}$. O processamento da tarefa J_j na máquina M_i é denominado como operação e designado por op_{ij} . O tempo de processamento associado à operação op_{ij} é designado por p_{ij} .

Existem diversos fatores que fazem da programação da produção um problema difícil. Uma característica preeminente do problema é a sua natureza combinatória, o que significa que o número de possíveis soluções cresce exponencialmente em várias dimensões de acordo com a quantidade de tarefas, operações, máquinas, ferramentas, pessoal, etc. Como exemplo, apresentado por Rickel (1988), a programação de 12 tarefas com 6 operações tem $(12!)^6$ possíveis soluções em um *jobshop* tradicional, sem

máquinas alternativas. Esta situação piora quando existem máquinas alternativas que realizam a mesma operação, ou máquinas que podem realizar diversas operações.

Segundo Yang e Liao (1999), os problemas de programação podem ser classificados como P , se são resolvidos eficientemente em tempo polinomial e NP caso não possuam métodos para obtenção da solução ótima em tempo polinomial. Os problemas de programação *flowshop* com mais de 3 operações e o de *jobshop* com mais de 2 operações em máquinas distintas pertencem à classe de problemas classificados como *NP-completos* (GAREY; JOHNSON, 1979 e RICKEL, 1988).

As suposições comumente consideradas em problema de programação *flowshop* (BAKER, 1974 e PINEDO, 2008) impedem várias aplicações práticas importantes (DUDEK; PANWALKAR; SMITH, 1992). Uma destas suposições é a não consideração de tempos de *setup*, os quais são incluídos nos tempos de processamento das tarefas. Esta suposição pode ser válida quando tempos de *setup* são pequenos, independentes da sequência de processo e inseparáveis dos tempos de processamento. Entretanto, este não é o caso para muitos ambientes de produção.

Neste trabalho, o problema de programação em ambiente *flowshop* com n tarefas e m máquinas será pesquisado com tempos de *setup* assimétricos e dependentes da sequência. Este problema é conhecido na literatura por sua sigla em inglês *SDST Flowshop* (*Sequence-dependent Setup Time Flowshop*) que será utilizado aqui por simplicidade.

O critério de desempenho utilizado será a duração total da programação (*Makespan* ou C_{max}), ou seja, o intervalo de tempo entre o início de execução da primeira tarefa na primeira máquina e o término de execução da última tarefa na última máquina. Este critério de desempenho visa a melhor utilização do sistema.

A notação deste problema, de acordo com Pinedo (2008) é definida por $Fm/SDS, pmu/C_{max}$.

1.2.Métodos de Solução

Existem basicamente dois tipos de métodos desenvolvidos para resolver os problemas de programação: os métodos de solução ótima e os métodos heurísticos.

Os métodos de solução ótima podem ainda ser divididos, segundo MacCarthy e Liu (1993) em (i): métodos de ótimos eficientes: estes métodos geram uma programação ótima em relação a um critério de programação em tempo polinomial, ou seja, o tempo para o método encontrar uma solução ótima é função do número de máquinas e tarefas; e (ii) métodos de ótimos enumerativos, como os métodos *Branch & Bound* (*B&B*), em que as potenciais soluções parciais são numeradas e são excluídas as que não obedecem a determinado critério de corte. Métodos destes tipos são encontrados apenas para uma pequena classe de problemas, já que problemas mais gerais levam a métodos mais complexos. Ou ainda, suas soluções são computacionalmente dispendiosas, tornando-se impraticáveis para problemas maiores. Tais métodos, no entanto, permitem um melhor entendimento dos problemas e o desenvolvimento de relações de dominância, limites inferiores e superiores, e o desenvolvimento dos métodos heurísticos (SETHANAN; 2001).

Os métodos heurísticos são aqueles que levam a soluções viáveis através de avaliações baseadas nas características do problema ou ainda com base em critérios computacionais, mas sem percorrer todas as soluções possíveis. O objetivo é encontrar boas soluções (viáveis, porém não necessariamente ótimas) em tempo computacional razoável. Os métodos heurísticos classificam-se de modo geral, conforme Souza e Moccellini (2000), em métodos construtivos e métodos melhorativos ou meta-heurísticas. Segundo Nagano (1999), os métodos construtivos obtêm soluções únicas diretamente a partir do problema de programação, utilizando procedimentos que designam índices de prioridade para construir a solução. Os métodos melhorativos partem de uma solução inicial e procuram por uma solução melhor em sua vizinhança.

1.3.Objetivos e Estrutura deste Trabalho

Este trabalho integra a Linha de Pesquisa designada “Pesquisa Operacional Aplicada aos Sistemas de Produção”, pertencente ao Programa de Pós-Graduação do Departamento de Engenharia de Produção da Escola de Engenharia de São Carlos – Universidade de São Paulo.

O principal tema de pesquisa desta Linha é a “Programação de Operações em Sistemas de Operações Intermitentes”, que consiste no estudo de modelos e desenvolvimento de novos métodos de solução.

Neste contexto, este trabalho tem por objetivo principal investigar o problema de programação de operações em *Flowshop* com m máquinas, com apenas a restrição adicional de tempos de setup das máquinas, para as tarefas, separados dos tempos de processamento e dependentes da seqüência de execução das tarefas, buscando propor dois métodos: um método heurístico construtivo e outro meta-heurístico para a minimização do *Makespan*.

O texto está estruturado em cinco capítulos. No primeiro capítulo, apresenta-se uma introdução sobre os problemas de programação. No segundo capítulo apresenta-se a revisão de literatura das pesquisas realizadas em programações de operações em *Flowshop* envolvendo considerações de tempo de setup das máquinas para as tarefas. O terceiro capítulo constará da ambientação do problema, apresentação das propriedades do problema considerado, proposição de um novo método heurístico construtivo e um meta-heurístico e apresentação de métodos existentes na literatura. A experimentação computacional será detalhada no quarto capítulo. No quinto capítulo estarão as considerações finais do trabalho.

2. *Flowshop* com tempos de *setup* dependentes da seqüência

De acordo com Gupta e Stafford (2006), o *flowshop* é caracterizado por um fluxo de tarefas mais ou menos contínuo e ininterrupto através de múltiplas máquinas em série. Nele, o fluxo de trabalho é unidirecional já que todas as tarefas seguem a mesma seqüência tecnológica em todas as máquinas.

O primeiro artigo publicado sobre o problema de *flowshop* é o de Johnson (1954), sobre o problema $F2/C_{max}$. Para o ambiente com duas máquinas ($m = 2$) as seqüências permutacionais são dominantes com respeito a qualquer medida de desempenho regular. Quaisquer outras generalizações, como o aumento do número de máquinas, a mudança no critério de desempenho ou a consideração de outras restrições, como a existência de tempo de *setup*, por exemplo, podem fazer com que esta característica não seja verdadeira. Porém, devido à complexidade da consideração de seqüências não permutacionais, a vasta maioria da literatura sobre o *flowshop* e suas variações considera apenas seqüências permutacionais (GUPTA; STAFFORD, 2006).

O método de Johnson (1954) encontra soluções ótimas em tempo polinomial para o problema $F2/C_{max}$. Dada a natureza dos problemas mais gerais de *flowshop* ser *NP-completo*, os principais métodos desenvolvidos para a solução do problema são heurísticos, como CDS de Campbell, Dudek e Smith (1970), e o método NEH de Nawaz, Ensore Jr. e Ham (1983); e meta-heurísticos, como o *Simulated Annealing* de Osman e Potts (1989), busca Tabu de Widmer e Hertz (1989) e o Algoritmo Genético de Reeves (1995). Uma revisão dos problemas de *flowshop* nos 50 anos após a publicação de Johnson (1954) pode ser encontrada em Gupta e Stafford (2006), e em Hejazi e Saghafian (2005), uma revisão de *flowshop* com *Makespan* como critério de otimização.

A maioria dos trabalhos, no entanto assume que os tempos/custos de troca e preparação (*setup*) são insignificantes ou fazem parte do processo. Isso só é verdade se estes tempos não dependerem da seqüência de tarefas, ou se é necessário esperar que a tarefa chegue à máquina para realizar a preparação, do contrário, não considerar os tempos de *setup* separadamente afeta diretamente a qualidade das soluções

(ALLAHVERDI; GUPTA; ALDOWAISAN, 1999 e 2008, YANG; LIAO, 1999; CHENG; GUPTA; WANG, 2000).

As operações de troca de produtos (e os ajustes associados a estas trocas), realizadas nas máquinas e que não estão estritamente relacionadas com o processamento das tarefas possuem um tempo denominado “tempo de *setup*”. Tais atividades consistem da configuração do equipamento para o próximo trabalho, a obtenção de ferramentas, limpeza das máquinas, posicionamento da matéria-prima, ajustes, inspeções ou muitas outras atividades. Da mesma forma, existem custos associados a estas operações de troca. Estes custos podem estar explícitos para o problema, mas em geral são considerados proporcionais aos tempos de *setup*.

De acordo com Allahverdi e Soroush (2008), existe um interesse crescente na consideração de tempos/custos de *setup* separados, pois há imensas economias quando as atividades de *setup* são explicitamente incorporadas nas decisões de programação de produção. Allahverdi, Gupta e Aldowaisan (1999) destacaram que a consideração do tempo de *setup* separadamente do tempo de processamento é inerente às recentes filosofias de gerenciamento de produção tais como *Just-in-time (JIT)*, a tecnologia de produção otimizada (*OPT*), tecnologias de grupo (*GT*), manufatura celular (*CM*) e competição baseada no tempo. Em geral, nestes casos o objetivo é realizar as operações simultaneamente e melhorar o desempenho. Allahverdi, Gupta e Aldowaisan (2008) listaram exemplos recentes que mostram a importância e aplicações de modelos de programação com consideração explícita dos tempos (custos) de *setup*. A seguir estão algumas:

- Laguna (1999) considerou uma fábrica que produz suprimentos para fotocopadoras e impressoras a laser. Ele mostrou que para a mudança de um *toner* para outro o tempo de mudança é da ordem de dias.
- Schaller, Gupta e Vakharia (2000) afirmaram que o problema de fabricar placas de circuito impresso em uma máquina de inserção automática é um problema de programação com tempos de *setup*.
- Gendreau, Laporte e Guimarães (2001) mostraram que a troca do tecido na indústria têxtil possui tempo de *setup* que depende do tecido atual e do próximo a ser processado.

- Segundo Allahverdi e Al-Anzi (2002), aplicações *World Wide Web* (WWW) de distribuição de dados multimídia podem ser modeladas como um problema *flowshop* com duas máquinas e tempos de *setup* separados.
- Tempos de *setup* dependentes também são observados na produção de componentes semicondutores em Kim et al. (2002).
- Chang, Hsieh e Wang (2003) descreveram uma fábrica de produtos plásticos como fitas adesivas, álbuns de fotografias, embalagens de alimentos, capas de livros em que os tempos e custos *setup* entre duas tarefas eram dependentes da sequência.
- A troca de pesados moldes em estamparias e sua significância foi considerada por Yi e Wang (2003).
- Lin e Liao (2003) descreveram o ambiente de um fabricante de etiquetas auto-adesivas como um problema de *flowshop* híbrido com dois estágios onde o primeiro estágio é uma máquina única de alta velocidade que aplica a cola e o *liner* na superfície do material. Tempos de *setup* dependentes da sequência ocorrem quando há a troca para diferentes “classes” de tarefas.

Como observado na revisão acima, os tempos de *setup* podem depender ou não da sequência de tarefas. Por isso, existem dois tipos de *setup*: se ele depender somente da tarefa a ser processada, independente da tarefa anterior, ele é dito tempo de *setup* independente da sequência; o segundo tipo é o tempo de *setup* dependente da sequência, quando depende tanto da tarefa a ser processada, quanto da anterior (ALLAHVERDI; GUPTA; ALDOWAISAN, 1999 e 2008, YANG; LIAO, 1999; CHENG; GUPTA; WANG, 2000).

O ambiente neste trabalho é o *flowshop* com tempos de *setup* assimétricos e dependentes da sequência (*SDST Flowshop*). Esta modelagem é aplicável em muitos casos reais onde máquinas multi-propósito são utilizadas para fabricar produtos diferentes (RUIZ; MAROTO; ALCARAZ, 2005). Exemplos comuns podem ser encontrados nas indústrias química, farmacêutica, e de plásticos. Na indústria de tintas, a limpeza dos equipamentos pode demandar menos tempo caso a sequência de produção envolva a troca de uma tinta mais clara para uma mais escura e mais tempo caso contrário. Na indústria de peças plásticas injetadas, as peças podem possuir a mesma forma, mas diferentes materiais, exigindo a preparação para troca da matéria prima, ou

podem possuir também formatos pouco diferentes, exigindo a troca de partes do ferramental, ou ainda serem totalmente diferentes, exigindo a troca completa do ferramental. É óbvio que quanto mais atividades envolvidas com a troca, maior o tempo (custo) de *setup*. Allahaverdi e Soroush (2008) mostraram a importância da consideração explícita dos tempos de *setup* em vários casos e os benefícios de um trabalho específico de redução destes tempos para os sistemas de produção. Estes benefícios são: redução de gastos, aumento na velocidade de produção, aumento na produção, redução de *lead times*, trocas mais rápidas, aumento da lucratividade e satisfação, permitindo uma manufatura enxuta, fluxos suaves, maior variação nos tamanhos de lote, curva de custo total menor, menor número de situações sem estoque, menores níveis de estoque, menores tamanho de estoque para pedido de reposição, margens maiores em ordens acima do mínimo, entregas mais rápidas e aumento da satisfação do cliente.

Apesar de o problema *SDST Flowshop* diferir do problema de *flowshop* permutacional apenas pela consideração dos tempos de *setup* separados e dependentes da seqüência, o primeiro é muito mais complexo que o segundo. Gupta (1986) demonstrou que o problema *SDST* é *NP-Completo* para o caso com apenas uma máquina ($m = 1$). Os métodos utilizados para a solução do problema *SDST Flowshop* devem possuir adaptações específicas para o ambiente, ainda que as semelhanças sugiram aplicações diretas de métodos desenvolvidos para o ambiente *flowshop* permutacional.

As próximas seções são dedicadas às revisões dos métodos de solução encontrados na literatura para o problema *SDST Flowshop*, divididos por: métodos de solução exata, métodos heurísticos e métodos meta-heurísticos. A revisão foca principalmente os trabalhos com critério de minimização do *makespan*.

2.1. Métodos de Solução Exata

Corwin e Esogbue (1974) desenvolveram um algoritmo baseado em programação dinâmica para o problema *SDST flowshop* com $m = 2$, onde os tempos de

setup são dependentes da seqüência somente em uma das máquinas, e independentes da seqüência na outra máquina. Os algoritmos propostos somente resolveram de maneira ótima problemas entre 12 e 14 tarefas.

Uskup e Smith (1975) desenvolveram um algoritmo *B&B* para o mesmo problema de duas máquinas, porém com a existência de datas de entrega. O critério de otimização considerado é a minimização dos custos de *setup* onde estes são diretamente proporcionais aos tempos de *setup*. O algoritmo foi capaz de resolver problemas de até 30 tarefas, porém unicamente no caso de duas máquinas. A aplicação dos métodos *B&B* a problemas com tempos de *setup* dependentes da seqüência não se limitou ao problema de *flowshop*.

Gupta (1975) apresentou um algoritmo de otimização mediante busca lexicográfica. O método apresentado é muito geral e permite incorporar numerosas restrições e condições ao problema, entre elas os tempos de *setup* dependentes da seqüência. O critério que se busca otimizar neste trabalho é uma função completa do custo total de oportunidade, da qual se pode derivar o *Makespan*.

Srikan e Ghosh (1986) desenvolvem uma formulação em programação linear mista (*MILP- Mixed Integer Linear Program*) para o caso de minimização do *Makespan* com m máquinas. O modelo proposto necessitou menor número de variáveis do que as formulações baseadas no *TSP (Traveling Salesman Problem – Problema do Caixeiro Viajante)* sendo, portanto, mais rápido. Entretanto, os autores não puderam resolver de maneira ótima problemas maiores do que seis tarefas e seis máquinas.

O caso especial de *setup* dependente da seqüência em que os tempos de *setup* são aditivos foi estudado por Szwarc e Gupta (1987). Eles desenvolveram um algoritmo de solução em tempo polinomial para minimização do *makespan* que encontra soluções ótimas para o caso de duas máquinas.

Stafford e Tseng (1990) realizaram correções no modelo de Srikan e Ghosh (1986) e propuseram também modelos para o problema $Fm/S_{sb} \text{ } prmu, nwt/C_{max}$, entre outros. Neste caso, o tamanho máximo de problema que se pôde resolver foi de sete tarefas e cinco máquinas, sendo necessário mais de seis horas de tempo de CPU em um computador PC/AT.

Gupta, Das e Ghosh (1995) desenvolveram um algoritmo *B&B* para a solução do problema $Fm/S_{sd}, prmu/C_{max}$, resolvendo otimamente problemas com 20 tarefas.

Ríos-Mercado e Bard (1996) basearam-se na formulação em programação linear mista (*MILP- Mixed Integer Linear Program*) de Srikar e Ghosh (1986) e desenvolveram várias inequações válidas para as formulações MILP e MIP (*Mixed Integer Programming* – programação inteira mista) para o problema $Fm/S_{sd}/C_{max}$.

Ríos-Mercado e Bard (1998a) utilizaram a programação MIP para o problema $Fm/S_{sd}/C_{max}$ em duas formulações diferentes: uma baseada no problema do caixeiro viajante e outra menos estruturada que a primeira. Elas foram avaliadas dentro da estrutura *branch-and-cut* que se mostrou superior ao algoritmo *B&B*. Os resultados computacionais mostraram que a segunda formulação é mais rápida que a primeira.

Ríos-Mercado e Bard (1999a) apresentaram um novo algoritmo *B&B* para o problema $Fm/S_{sd}/C_{max}$ incorporando limites inferiores e superiores e critérios de eliminação. Os resultados computacionais mostraram que ele é superior aos algoritmos baseados em programação linear devido ao critério de eliminação utilizado, que propicia economia de tempo.

Sethanan (2001) abordou o problema *SDST flowshop* flexível, com uma ou mais máquinas em cada estágio, *setup* dependente da sequência em todas as máquinas e objetivo de minimização de *makespan*. Seu modelo matemático resolveu problemas pequenos, com até 5 tarefas, 5 estágios e 8 máquinas.

Stafford e Tseng (2001) propuseram dois modelos *MILP* (*Mixed-integer linear programming* – Programação linear inteira mista) para o *SDST flowshop* e para o problema $F/S_{sd}, prmu, nwt/C_{max}$. Em ambos os casos os autores puderam resolver problemas de até nove tarefas e nove máquinas em aproximadamente cinco minutos de tempo de CPU em um computador do tipo PC/AT com um processador de 800 MHz. Stafford e Tseng (2002) apresentaram outros dois modelos similares, com algumas melhorias em relação aos anteriores, mas sem conseguir resolver problemas de tamanho maior.

Ríos-Mercado e Bard (2003) estudaram a estrutura poliédrica de duas formulações diferentes de programação inteira mista. Uma delas relaciona-se com o

problema do caixeiro viajante (*TSP*) e a outra derivada de um modelo proposto anteriormente. Ambas foram avaliadas usando algoritmo *branch-and-cut*, que mostrou que o modelo baseado no *TSP* é inferior em termos de tempo computacional.

Crowder (2006) também abordou o problema *SDST flowshop* flexível, com limite de estocagem entre as máquinas e objetivo de minimização de *makespan*. Seu modelo matemático conseguiu solucionar problemas pequenos com alto tempo computacional.

Eren (2010) propôs um modelo de programação inteira para o problema *SDST Flowshop* bicritério: *makespan* e soma ponderada das datas de término. O modelo conseguiu resolver problemas com 18 tarefas e 6 máquinas.

2.2. Métodos Heurísticos

Gupta e Darrow (1986) propuseram quatro métodos heurísticos para o problema com duas máquinas. Neste estudo determinou-se que conforme a razão entre os tempos de *setup* e de processamento aumenta, as heurísticas propostas deterioram drasticamente. Os autores propuseram dois métodos heurísticos baseados na regra de Johnson (1954) e outros dois métodos baseados nos dois primeiros incorporando uma técnica de busca local.

Szwarc e Gupta (1987) propuseram um algoritmo heurístico para um caso particular do *SDST flowshop* em que se consideraram os tempos de *setup* dependentes da sequência, separáveis e aditivos. Por aditivos entende-se que uma parte do tempo depende da tarefa que se está processando e a outra parte da tarefa seguinte na sequência. Os autores apresentam um algoritmo ótimo para o caso $m = 2$ e uma extensão heurística para o caso geral de m máquinas.

Uma das primeiras heurísticas para o caso geral de m máquinas deve-se a Simons (1992). No trabalho apresentado, o autor propôs quatro métodos, dois dos quais sendo modificações dos métodos MINIT e MICOT de Gupta (1972), com adaptações

que levam em consideração a existência de tempos de *setup* dependentes da sequência. As outras duas heurísticas partem do trabalho de Stinson e Smith (1982), em que, através de uma analogia com o problema do caixeiro viajante (*TSP*), é construída uma matriz de custos a partir dos dados de entrada do problema. A solução é então encontrada através da aplicação de uma versão adaptada do algoritmo de Vogel (*VAM – Vogel’s Approximation Method*, desenvolvido originalmente para o problema de transporte), diretamente na matriz de custos. Na primeira destas heurísticas, chamada *TOTAL*, a matriz de custos é formada pela soma de todos os tempos de processamento e de *setup* das tarefas. Na segunda heurística chamada *SETUP*, a matriz de custos contém somente a soma dos tempos de *setup*. Após avaliação, as heurísticas *TOTAL* e *SETUP* superaram facilmente as adaptações dos métodos *MINIT* e *MICOT*.

Das, Gupta e Khumawala (1995) desenvolveram uma heurística baseada em índices. Estes índices selecionam uma tarefa não sequenciada para colocá-la a uma posição de uma sequência parcialmente construída. O índice considera três conceitos: redução no tempo de *setup* na sequência, redução na duração da sequência a partir dos tempos de processamento e redução nas datas de entrega das tarefas. A eficácia desta heurística foi comprovada resolvendo um conjunto de 240 problemas para os quais se conhecia a solução ótima, e calculando-se o desvio da heurística em relação às soluções ótimas. Nestes casos, a heurística proposta nunca se desviou mais de 6 %.

Kim, Lim e Park (1996) apresentaram um extenso estudo sobre um problema real de produção de placas de circuitos impressos (*PCB*). O sistema produtivo é muito específico e pode ser definido como $F / S_{sd}, prmu, nwt / \bar{T}$, ainda que tenha outros aspectos mais concretos para os quais a notação aqui utilizada não abrange. Os autores propuseram várias heurísticas, todas novas, para resolver o problema, entre as quais está uma adaptação da heurística *NEH*, e métodos meta-heurísticos como: *simulated annealing*, busca tabu e técnica de busca local descendente. Dentre todos, os algoritmos baseados em *simulated annealing* apresentaram melhores resultados.

Rajendran e Ziegler (1997a) propuseram três heurísticas para o problema $Fm/S_{sd}, prmu/C_{max}$ e incorporaram a elas um método de melhoria a fim de aumentar a qualidade das soluções. Os métodos foram comparados durante a experimentação computacional com a heurística de duas fases de Proust, Gupta e Deschamps (1991) adaptada para o

problema *SDST Flowshop*. Uma das heurísticas mostrou-se superior às outras na qualidade e tempo computacional das suas soluções. Em outro trabalho, Rajendran e Ziegler (1997b) abordaram o problema $Fm/S_{sd}/\sum w_j f_j$ onde também propuseram uma heurística para o problema e um esquema de melhoria da qualidade da solução da heurística.

Ríos-Mercado e Bard (1998b) propuseram dois métodos para o *SDST flowshop*, um heurístico e o outro meta-heurístico. O método heurístico baseia-se na adaptação do método de inserção do NEH, levando em consideração os tempos de *setup*. Na adaptação, os autores consideraram as melhorias propostas por Taillard (1990) para o cálculo do *Makespan* e chamaram a heurística resultante de NEHT-RB. A proposta meta-heurística se baseia no método GRASP (*Greedy Adaptive Search Procedure*). Os autores compararam estes dois métodos com a heurística SETUP, aplicando uma busca local nos métodos. A heurística SETUP obteve, geralmente, melhores resultados globais, enquanto que no caso onde os tempos de *setup* são menores do que os de processamento, as duas heurísticas propostas, em especial o método GRASP obtiveram resultados melhores, ainda que este último método seja consideravelmente mais lento. Em um estudo posterior, (Ríos-Mercado e Bard, 1999b), os autores desenvolvem um método baseado nas heurísticas de Simons (1992). Este método, chamado de HYBRID, baseia-se em um parâmetro θ ponderador da função custo, isto é, o peso que têm os tempos de *setup* e os tempos de processamento, de maneira que para valores distintos de θ são obtidas seqüências distintas. A função custo para cada par de tarefas foi calculada da seguinte maneira:

$$C_{jk} = \theta \cdot R_{jk} + (1 - \theta) \cdot S_{jk}$$

Onde $\theta \in [0, 1]$ e S_{jk} é a soma dos tempos de *setup* entre as tarefas j e k para toda máquina i , ou seja: $S_{jk} = \sum_{i=1}^m s_{jk}^i$. R_{jk} é o resíduo de programar j antes de k . Nos testes realizados, o método HYBRID supera a heurística SETUP e o método GRASP para valores pequenos de m , enquanto que GRASP obtém melhores resultados para problemas maiores.

Allahverdi e Aldowaisan (2001) resolveram um tipo específico de *SDST flowshop* onde as tarefas não podem esperar entre máquinas e o critério de otimização considerado é a soma das datas de entrega das tarefas, ou seja, o problema considerado

é o F/S_{sd} , $prmu$, $nwt/\sum_{j=1}^n C_j$. Os autores discutiram uma série de métodos que proporcionam a solução ótima no caso de $m = 2$ e quando uma série de condições especiais é cumprida. Também propuseram cinco heurísticas diferentes para o problema de m máquinas. Neste caso, os tempos de *setup* considerados são do tipo aditivo.

Maddux III e Gupta (2003) estudaram o problema $F2/ST_{sd}/C_{max}$ sem estoques entre as máquinas (*buffers* de capacidade zero), e onde algumas tarefas passam apenas pela primeira máquina e algumas tarefas devem ser processadas por ambas as máquinas. Eles desenvolveram limitantes inferiores e apresentaram limitantes inferiores e uma heurística para o problema.

Rajendran e Ziegler (2003) estudaram o problema *SDST Flowshop* com uma combinação de dois objetivos considerados por Rajendran e Ziegler (1997b) e Sonmez e Baykasoglu (1998), ou seja, $Fm/S_{sd}/\sum w_j T_j + \sum w_j f_j$ (soma ponderada dos atrasos e soma ponderada dos tempos de fluxo). Eles propuseram heurísticas e as compararam com a heurística de Gelders e Sambandan (1978), com um procedimento de busca aleatória e com uma busca local gulosa. Os resultados mostram que a heurística proposta supera as demais comparadas.

Tseng, Gupta e Sttafford (2005) desenvolveram uma heurística baseada em penalidades para o problema Fm/S_{sd} , $prmu/C_{max}$ e a comparou com sua heurística de indexação de economias anterior (DAS; GUPTA; KHUMAWALA, 1995).

No mesmo trabalho citado na seção anterior, Crowder (2006) desenvolveu duas novas heurísticas construtivas e dois limitantes inferiores para o problema *SDST Flowshop* flexível, com máquinas uniformes e limite de estocagem entre as máquinas, com objetivo de minimização de makespan. A experimentação computacional mostrou bons resultados para problemas pequenos, mas apresentou diminuição de desempenho com o aumento do número de estágios de produção.

Moccellin e Nagano (2007) apresentaram uma propriedade estrutural para o problema de programação da produção com tempos de *setup* separados dos tempos de processamento, sejam dependentes ou independentes da sequência de programação. Essa propriedade fornece um limitante superior para o tempo em que a máquina fica ociosa entre o fim de sua preparação (*setup*) e o início do processamento. Esta

propriedade pode ser utilizada para a solução do problema de *flowshop* com tempos de setup separados e objetivo de minimização de *makespan* através de uma analogia com o *ATSP*. Os resultados da experimentação computacional mostram que a analogia é adequada para problemas com 10 ou mais tarefas.

Dong, Huang e Chen (2009) estudaram a heurística NEHT-RB e propuseram três novas regras de prioridade para a primeira etapa da heurística, em substituição à regra original *LPT*. Todas as novas regras de prioridade são baseadas em relações entre a média e o desvio padrão dos tempos de processamento e uma média adaptada para os tempos de *setup*. A experimentação computacional mostrou que as heurísticas adaptadas com as novas regras de prioridade superam a heurística NEHT-RB quando a distribuição dos tempos de *setup* é no máximo igual à distribuição dos tempos de processamento. No mesmo trabalho eles propuseram uma estratégia de desempates para a segunda etapa da heurística NEHT-RB, porém sem conseguir melhores resultados.

Eren (2010), no mesmo trabalho revisado na seção anterior, apresentou três heurísticas para o problema *SDST flowshop* com m máquinas e objetivo de minimização de *makespan* e soma ponderada das datas de término. Todas as heurísticas são baseadas na heurística NEH. As soluções das heurísticas foram comparadas com as soluções ótimas para problemas pequenos (6 máquinas e 18 tarefas) e comparadas entre si para problemas maiores.

2.3.Métodos Meta-heurísticos

Parthasarathy e Rajendran (1997a) desenvolveram um algoritmo *simulated annealing* para minimizar atraso ponderado no problema *SDST Flowshop*. Os resultados computacionais mostraram que seu método supera a busca tabu em 70% dos problemas avaliados. Eles estenderam sua heurística para minimizar atraso ponderado máximo (PARTHASARATHY; RAJENDRAN, 1997b).

Já citado na seção anterior, Ríos-Mercado e Bard (1998b) apresentaram a meta-heurística GRASP (*Greedy Adaptive Search Procedure*) e a compararam com os métodos heurísticos construtivos NEHT-RB e SETUP.

Norman (1999) propôs vários métodos para um problema particular, o *SDST flowshop* com capacidade de armazenamento limitada entre máquinas ou $F / S_{sd}, prmu, block / C_{max}$. Neste caso, o autor avaliou métodos de busca tabu para o problema. A experimentação computacional demonstrou a efetividade da abordagem busca tabu perante as heurísticas utilizadas para comparação.

Ruiz, Maroto e Alcaraz (2005) apresentaram dois algoritmos genéticos avançados, assim como várias adaptações de meta-heurísticas existentes para o problema *SDST flowshop* com critério de minimização do *makespan*. Os parâmetros do algoritmo genético foram calibrados pela técnica de Planejamento de Experimentos (*DOE – Design of Experiments*). Nos algoritmos, parte da população inicial foi gerada através do método NEHT-RB, parte foi gerada aleatoriamente e o restante através de uma adaptação da inserção de tarefas do NEH em seqüências aleatórias. Uma hibridização do método é proposta através da adoção de uma busca local. Os métodos foram comparados, utilizando os problemas de Taillard, com outros métodos: outros algoritmos genéticos, *simulated annealing*, busca local iterativa e busca tabu além das heurísticas construtivas TOTAL, SETUP, NEHT-RB e obteve desempenho claramente superior aos demais.

Zandieh, Ghomi e Hussein (2006) estudaram o problema *SDST flowshop* híbrido com uma ou mais máquinas em cada estágio e objetivo de minimização do *makespan*. Eles propuseram um algoritmo imune (*IA*) que foi comparado com um algoritmo genético de chave aleatória (*RKGA*), e obteve maior porcentagem de sucesso.

Gajpal, Rajendran e Ziegler (2005) desenvolveram um algoritmo de otimização de colônia de formigas (*ACO – ant colony optimization*) para o problema *SDST Flowshop* com objetivo de minimização de *makespan*. Sua meta-heurística foi comparada durante a experimentação computacional a uma heurística *ACO* existente para o problema de *flowshop* permutacional, à heurística de indexação de economias (*SI*) e à meta-heurística GRASP, obtendo resultados superiores ao GRASP e SI, porém com maior tempo computacional que GRASP.

Ruiz e Stützle (2008) propuseram dois algoritmos Iterativos Gulosos (*IG – Iterated Greedy*), considerando dois objetivos de otimização diferentes: minimização do makespan e minimização do atraso (*tardiness*) ponderado. Seu algoritmo guloso é baseado em seu trabalho anterior para o *flowshop* permutacional (RUIZ; STÜTZLE, 2007) e começa como solução inicial com uma sequência obtida pelo método NEHT-RB e então, iterativamente, remove tarefas da solução atual e as re-insere, buscando melhorar a sequência. Além do critério de aceitação baseado na melhoria do critério de desempenho é também utilizado um critério utilizado em algoritmos *simulated annealing* com temperatura constante, a fim de evitar estagnação em ótimos locais. Um dos métodos inclui uma busca local a fim de melhorar seu desempenho. Os métodos foram comparados utilizando os problemas de Taillard com o algoritmo genético de Ruiz, Maroto e Alcaraz (2005), um algoritmo memético com e um sem busca local e um algoritmo de otimização de colônia de formigas (*ACO – ant colony optimization*). O algoritmo iterativo guloso (IG) foi superior aos demais para as três categorias de limite de tempo definido por $(n \cdot \frac{m}{2} \cdot t)$ milissegundos com $t = \{30, 60, 90\}$.

2.4.Considerações

Os métodos de solução exata possuem aplicação limitada a ambientes específicos, em especial $m = 2$, ou a números pequenos de tarefas. Este fato direciona o desenvolvimento das heurísticas construtivas e por sua vez às meta-heurísticas.

Dentro desse contexto, as heurísticas construtivas mais expressivas encontradas na literatura são as heurísticas: NEHT-RB, para distribuições de tempo de setup menores que a distribuição dos tempos de processamento; e as heurísticas TOTAL e SETUP para distribuições de tempos de setup iguais ou maiores à distribuição dos tempos de processamento. Cabe observar aqui que esta conclusão refere-se às análises sem quaisquer procedimentos de busca local, que descaracterizam o conceito de heurística construtiva (como ocorre com a heurística HYBRID, por exemplo).

Para o caso das meta-heurísticas, o método mais expressivo encontrado na literatura até o momento é o IG.

A seguir, no capítulo 3 serão desenvolvidos dois novos métodos: um método heurístico e outro meta-heurístico. Como parte do desenvolvimento destes métodos serão apresentados dois parâmetros de uma programação.

3. Proposição do método heurístico e do método meta-heurístico

Os métodos propostos neste trabalho partem da investigação de duas propriedades do problema *SDST flowshop*: a primeira, explorada por Ríos-Mercado e Bard (1999b), chamada fator residual da relação entre duas tarefas; e a segunda, a propriedade *UBX*, investigada por Moccellini e Nagano (2007).

A investigação dessas propriedades levou ao desenvolvimento de dois parâmetros: o parâmetro *XR*, que representa (i) a ociosidade de uma máquina entre sua preparação para a próxima tarefa (*setup*) e o início do processamento daquela tarefa, e (ii) o tempo de bloqueio de uma tarefa, ou seja, o tempo entre o fim do processamento de uma tarefa em uma máquina e o início de seu processamento na máquina seguinte; e o parâmetro *QR*, que representa o mesmo, mas para as tarefas sendo programadas na data mais tarde de início.

Tais parâmetros são uma função do fator residual, adaptado de Ríos-Mercado e Bard (1999b) e são utilizados diretamente no cálculo eficiente do *makespan*, conforme Taillard (1990) que já é utilizado no método NEHT-RB.

O uso de parâmetros pré-calculados (os fatores residuais) acelera o cálculo eficiente de *makespan*. Este fato motivou a proposta de um método de inserção de tarefas como o NEHT-RB, porém mais “guloso”, ou seja, testando mais tarefas em diferentes posições. Este conceito é utilizado tanto no método heurístico quanto no meta-heurístico.

3.1. Notação

A seguinte notação será utilizada pelo resto deste trabalho:

- Enunciado:

- Dado um conjunto de n tarefas $J = \{J_1, J_2, \dots, J_j, \dots, J_n\}$, em que cada tarefa deve ser processada em um conjunto de m máquinas $M = \{M_1, M_2, \dots, M_k, \dots, M_m\}$ em série (*flowshop* permutacional), sujeitas a tempos de preparação (*setup*) separados e dependentes da seqüência, encontrar a seqüência $\sigma = \{J_{[1]}, J_{[2]}, \dots, J_{[j]}, \dots, J_{[n]}\}$ que minimiza o critério de desempenho *Makespan* (M).
- Por simplicidade de notação, o conjunto de tarefas pode ser tratado por $J = \{1, 2, \dots, j, \dots, n\}$, e quando programados numa seqüência qualquer podem ser tratados por $\sigma = \{[1], [2], \dots, [j], \dots, [n]\}$. Da mesma forma, o conjunto de máquinas pode ser representado por $M = \{1, 2, \dots, i, \dots, m\}$.
- Índices e parâmetros:
 - m número de máquinas
 - n número de tarefas
 - i índice de máquinas, $i \in M = \{1, 2, \dots, m\}$
 - j, k índice de tarefas, $j, k \in J = \{1, 2, \dots, n\}$
- Dados de entrada:
 - p_{ij} tempo de processamento da tarefa j na máquina i , $i \in M, j \in J$
 - s_{jk}^i tempo de *setup* na máquina i quando a tarefa j é programada imediatamente antes da tarefa k . Considera-se que não há tempo de *setup* para primeira tarefa, ou seja $s_{0k}^i = 0$
- Parâmetros calculados:
 - X_{jk}^i tempo ocioso da máquina i entre as tarefas j e k
 - Y_{jk}^i tempo de espera para o início da tarefa k na máquina $i+1$ depois de finalizado seu processamento na máquina i , com a tarefa j precedendo diretamente a tarefa k

3.2. Os conceitos de Ociosidade de máquina e Bloqueio de tarefas

Em um problema *SDST Flowshop* existem duas possibilidades para a relação entre duas tarefas programadas consecutivamente, aqui denominadas J_u e J_v em duas

máquinas consecutivas, aqui chamadas i e $i+1$. Estas relações estão representadas nas figuras 3.1 e 3.2.

Figura 3.1 – Exemplo de ociosidade na máquina $i+1$, representado na figura por $X_{uv}^{i+1} > 0$

Figura 3.2 – Exemplo de bloqueio da tarefa J_v entre as máquinas i e $i+1$, representado na figura por $Y_{uv}^i > 0$

3.3. Investigação do fator residual e da propriedade UBX

A figura 3.1 mostra que se o *setup* entre as tarefas J_u e J_v na máquina $i+1$ ocorre antes que o processamento da tarefa J_v termine na máquina i , a máquina $i+1$ ficará ociosa, aguardando a chegada da tarefa J_v para processamento. Isto é representado por $X_{uv}^{i+1} > 0$. Já a figura 3.2 mostra que se ao terminar o processamento da tarefa J_v na máquina i , o *setup* da tarefa J_u para a tarefa J_v ainda não tiver ocorrido na máquina $i+1$, a tarefa J_v ficará bloqueada, aguardando a liberação da máquina $i+1$. Isto é representado por $Y_{uv}^i > 0$.

Ríos-Mercado e Bard (1999b) utilizam essa característica do problema *SDST Flowshop* ao descrever a data de término (t_{ij}) da tarefa j , na máquina i . Assumindo que a tarefa k suceda imediatamente a tarefa j , sua data de término na máquina i é dada por:

$$t_{ik} = \max\{t_{ij} + s_{jk}^i, t_{i-1,k}\} + p_{ik}$$

Com $t_{0k} = 0$ para todo k e $s_{0k}^i = 0$ para todo i e k .

Desta forma verifica-se que a relação entre $t_{ij} + s_{jk}^i$ e $t_{i-1,k}$ é crítica na verificação da existência de ociosidade na máquina i entre o fim do *setup* entre as tarefas j e k e o início do processamento da tarefa k ; ou o bloqueio da tarefa k após seu processamento na máquina i . Estas condições tomam a seguinte forma:

1. Se $t_{ij} + s_{jk}^i > t_{i-1,k}$, a tarefa k finaliza seu processamento na máquina $i-1$ antes que a tarefa j tenha terminado seu processamento na máquina i . A tarefa k fica, portanto, bloqueada até o fim do processamento da tarefa j na máquina i .
2. Se $t_{ij} + s_{jk}^i < t_{i-1,k}$, o *setup* entre as tarefas j e k na máquina i já ocorreu, porém a tarefa k ainda não finalizou seu processamento na máquina $i-1$, por isso a máquina i fica ociosa até o fim do processamento da tarefa k na máquina i .
3. Se $t_{ij} + s_{jk}^i = t_{i-1,k}$, a máquina i está pronta para iniciar o processamento da tarefa k no momento exato em que a tarefa k deixa a máquina $i-1$.

A partir destes conceitos, Ríos-Mercado e Bard (1999b) observaram que para o objetivo de minimização de *makespan*, o ideal seria que todas as relações entre tarefas fossem do terceiro tipo, ou seja, $t_{ij} + s_{jk}^i = t_{i-1,k}$. Isto ocorre apenas quando

$$p_{ij} + s_{jk}^i = s_{jk}^{i-1} + p_{i-1,k}$$

A fim de explorar tais relações eles definiram o fator residual como

$$r_{jk}^i = (p_{ij} + s_{jk}^i) - (s_{jk}^{i-1} + p_{i-1,k}) \quad i > 1; j, k \in J$$

Pode-se observar que $r_{jk}^i > 0$ remete à primeira condição; $r_{jk}^i < 0$ à segunda condição; e $r_{jk}^i = 0$ à terceira. Sua praticidade consiste no fato que tais relações podem ser calculadas para cada par de tarefas em cada uma das máquinas. Seu cálculo teve por objetivo desenvolver um algoritmo que encontrasse programações através das relações com menor fator residual. As figuras 3.3, 3.4 e 3.5 ilustram as possibilidades para o fator residual.

Figura 3.3 – Exemplo de relação entre duas tarefas j e k com $r_{jk}^i > 0$

Figura 3.4 – Exemplo de relação entre duas tarefas j e k com $r_{jk}^i < 0$

Figura 3.5 – Exemplo de relação entre duas tarefas j e k com $r_{jk}^i = 0$

Ríos-Mercado e Bard (1999b) citaram que a ociosidade de máquina possui efeito cumulativo nas próximas máquinas. O trabalho de Moccellini e Nagano (2007) também estudaram estas relações características do problema de *flowshop* permutacional com tempos de *setup*, explorando o efeito cumulativo da ociosidade de uma máquina nas demais máquinas. Eles desenvolveram um limitante superior UBX_{jk}^i para a ociosidade X_{jk}^i que é definido pela seguinte expressão, adaptada de Moccellini e Nagano (2007):

$$UBX_{jk}^i = \max\{0; UBX_{jk}^{i-1} + (s_{jk}^{i-1} + p_{i-1,k}) - (p_{ij} + s_{jk}^i)\}$$

Com $UBX_{jk}^1 = 0$ para quaisquer tarefas j e k e $s_{0k}^i = 0$ para todo i .

Assim como o fator residual, estas relações podem ser previamente calculadas para cada par de tarefas. Os autores sugeriram a solução do problema *SDST Flowshop* através de uma analogia com *ATSP*, com a matriz de distâncias dada por $UBX_{jk}^m + s_{jk}^m$ e realizaram uma experimentação computacional para comprovar a adequação da analogia. Essa experimentação mostrou que a analogia é válida para problemas com $n \geq 10$. Neste trabalho será analisada a proposta de solução através da analogia, aplicando o algoritmo VAM utilizado nas heurísticas TOTAL e SETUP. Este método será chamado UBX+S.

3.4. Desenvolvimento do parâmetro *XR*

A partir da seção anterior já se pode delinear uma comparação inicial entre o fator residual e *UBX*:

- Ambos baseiam-se nas características do problema *SDST Flowshop* que definem a existência de ociosidade de máquina ou o bloqueio de tarefa.
- Ambos podem ser calculados para cada par de tarefas em cada uma das máquinas.
- Apenas *UBX* considera o efeito cumulativo da ociosidade de uma máquina.
- Apenas o fator residual trabalha com valores positivos e negativos, indicando respectivamente bloqueio de tarefa e ociosidade de máquina.
- Nenhuma das duas incorpora os efeitos causados pela programação de tarefas anteriores.

O objetivo desta seção é estudar a relação entre duas tarefas programadas em sequência e desenvolver expressões para o efeito que estas relações possuem não apenas nas máquinas seguintes (como em *UBX*), mas também na programação das próximas

tarefas. Para isso, serão comparadas as expressões do fator residual e UBX , além de revisar as relações observadas nas seções anteriores.

Primeiramente, é importante avaliar as similaridades e diferenças nas expressões do fator residual e UBX . Para isso, elas serão colocadas lado a lado, e a expressão do fator residual re-arranjada para auxiliar a comparação:

$$r_{jk}^i = -(s_{jk}^{i-1} + p_{i-1,k}) + (p_{ij} + s_{jk}^i)$$

$$UBX_{jk}^i = \max\{0; UBX_{jk}^{i-1} + (s_{jk}^{i-1} + p_{i-1,k}) - (p_{ij} + s_{jk}^i)\}$$

É fácil notar com esse re-arranjo que as parcelas $(s_{jk}^{i-1} + p_{i-1,k})$ e $(p_{ij} + s_{jk}^i)$ são comuns, apesar de aparecerem com sinais invertidos. Dessa forma, a expressão para UBX poderia ser re-escrita como uma função do fator residual da seguinte maneira, por exemplo:

$$UBX_{jk}^i = \max\{0; UBX_{jk}^{i-1} - r_{jk}^i\}$$

Como notado antes, UBX está limitado a valores não negativos, e seus valores positivos estão associados à ociosidade de máquinas, enquanto o fator residual representa as ociosidades de máquina com seus valores negativos.

A análise das relações entre as tarefas utilizará neste momento um exemplo numérico. A figura 3.6 traz a programação das tarefas J_t , J_u e J_v , com os tempos de processamento e tempos de *setup* sobre as barras do gráfico de Gantt:

Figura 3.6 – Exemplo de uma programação de 3 tarefas em 2 máquinas

O gráfico mostra ainda o bloqueio da tarefa J_u depois de processada na máquina 1 ($Y^1_{tu} = 2$), e a ociosidade da máquina 2 entre o tempo de *setup* da tarefa J_u para J_v e o processamento da tarefa J_v ($X^2_{uv} = 5$).

Utilizando os valores do exemplo, os valores do fator residual e de *UBX* podem ser calculados para os pares de tarefas tu e uv :

$$r^2_{tu} = (p_{2t} + s^2_{tu}) - (s^1_{tu} + p_{1u}) = (7 + 2) - (4 + 3) = 2$$

$$r^2_{uv} = (p_{2u} + s^2_{uv}) - (s^1_{uv} + p_{1v}) = (6 + 2) - (9 + 6) = -7$$

Os cálculos mostram que o fator residual entre as tarefas J_t e J_u equivalem ao Y^1_{tu} , porém o valor do fator residual entre as tarefas J_u e J_v difere da ociosidade real da programação X^2_{uv} em 2 unidades. A análise da figura 3.6 mostra que esta diferença é devida ao bloqueio anterior (Y^1_{tu}).

Esta análise motiva a adaptação de uma expressão para a relação entre duas tarefas que considere: a praticidade da relação entre tarefas do fator residual; o efeito cumulativo da ociosidade de uma máquina nas próximas máquinas, como *UBX*; o efeito do bloqueio de uma tarefa nas tarefas programadas depois; e, por fim, condense os parâmetros X e Y em um só parâmetro. Tal parâmetro será denominado XR , para diferenciá-lo do parâmetro original X .

Para facilitar a utilização do novo parâmetro é realizada uma adaptação no fator residual que toma a seguinte forma:

$$r_{jk}^i = (s_{jk}^{i-1} + p_{i-1,k}) - (p_{ij} + s_{jk}^i)$$

Uma vez definido o novo padrão para o fator resíduo, pode-se, por exemplo re-escrever UBX com base no fator resíduo:

$$UBX_{jk}^i = \max \{0; UBX_{jk}^{i-1} + r_{jk}^i\}$$

De acordo com os desenvolvimentos e adaptações anteriores, o parâmetro XR pode ser expresso por:

$$XR_{[j][j+1]}^i = r_{[j][j+1]}^i + \max\{XR_{[j][j+1]}^{i-1}; 0\} + \min\{XR_{[j-1][j]}^i; 0\} \text{ com } i = 2, \dots, m \text{ e } j = 1, \dots, n-1.$$

Neste ponto é importante notar o uso dos colchetes para as tarefas, pois a relação trata de tarefas já programadas.

Os valores de XR para $i = 1$ são, como o fator residual e UBX , iguais a zero. Para a ociosidade entre o início da programação e o início da primeira tarefa nas máquinas depois da primeira, utiliza-se a seguinte expressão:

$$XR_{[0][1]}^i = XR_{[0][1]}^{i-1} + p_{i-1,[1]} \text{ com } i = 2, \dots, m \text{ e } XR_{[0][1]}^1 = 0$$

Os valores de XR ser, como o fator residual, positivos, negativos ou zero. Os valores positivos representam a ociosidade da máquina i entre as tarefas $[j]$ e $[j+1]$; os valores negativos representam o bloqueio da tarefa $[j+1]$ depois de processada na máquina $i-1$; e zero quando não houver nem bloqueio nem ociosidade.

A originalidade da abordagem de XR consiste na introdução da parcela $\min\{XR_{[j-1][j]}^i; 0\}$, que é a consideração do efeito dos bloqueios de tarefas anteriores nas tarefas programadas depois.

O parâmetro XR assemelha-se a UBX nos seguintes aspectos:

- Ambos podem ser descritos como função do fator residual;
- Ambas as expressões consideram o efeito cumulativo da ociosidade de uma máquina.

Por outro lado, XR difere de UBX nos seguintes aspectos:

- UBX estabelece um limite superior para o parâmetro X , enquanto XR é um parâmetro calculado de uma programação, em substituição aos parâmetros X e Y ;
- Para substituir os parâmetros X e Y os bloqueios são representados por valores negativos de XR , enquanto UBX admite apenas valores não negativos;
- O parâmetro XR considera o efeito cumulativo do bloqueio de uma tarefa;

A seguir é demonstrada uma utilização para o parâmetro XR .

3.5.O cálculo eficiente de Makespan e o parâmetro QR

A heurística construtiva NEH é a mais bem sucedida heurística construtiva para o *flowshop* permutacional. Sua adaptação para o problema *SDST flowshop*, NEHT-RB, também. Este sucesso baseia-se na boa qualidade de suas soluções em comparação com sua velocidade em termos computacionais.

Ambos os métodos consistem de duas fases: uma primeira fase de ordenação das tarefas segundo LPT e uma segunda fase em que a solução final é iterativamente construída através da inserção das tarefas, na ordem da primeira fase, na posição da subsequência que minimiza o *makespan* parcial

Porém, originalmente a complexidade do método NEH seria $O(mn^3)$ devido à necessidade de calcular $|S+I|$ vezes o *makespan* para cada tarefa inserida, onde $|S+I|$ é o tamanho atual da sequência parcial. Tailard (1990) propõe uma melhoria que acelera o cálculo do *makespan* quando uma tarefa é inserida em cada uma das $|S+I|$ posições da sequência S .

Sua melhoria consiste em calcular para a subsequência atual os valores:

- a) Data mais cedo de término da programação: (e_{ij}) , com $i = 1, \dots, m$ e $j = 1, \dots, k-1$
- $e_{i0} = 0, e_{0j} = 0$
 - $e_{ij} = \max\{e_{i-1,j}, e_{i,j-1} + s_{j-1,j}^i\} + p_{ij}$
- b) Intervalo entre o início da tarefa j na máquina i e o término da programação (q_{ij}) , com $i = m, m-1, \dots, 1$ e $j = k-1, k-2, \dots, 1$
- $q_{ik} = 0, q_{m+1,j} = 0$
 - $q_{ij} = \max\{q_{i+1,j}, q_{i,j+1} + s_{j,j+1}^i\} + p_{ij}$
- c) Data de término para cada uma das posições a serem inseridas (f_{ij}) com $i = 1, \dots, m$ e $j = 1, \dots, k$
- $f_{i0} = 0, f_{0j} = 0$
 - $f_{ij} = \max\{f_{i-1,j}, e_{i,j-1} + s_{j-1,h}^i\} + p_{ih}$

O valor do *makespan* da sequência parcial $\psi(j)$ ao adicionar a tarefa k à j -ésima posição, com $j = 1, \dots, k$ é dado por:

$$\psi(j) = \max_{i=1,\dots,m} \{f_{ij} + s_{hj}^i + q_{ij}\}$$

Onde $s_{hj}^i = q_{ij} = 0$ para $j = k$.

O parâmetro XR pode ser aplicado diretamente aqui, através da adaptação do cálculo da data mais cedo de término:

$$e_{ij} = e_{i,j-1} + s_{ij}^i + \max\{XR_{j,j+1}^i; 0\} + p_{ij}$$

Potencializando uma aceleração no seu cálculo devido ao uso dos fatores residuais.

Esta utilização sugere a adaptação também para q_{ij} . Para isso é necessário adaptar o parâmetro XR para as tarefas se iniciando na em sua data mais tarde de início. O parâmetro adaptado será denominado QR :

$$QR_{[j][j+1]}^i = -r_{[j][j+1]}^{i+1} + \max\{QR_{[j][j+1]}^{i+1}; 0\} + \min\{QR_{[j+1][j+2]}^i; 0\}$$

E q_{ij} seria simplificado para:

$$q_{ij} = \max\{QR_{[j][j+1]}^i; 0\} + s_{ij}^i + p_{ij} + q_{i,j+1}$$

3.6.A nova heurística CNIT

A simples substituição das novas expressões para e_{ij} e q_{ij} na heurística NEHT-RB não traria benefício algum, pois por se tratar de uma heurística construtiva, o resultado alcançado seria o mesmo. Em se tratando de tempo computacional, apesar de uma redução no tempo de inserção ser esperado, seria esperado um aumento do tempo total em função do cálculo dos valores de r_{jk}^i para cada par de tarefas.

Assim, a proposta do novo método denominado CNIT (Carneiro-Nagano Inserção Total) é eliminar a primeira fase do método NEHT-RB e tornar a segunda fase mais “gulosa”: ao invés de inserir as tarefas em uma seqüência previamente estabelecida (no caso das heurísticas NEH e NEHT-RB a seqüência *LPT*), o método CNIT testa em todas as iterações todas as tarefas não programadas em todas as posições. A subsequência com menor *makespan* é mantida para a próxima iteração e a tarefa inserida é retirada do conjunto de tarefas não programadas. O pseudo-código da nova heurística é dado a seguir:

Entrada:

- Conjunto $J = \{J_1, J_2, \dots, J_n\}$ de n tarefas com tempos de setup assimétricos e dependentes da seqüência;

Saída:

- Seqüência de tarefas programadas $\sigma = \{J[1], J[2], \dots, J[n]\}$;
- *Makespan* da programação solução;

Definições:

- Seja U o conjunto de tarefas não programadas

Passo 1: Inicialização:

Passo 1.1: Calcular o fator residual r_{jk}^i para cada par de tarefas em cada uma das máquinas

Passo 1.2: $U \leftarrow J$;

Passo 1.3: $\sigma \leftarrow \emptyset$

Passo 2: Enquanto $U \neq \emptyset$ faça:

Passo 2.0: Inicialização. Faça:

Passo 2.0.1: *Melhor Makespan atual* = 0;

Passo 2.0.2: *Tarefa inserida* $\leftarrow \emptyset$;

Passo 2.0.3: *Posição da tarefa inserida* $\leftarrow \emptyset$;

Passo 2.1: Para cada uma das tarefas J_j em U faça:

Passo 2.1.1: Calcule o valor do *Makespan atual* da subsequência atual através do cálculo eficiente de *makespan* conforme proposto neste trabalho, depois de inserir a tarefa J_j em cada uma das posições possíveis.

Passo 2.1.1.1: Calcule os valores de e_{ij}

Passo 2.1.1.2: Calcule os valores de q_{ij}

Passo 2.1.1.3: Calcule os valores de f_{ij}

Passo 2.1.1.3: Calcule os valores de $\psi(j)$

Passo 2.1.1.4: Determine a melhor posição k de inserção da tarefa J_j : $k = \operatorname{argmin}_{j=1, \dots, |S+1|} \{\psi(j)\}$

Passo 2.1.2: Se *Makespan atual* < *Melhor Makespan atual*:

Passo 2.1.2.1: *Melhor Makespan atual = Makespan atual;*

Passo 2.1.2.2: *Tarefa inserida = J_j ;*

Passo 2.1.2.3: *Posição da tarefa inserida = k*

Passo 2.2: Faça:

Passo 2.2.1: $\sigma \leftarrow$ *Tarefa inserida na Posição da tarefa inserida*

Passo 2.2.2: $U \leftarrow U -$ *Tarefa Inserida*

Passo 2.2.3: *Makespan = Melhor Makespan atual*

Passo 3: Saídas:

Passo 3.1: σ

Passo 3.2: *Makespan*

3.7.Exemplo numérico

Seja o exemplo numérico da tabela 3.1:

	1	2	3	4
p_{ij}				
1	6	3	2	1
2	2	2	4	2
s_{jk}^1				
1	-	5	3	2
2	5	-	3	1
3	2	1	-	5
4	3	2	5	-
s_{jk}^2				
1	-	1	3	5
2	4	-	3	1
3	3	4	-	1
4	7	8	4	-

Tabela 3.1 – Exemplo numérico de um problema $F2/S_{DS}, prmu/C_{max}$

Passo 1: Os fatores residuais podem ser calculados através da expressão $r_{jk}^i = (s_{jk}^{i-1} + p_{i-1,k}) - (p_{ij} + s_{jk}^i)$ e são mostrados na tabela a seguir:

	1	2	3	4
r_{jk}^1				
1	-	0	0	0
2		-	0	0
3	0	0	-	0
4	0	0	0	-
r_{jk}^2				
1	-	5	0	-4
2	5	-	0	-1
3	1	-4	-	1
4	0	-5	1	-

Tabela 3.2 – Fatores residuais para o problema da tabela 3.1

Neste todas as todas as tarefas ainda não foram programadas, assim $U = \{1, 2, 3, 4\}$.

Passo 2:

Iteração 1: Todas as tarefas são testadas para serem inseridas. Os valores de Makespan atual são calculados conforme o cálculo eficiente de *makespan* alterado neste trabalho.

J_j	k	$\psi(j)$
1	1	8
2	1	5
3	1	6
4	1	3

Tabela 3.3 – Valores de *Makespan atual* da primeira tarefa a ser inserida do problema da tabela 3.1

A seqüência com menor valor de *Makespan atual* é selecionada. Neste ponto, então é inserida a tarefa 4 na posição $k = 1$ da subsequência. O conjunto de tarefas não programadas fica $U = \{1, 2, 3\}$

Iteração 2: São testadas as tarefas em U em todas as posições disponíveis. Os cálculos de *Makespan atual* são dados pela tabela a seguir:

J_j	k	$\psi(j)$
1	1	15
	2	12
2	1	8
	2	13
3	1	10
	2	12

Tabela 3.4 – Valores de *Makespan atual* da segunda tarefa a ser inserida do problema da tabela 3.1

O menor valor de *Makespan* é obtido com a inserção da tarefa 2 na primeira posição, ficando $\sigma = \{2, 4\}$, e $U = \{1, 3\}$.

Iteração 3: A tabela 3.5 mostra os valores de *makespan* para as tarefas em U inseridas em todas as posições:

J_j	k	$\psi(j)$
1	1	19
	2	23
	3	17
3	1	15
	2	16
	3	16

Tabela 3.5 – Valores de *Makespan atual* da terceira tarefa a ser inserida do problema da tabela 3.1

O menor valor de makespan é obtido com a inserção da tarefa 3 na posição 1. $\sigma = \{3, 2, 4\}$, e $U = \{1\}$.

Iteração 4: A tabela 3.6 mostra os valores de makespan para as tarefas em U inseridas em todas as posições:

J_j	k	$\psi(j)$
1	1	24
	2	23
	3	26
	4	24

Tabela 3.6 – Valores de *Makespan atual* da quarta tarefa a ser inserida do problema da tabela 3.1

E a solução é $\sigma = \{3, 1, 2, 4\}$, com valor de Makespan igual a 23. A solução ótima deste problema é $\{2, 4, 3, 1\}$ com valor de Makespan igual a 22.

3.8.A nova meta-heurística CNIT-IG

Até o momento, o método IG de Ruiz e Stützle (2008) é o método mais expressivo encontrado na literatura. Seu algoritmo é composto de duas partes: na primeira, uma solução inicial é encontrada através do método NEHT-RB de Rios-Mercado e Bard (1999a) seguida de uma busca local descendente; na segunda parte a solução inicial é submetida a uma fase de destruição e construção seguida de uma busca local até que um critério de parada seja alcançado (em geral associado ao tempo de computação ou ao número de tentativas). Na fase de destruição um número d de tarefas é retirado aleatoriamente da solução inicial e reinserido em uma fase de construção, utilizando a regra de inserção do NEHT-RB. Depois das fases de destruição e construção e da busca local descendente, um critério de aceitação baseado em algoritmos de *simulated annealing* mas com valores de temperatura constante é utilizado para decidir se a solução tentativa pode ser utilizada na próxima iteração. Este critério de aceitação busca evitar a estagnação em ótimos locais. Em seu trabalho anterior, para o *flowshop* regular no qual a adaptação para o *SDST flowshop* foi baseado, Ruiz e Stützle (2007) encontram através de análises estatísticas *DOE* (*Design of Experiment* – Projeto de Experimentos) os dois parâmetros: d , que estabelece a quantidade de tarefas nas fases de destruição e construção e T , que estabelece o critério de aceitação de temperatura constante.

Este trabalho propõe a adaptação de um novo método meta-heurístico baseado na heurística IG com as melhorias propostas pelo método CNIT, o qual vai ser chamado daqui em diante de CNIT-IG.

O novo método CNIT-IG, assim como o IG, possui duas partes:

- i. A primeira parte encontra a solução inicial através de uma heurística construtiva seguida por uma fase de busca local descendente. Para fins de avaliação pode ser utilizada a nova heurística ou qualquer uma das heurísticas construtivas avaliadas.
- ii. A segunda parte é constituída pelas fases de destruição e construção seguida de uma busca local. A diferença em relação ao método IG original está na

fase de construção em que, ao invés de utilizar a inserção na ordem da remoção, utiliza a regra de inserção de CNIT, em que todas as tarefas não programadas são testadas em todas as posições e a tarefa que produzir a subsequência de menor *makespan* é retirada do conjunto de tarefas retiradas da sequência inicial e a subsequência de menor *makespan* é mantida para a próxima operação. As demais etapas, como a busca local e a sujeição ao critério de aceitação de temperatura constante são mantidos.

Detalhamento do método CNIT-IG:

- a) Solução Inicial
 - a. CNIT
 - b. Melhor método na análise da experimentação computacional
- b) Busca Local Descendente

Segue abaixo o pseudo código do algoritmo de busca local descendente utilizado:

Entrada: Solução inicial σ

Saída: Solução melhorada em sua vizinhança

Passo 1: Inicialização:

Passo 1.1: Solução atual = Solução inicial

Passo 1.2: *Melhoria* = verdadeiro;

Passo2: Enquanto *Melhoria* = verdadeiro

Passo 2.1: *Melhoria* = falso

Passo 2.1: De $j = 1$ até n faça

Passo 2.1.2: Remova uma tarefa k aleatoriamente da sequência inicial

Passo 2.1.2: Testar a tarefa k inserindo em todas as posições possíveis e calculando o valor de *makespan* utilizando o cálculo eficiente de *makespan*. A sequência de menor *makespan* é a Sequência promissora.

Passo 2.1.3: Se o Makespan da Seqüência promissora for menor que o makespan da Solução atual faça:

Passo 2.1.3.1: Solução atual = Solução Promissora

Passo 2.1.3.2: Melhoria = falso

Passo 3: Saída: Solução melhorada em sua vizinhança.

c) Fases de Destruição e Construção:

A fase de Destruição consiste em remover d tarefas aleatoriamente da programação atual, sendo d um parâmetro que deve ser estabelecido. Na prática, esta fase consiste em remover tarefas aleatoriamente da solução atual e inseri-las em um conjunto U , de tarefas não programadas.

A fase de Construção utiliza o conceito apresentado para a heurística CNIT para inserção das tarefas do conjunto de tarefas ainda não programadas (U).

Além do critério de aceitação óbvio, que aceita soluções com menor valor de makespan, um critério de aceitação de seqüências promissoras semelhante ao utilizado em algoritmos *simulated annealing* com temperatura constante é utilizado:

$$Temperatura = T \frac{\sum_{i=1}^m \sum_{j=1}^n p_{ij}}{n \cdot m \cdot 10}$$

E a seqüência proposta é aceita como uma seqüência promissora se a função objetivo, no caso o *makespan*, for melhor e também se

$$aleatório \leq \exp \left(\frac{Makespan\ da\ sol.\ promissora - Makespan\ atual}{Temperatura} \right)$$

Onde aleatório é um número aleatoriamente gerado com distribuição uniforme em $[0,1]$. O parâmetro T também deve ser estabelecido.

A cada etapa de Destruição e Construção é realizado uma busca local descendente.

4. Experimentação computacional, resultados e análise

A experimentação computacional neste trabalho pode ser dividida em três etapas:

- i. A avaliação do desempenho das novas heurísticas construtivas UBX+S e CNIT em relação às demais heurísticas construtivas (TOTAL, SETUP e NEHT-RB);
- ii. A avaliação do efeito da busca local descendente como apresentada em Ruiz e Stützle (2008) nas heurísticas construtivas avaliadas na primeira etapa.
- iii. A avaliação do efeito da solução inicial para a meta-heurística IG e sua comparação com sua adaptação CNIT-IG.

4.1.Delineamento do experimento

Todos os métodos foram implementados em linguagem de programação C e estão disponíveis no apêndice ao final do texto. Os programas foram rodados em um computador com processador Intel® Core™ 2 Duo 2.00 GHz e 2Gb de memória RAM.

Os parâmetros de entrada do programa seguem o seguinte padrão:

- Escolha da heurística construtiva para solução inicial (TOTAL, SETUP, UBX+S, NEHT-RB ou CNIT);
- Escolha das opções por busca local descendente, IG ou CNIT-IG
 - Escolha dos parâmetros d (número de tarefas na fase destruição) e T (parâmetro de temperatura constante) caso seja escolhido IG ou CNIT-IG;
 - Escolha do parâmetro t , que define o limite para o tempo de processamento, utilizado na função $n \times m/2 \times t / 1000$ segundos;

- Faixa de problemas do banco de dados. Cada problema tem nome e formato padronizados:
 - Nome do problema: SDST[nº do grupo]_ta[nº do problema].txt.
 - Conteúdo do problema:
 - Número de tarefas n ;
 - Número de máquinas m ;
 - Tempos de processamento em uma matriz $n \times m$;
 - Tempos de setup em m matrizes $n \times n$;
- Quantidade de vezes que deseja repetir o experimento;
- Opção para o armazenamento da sequência solução encontrada;

Os parâmetros de saída são fornecidos em um arquivo chamado “saída.xls”. Cada linha do arquivo segue o seguinte formato:

- Nome do problema;
- Valor do *makespan* obtido com a sequência encontrada;
- Tempo de computação com precisão de milésimos de segundo;
- Sequência encontrada (opcional);

4.1.1. Banco de dados

Para a experimentação computacional foi utilizado um banco de dados adaptado de Taillard para o *SDST flowshop* obtido em <http://soa.iti.es/problem-instances>.

O banco de dados original de Taillard foi gerado para o problema de *flowshop* regular com tempos de processamento aleatoriamente distribuídos no intervalo $[1, 99]$. Ele é composto por 120 problemas, separados em 12 classes, definidas pela combinação entre o número de máquinas e tarefas de cada classe, perfazendo um total de 10 problemas por classe. São 9 classes que combinam o número de tarefas $n = \{20, 50, 100\}$ com número de máquinas $m = \{5, 10, 20\}$; 2 classes que combinam $n = \{200\}$ com $m = \{10, 20\}$ e uma classe com $n = \{500\}$ e $m = \{20\}$.

O banco adaptado para o *SDST flowshop* utiliza os tempos de processamento do banco de dados original em 4 grupos com distribuições distintas dos tempo de *setup*:

- SSD_10: a distribuição dos tempos de setup está limitada a 10% do limite para o intervalo dos tempos de processamento. Ou seja, a os tempos de setup foram gerados conforme uma distribuição uniforme no intervalo [1, 9].
- SSD_50: a distribuição dos tempos de setup está limitada a 50% do limite para o intervalo dos tempos de processamento. Ou seja, os tempos de setup foram gerados conforme uma distribuição uniforme no intervalo [1, 49].
- SSD_100: a distribuição dos tempos de setup está limitada a 100% do limite para o intervalo dos tempos de processamento. Ou seja, a os tempos de setup foram gerados conforme uma distribuição uniforme no intervalo [1, 99].
- SSD_125: a distribuição dos tempos de setup está limitada a 125% do limite para o intervalo dos tempos de processamento. Ou seja, a os tempos de setup foram gerados conforme uma distribuição uniforme no intervalo [1, 124].

Portanto, o banco de dados Taillard adaptado para o problema *SDST flowshop* é constituído por quatro grupos de problemas, cada um com 120 problemas, totalizando 480.

4.1.2. Processo de análise dos resultados

As estatísticas utilizadas para comparação dos resultados da experimentação computacional são:

- Porcentagem de sucesso;
- Desvio relativo médio;
- Tempo médio de computação;

4.1.2.1. Porcentagem de sucesso

Segundo Nagano (1999), a porcentagem de sucesso é o quociente entre o número de problemas para os quais o método obteve a melhor solução (*makespan*) e o número total de problemas resolvido. Assim, a porcentagem de sucesso de um método h (S_h) é dada por:

$$S_h(\%) = \frac{q_h}{T}$$

Onde:

q_h : quantidade de vezes que o método h forneceu a melhor solução, isoladamente ou não

T : total de problemas resolvidos

Obviamente, quando os métodos obtêm a melhor solução para um mesmo problema, suas Porcentagens de sucesso são simultaneamente melhoradas.

4.1.2.2.Desvio relativo médio

O Desvio Relativo (DR_h) quantifica o desvio que o método h obtém em relação ao melhor *makespan* obtido para o mesmo problema, sendo calculado conforme segue:

$$DR_h(\%) = \frac{D_h - D^*}{D^*} \times 100$$

Onde:

D_h : duração total da programação (*makespan*) obtida pelo método h ;

D^* : *makespan* de referência para um determinado problema. Em geral o melhor obtido pelos métodos testados ou uma referência externa.

Juntamente com a base de dados de problemas de Tailard modificados para o problema SDST flowshop, na site <http://soa.iti.es/problem-instances> pode ser obtido um conjunto de valores de referência para os exemplos. Tais referências são os melhores valores de *makespan* disponíveis para a base de dados em questão. Quando esses valores forem utilizados como base para comparação do Desvio Relativo Médio, estes valores serão referenciados por BM (*Benchmark*).

4.1.2.3.Tempo médio de computação

O tempo médio de computação de um método é calculado por classe de problemas. Ele é a soma dos tempos de uma classe de problemas dividida pela quantidade de problemas da classe, ou seja, a média aritmética. O tempo médio de computação é fornecido em segundos. Ele pode ser expresso por:

$$TMC_{mn}^h = \frac{\sum_{p=1}^P TC_p^h}{P}$$

Onde:

TC_p^h : tempo de computação de um método h para o problema p;

P: número de problemas resolvidos com o método na classe m x n;

4.2.Análise dos resultados

Para a tabulação dos dados, análise e apresentação, utilizou-se o Microsoft® Office Excel® 2007.

Esta seção está dividida em 3 partes: a primeira compara os métodos heurísticos construtivos; a segunda avalia os resultados da aplicação de uma simples busca local

descendente; a terceira compara os resultados para as diferentes soluções iniciais para o método IG e o método CNIT.

4.2.1. Avaliação das heurísticas construtivas

As próximas tabelas mostram os resultados da porcentagem de sucesso para cada grupo do banco de dados Taillard modificado para o problema SDST flowshop (SSD_10, SSD_50, SSD_100 e SSD_125) apresentados em função do número do problema e da sua classe (n x m).

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	0%	0%	0%	70%	30%
20	10	0%	0%	0%	90%	10%
20	20	0%	0%	0%	90%	10%
50	5	0%	0%	0%	80%	20%
50	10	0%	0%	0%	90%	10%
50	20	0%	0%	0%	80%	20%
100	5	0%	0%	0%	70%	30%
100	10	0%	0%	0%	90%	10%
100	20	0%	0%	0%	70%	30%
200	10	0%	0%	0%	70%	30%
200	20	0%	0%	0%	40%	60%
500	20	0%	0%	0%	20%	80%
Médias		0%	0%	0%	72%	28%

Tabela 5.1 – Porcentagem de sucesso das heurísticas construtivas para o grupo SSD_10

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	0%	0%	0%	70%	30%
20	10	0%	0%	0%	50%	50%
20	20	10%	0%	0%	80%	10%
50	5	20%	0%	0%	30%	50%
50	10	0%	0%	0%	70%	30%
50	20	0%	0%	0%	60%	40%
100	5	50%	10%	0%	0%	40%
100	10	0%	0%	0%	20%	80%
100	20	0%	0%	0%	40%	60%
200	10	0%	0%	0%	0%	100%
200	20	0%	0%	0%	10%	90%
500	20	0%	0%	0%	0%	100%
Médias		7%	1%	0%	36%	57%

Tabela 5.2 – Porcentagem de sucesso das heurísticas construtivas para o grupo SSD_50

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	10%	20%	0%	10%	60%
20	10	0%	0%	0%	40%	60%
20	20	0%	0%	0%	70%	30%
50	5	60%	30%	0%	0%	10%
50	10	0%	0%	0%	0%	100%
50	20	0%	0%	0%	70%	30%
100	5	90%	10%	0%	0%	0%
100	10	20%	0%	0%	0%	80%
100	20	0%	0%	0%	10%	90%
200	10	50%	40%	0%	0%	10%
200	20	0%	0%	0%	0%	100%
500	20	0%	0%	0%	0%	100%
Médias		19%	8%	0%	17%	56%

Tabela 5.3 – Porcentagem de sucesso das heurísticas construtivas para o grupo SSD_100

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	20%	40%	0%	20%	20%
20	10	0%	0%	0%	60%	40%
20	20	0%	0%	0%	50%	50%
50	5	60%	30%	0%	0%	10%
50	10	10%	0%	0%	30%	60%
50	20	0%	0%	0%	10%	90%
100	5	90%	10%	0%	0%	0%
100	10	10%	0%	0%	0%	90%
100	20	0%	0%	0%	0%	100%
200	10	60%	10%	0%	0%	30%
200	20	0%	0%	0%	0%	100%
500	20	40%	0%	0%	0%	60%
Médias		24%	8%	0%	14%	54%

Tabela 5.4 – Porcentagem de sucesso das heurísticas construtivas para o grupo SSD_125

Figura 5.1: Porcentagem de sucesso das heurísticas construtivas

A análise das tabelas e da figura permite as seguintes observações:

- Assim como observa do por Rios-Mercado e Bard (1999b), o desempenho da heurística SETUP cresce com o limite da distribuição do tempo de *setup* do grupo (sendo, portanto, melhor para o grupo SSD_125 e pior para o grupo SSD_10) e é melhor para os exemplos com menor número de máquinas, devido à analogia com o SDST – estes resultados são acompanhados, de perto pelos resultados alcançados pela heurística TOTAL.
- A heurística NEHT-RB domina para os problemas com menor número de tarefas ($n \leq 50$), em especial para o grupo SSD_10 em que domina até para $n = 200$ e $m = 10$.
- De modo geral, observando-se as médias dos grupos de exemplos SSD_50, SSD_100 e SSD_125, é possível concluir que a nova heurística CNIT é superior às demais, obtendo uma porcentagem média de sucesso superior às demais em especial nos problemas com maior número de tarefas. Isto pode ser observado inclusive no grupo SSD_10 para os exemplos com $n = \{200, 500\}$ e $m = 20$.

A seguir são apresentados os desvios relativos médios em relação ao *benchmark* (BM). Este procedimento permite comparar os resultados obtidos em cada fase da análise de maneira absoluta, possibilitando, inclusive a comparação entre os métodos heurísticos e melhorativos.

Em conjunto nas tabelas estão os tempos médios de processamento (em segundos) para os exemplos.

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	13,61%	14,30%	11,87%	3,82%	4,31%
20	10	16,83%	18,07%	15,14%	4,56%	6,05%
20	20	13,82%	15,08%	13,21%	3,90%	5,65%
50	5	10,57%	12,28%	10,23%	3,19%	4,26%
50	10	18,80%	18,13%	16,50%	5,63%	6,54%
50	20	18,58%	19,75%	17,52%	5,18%	6,64%
100	5	9,87%	9,51%	8,15%	3,16%	3,50%
100	10	14,42%	15,46%	13,76%	3,61%	4,11%
100	20	17,21%	17,49%	18,28%	4,58%	4,94%
200	10	10,84%	11,84%	11,75%	2,88%	3,12%
200	20	15,14%	15,47%	15,19%	3,04%	2,92%
500	20	10,61%	10,50%	12,23%	1,81%	1,51%
Médias		14,19%	14,82%	13,65%	3,78%	4,46%

Tabela 5.5 – Desvio relativo médio das heurísticas construtivas em relação ao *BM* para o grupo SSD_10

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	0,0003	0,0007	0,0005	0,0000	0,0013
20	10	0,0004	0,0004	0,0006	0,0003	0,0010
20	20	0,0004	0,0007	0,0008	0,0003	0,0016
50	5	0,0024	0,0024	0,0029	0,0006	0,0084
50	10	0,0031	0,0030	0,0037	0,0013	0,0140
50	20	0,0041	0,0042	0,0054	0,0028	0,0260
100	5	0,0163	0,0157	0,0168	0,0040	0,0859
100	10	0,0191	0,0176	0,0202	0,0063	0,1275
100	20	0,0237	0,0233	0,0281	0,0127	0,2629
200	10	0,1255	0,1255	0,1312	0,0346	1,5842
200	20	0,1450	0,1476	0,1637	0,0716	2,7938
500	20	2,1706	2,1480	2,2004	0,7687	75,4070

Tabela 5.6 – Tempo médio de computação das heurísticas construtivas para o grupo SSD_10

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	12,39%	12,82%	14,19%	7,01%	7,78%
20	10	12,68%	16,13%	15,00%	6,63%	5,81%
20	20	11,64%	11,90%	13,18%	4,57%	6,40%
50	5	10,34%	12,54%	18,28%	8,80%	8,46%
50	10	13,46%	15,39%	19,97%	7,21%	7,60%
50	20	14,69%	16,00%	16,96%	6,05%	6,94%
100	5	6,44%	8,13%	14,67%	8,35%	6,64%
100	10	10,79%	12,13%	17,36%	6,65%	5,91%
100	20	12,70%	13,25%	17,96%	5,37%	5,37%
200	10	6,57%	7,40%	15,20%	5,29%	3,82%
200	20	9,14%	9,43%	15,63%	4,09%	3,00%
500	20	3,70%	4,66%	12,73%	2,58%	0,87%
Médias		10,38%	11,65%	15,93%	6,05%	5,72%

Tabela 5.7 – Desvio relativo médio das heurísticas construtivas em relação ao *BM* para o grupo SSD_50

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	0,0005	0,0003	0,0002	0,0003	0,0005
20	10	0,0005	0,0003	0,0005	0,0001	0,0010
20	20	0,0007	0,0005	0,0008	0,0005	0,0019
50	5	0,0027	0,0027	0,0028	0,0007	0,0090
50	10	0,0031	0,0031	0,0034	0,0017	0,0145
50	20	0,0039	0,0042	0,0050	0,0026	0,0280
100	5	0,0161	0,0161	0,0164	0,0041	0,0833
100	10	0,0185	0,0186	0,0205	0,0068	0,1255
100	20	0,0235	0,0233	0,0276	0,0120	0,2147
200	10	0,1258	0,1260	0,1503	0,0336	1,4126
200	20	0,1434	0,1469	0,1672	0,0674	2,4692
500	20	2,1439	2,1555	2,2477	0,7568	73,7797

Tabela 5.8 – Tempo médio de computação das heurísticas construtivas para o grupo SSD_50

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	10,83%	9,75%	16,36%	11,21%	7,94%
20	10	12,22%	13,38%	15,87%	7,33%	7,80%
20	20	11,03%	9,97%	13,07%	5,28%	6,50%
50	5	9,84%	11,59%	20,08%	13,24%	10,43%
50	10	12,23%	13,98%	20,48%	10,24%	8,66%
50	20	12,82%	13,80%	18,12%	7,29%	7,59%
100	5	4,93%	6,49%	15,77%	11,53%	8,15%
100	10	8,64%	9,28%	18,38%	8,71%	6,34%
100	20	9,73%	10,82%	18,32%	6,76%	5,02%
200	10	2,98%	3,98%	15,47%	7,35%	3,52%
200	20	5,59%	7,17%	16,30%	4,97%	2,62%
500	20	0,73%	1,71%	12,67%	3,32%	0,07%
Médias		8,46%	9,33%	16,74%	8,10%	6,22%

Tabela 5.9 – Desvio relativo médio das heurísticas construtivas em relação ao *BM* para o grupo SSD_100

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	0,0001	0,0001	0,0002	0,0002	0,0006
20	10	0,0004	0,0004	0,0003	0,0003	0,0010
20	20	0,0004	0,0004	0,0009	0,0006	0,0019
50	5	0,0027	0,0026	0,0025	0,0009	0,0089
50	10	0,0030	0,0032	0,0033	0,0014	0,0150
50	20	0,0040	0,0042	0,0053	0,0030	0,0274
100	5	0,0173	0,0161	0,0174	0,0043	0,0834
100	10	0,0184	0,0189	0,0208	0,0067	0,1245
100	20	0,0232	0,0242	0,0291	0,0123	0,2200
200	10	0,1278	0,1295	0,1340	0,0346	1,4199
200	20	0,1483	0,1543	0,1801	0,0773	2,5356
500	20	2,1963	2,1403	2,2680	0,7320	71,2157

Tabela 5.10 – Tempo médio de computação das heurísticas construtivas para o grupo SSD_100

MÉTODOS						
N	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	10,22%	9,48%	15,54%	10,93%	8,60%
20	10	12,45%	12,85%	16,03%	7,85%	7,94%
20	20	10,98%	11,25%	13,68%	5,30%	5,87%
50	5	8,74%	10,03%	19,53%	14,13%	10,31%
50	10	11,52%	12,31%	20,04%	10,44%	9,28%
50	20	11,61%	13,19%	19,52%	8,92%	6,91%
100	5	4,28%	6,85%	14,96%	12,80%	8,89%
100	10	8,57%	9,63%	18,66%	9,97%	6,76%
100	20	9,12%	9,87%	18,60%	6,78%	5,00%
200	10	2,18%	4,27%	15,21%	8,22%	3,81%
200	20	5,35%	5,82%	16,23%	5,42%	2,48%
500	20	-0,04%	0,70%	12,49%	3,81%	-0,30%
Médias		7,92%	8,86%	16,71%	8,71%	6,30%

Tabela 5.11 – Desvio relativo médio das heurísticas construtivas em relação ao *BM* para o grupo SSD_125

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	0,0002	0,0003	0,0005	0,0002	0,0006
20	10	0,0002	0,0003	0,0005	0,0004	0,0010
20	20	0,0004	0,0008	0,0009	0,0003	0,0018
50	5	0,0024	0,0030	0,0024	0,0007	0,0082
50	10	0,0030	0,0034	0,0035	0,0015	0,0146
50	20	0,0043	0,0044	0,0056	0,0029	0,0267
100	5	0,0173	0,0158	0,0170	0,0046	0,0882
100	10	0,0180	0,0186	0,0208	0,0066	0,1236
100	20	0,0242	0,0237	0,0275	0,0127	0,2177
200	10	0,1287	0,1408	0,1305	0,0338	1,4128
200	20	0,1515	0,1518	0,1639	0,0703	2,4738
500	20	2,1623	2,1837	2,2131	0,7520	70,4757

Tabela 5.12 – Tempo médio de computação das heurísticas construtivas para o grupo SSD_125

Figura 5.2: Desvio relativo médio das heurísticas em relação ao *BM* (médias e por classe de problemas)

Figura 5.3: Tempos médios de computação das heurísticas construtivas

A análise que se segue às tabelas e às figuras destaca os seguintes aspectos:

- A heurística UBX+S só não possui os piores resultados para o grupo SSD_10, sendo superior aos métodos TOTAL e SETUP.
- Assim como observado na análise das tabelas de porcentagem média de sucesso, a nova heurística CNIT é superior às demais. Mesmo nos exemplos em que ela não possui o menor desvio relativo médio, ele não se desvia mais que 4,61% do mínimo entre as heurísticas. Este desvio máximo ocorre na classe de problemas $n = 100$ e $m = 5$ dos grupos SSD_100 e SSD_125, em que a heurística SETUP é superior, como esperado.
- O último aspecto notável das tabelas de desvio relativo médio é o valor próximo de zero ou mesmo negativo para alguns exemplos: isso significa a estabelecimento de novas soluções para tais problemas. Este aspecto é observado no exemplo com $n = 500$ e $m = 20$ dos grupos SSD_100 e SSD_125.
- Com relação ao tempo médio de processamento, o que se observa é que, apesar do cálculo eficiente de *makespan* alterado pela introdução dos parâmetros *XR* e *QR*, o método CNIT ainda é intrinsecamente limitado por tempo computacional proporcional a mn^3 , enquanto os demais são limitados por mn^2 , o que faz, por exemplo, com que os tempos para os problemas maiores sejam cerca de 100 vezes maiores que o tempo computacional do método NEH-RB.

4.2.2. Avaliação da Busca Local Descendente sobre as soluções das heurísticas construtivas

Como observado anteriormente no capítulo 1, as heurísticas construtivas são a base para processos melhorativos. Um dos métodos mais simples e mais difundidos na literatura é o de busca local. Nestes métodos são procuradas soluções melhores na vizinhança de uma dada solução inicial. Estes métodos, em geral, encontram ótimos

locais em que a distância para a solução ótima dependerá da qualidade da solução inicial e da vizinhança procurada.

Aqui, cada um dos métodos analisados é submetido à busca local descendente de proposta em Ruiz e Stützle (2008), que além de ser computacionalmente eficiente, apresentou bons resultados em seu trabalho original. Esta busca local foi descrita no capítulo 3.

Cabe observar que, como a direção da busca local é escolhida aleatoriamente, a solução não é estática como no caso das soluções encontradas na seção anterior. Por isso os resultados a seguir referem-se à média e ao mínimo dos resultados resolvidos 10 vezes pelos métodos.

As próximas tabelas apresentam as porcentagens de sucesso para os métodos heurísticos analisados na seção anterior e, na sequência, as tabelas com os desvios relativos médios e tempos de computação.

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	0%	0%	0%	50%	50%
		50%	10%	20%	10%	30%
20	10	0%	0%	10%	60%	30%
		20%	20%	40%	20%	0%
20	20	0%	20%	0%	40%	40%
		10%	10%	30%	30%	20%
50	5	0%	0%	0%	60%	40%
		10%	0%	20%	40%	30%
50	10	0%	0%	0%	50%	50%
		10%	20%	10%	40%	20%
50	20	0%	0%	0%	50%	50%
		10%	20%	10%	10%	50%
100	5	50%	0%	0%	20%	40%
		40%	10%	0%	30%	20%
100	10	0%	0%	0%	60%	40%
		10%	0%	0%	50%	40%
100	20	0%	0%	0%	60%	40%
		0%	0%	10%	50%	40%
200	10	0%	0%	0%	30%	70%
		0%	10%	0%	10%	80%
200	20	0%	0%	0%	60%	40%
		0%	0%	0%	50%	50%
500	20	0%	0%	0%	10%	90%
		0%	0%	0%	10%	90%
Médias		4%	2%	1%	46%	48%
		13%	8%	12%	29%	39%

Tabela 5.13 – Porcentagem de sucesso para a solução com busca local do grupo SSD_10, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	20%	20%	0%	40%	30%
		20%	20%	10%	20%	30%
20	10	0%	0%	0%	30%	70%
		50%	30%	0%	10%	10%
20	20	10%	0%	10%	40%	40%
		20%	0%	40%	30%	10%
50	5	90%	0%	0%	10%	0%
		80%	0%	10%	10%	0%
50	10	10%	0%	0%	50%	40%
		40%	20%	0%	40%	0%
50	20	10%	0%	0%	10%	80%
		10%	40%	10%	0%	50%
100	5	90%	10%	0%	0%	0%
		100%	0%	0%	0%	0%
100	10	20%	20%	0%	30%	30%
		40%	30%	0%	10%	20%
100	20	0%	0%	0%	30%	70%
		20%	0%	0%	30%	50%
200	10	90%	10%	0%	0%	0%
		100%	0%	0%	0%	0%
200	20	0%	10%	0%	0%	90%
		20%	20%	0%	0%	60%
500	20	100%	0%	0%	0%	0%
		100%	0%	0%	0%	0%
Médias		37%	6%	1%	20%	38%
		50%	13%	6%	13%	19%

Tabela 5.14 – Porcentagem de sucesso para a solução com busca local do grupo SSD_50, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	50%	50%	0%	0%	0%
		50%	20%	10%	20%	0%
20	10	10%	0%	10%	20%	60%
		20%	30%	20%	0%	30%
20	20	20%	10%	10%	50%	10%
		10%	30%	20%	20%	20%
50	5	80%	20%	0%	0%	0%
		80%	20%	0%	0%	0%
50	10	50%	20%	0%	0%	30%
		40%	30%	0%	0%	30%
50	20	0%	20%	0%	30%	50%
		30%	30%	0%	20%	20%
100	5	90%	10%	0%	0%	0%
		80%	20%	0%	0%	0%
100	10	80%	10%	0%	0%	10%
		70%	20%	0%	0%	10%
100	20	30%	0%	0%	10%	60%
		60%	20%	0%	10%	10%
200	10	80%	20%	0%	0%	0%
		80%	20%	0%	0%	0%
200	20	90%	10%	0%	0%	0%
		80%	20%	0%	0%	0%
500	20	90%	10%	0%	0%	0%
		90%	10%	0%	0%	0%
Médias		56%	15%	2%	9%	18%
		58%	23%	4%	6%	10%

Tabela 5.15 – Porcentagem de sucesso para a solução com busca local do grupo SSD_100, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	20%	40%	10%	10%	20%
		20%	20%	20%	30%	10%
20	10	20%	20%	0%	20%	40%
		40%	10%	20%	10%	20%
20	20	0%	0%	20%	40%	40%
		30%	10%	20%	30%	10%
50	5	80%	20%	0%	0%	0%
		70%	30%	0%	0%	0%
50	10	70%	10%	0%	0%	20%
		70%	20%	0%	0%	10%
50	20	50%	0%	0%	0%	50%
		50%	10%	0%	0%	40%
100	5	100%	0%	0%	0%	0%
		80%	20%	0%	0%	0%
100	10	70%	30%	0%	0%	0%
		70%	30%	0%	0%	0%
100	20	50%	20%	0%	0%	30%
		80%	20%	0%	0%	0%
200	10	90%	10%	0%	0%	0%
		100%	0%	0%	0%	0%
200	20	80%	20%	0%	0%	0%
		70%	30%	0%	0%	0%
500	20	100%	0%	0%	0%	0%
		90%	10%	0%	0%	0%
Médias		61%	14%	3%	6%	17%
		64%	18%	5%	6%	8%

Tabela 5.16 – Porcentagem de sucesso para a solução com busca local do grupo SSD_125, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)

Figura 5.4: Porcentagem de sucesso para as soluções com busca local, considerando as médias (barras) e mínimos (linhas)

Depois da etapa de busca local descendente nota-se um crescimento no equilíbrio entre os métodos, porém com destaque para a nova heurística CNIT no grupo SSD_10 e nos exemplos com menor número de máquinas e tarefas dos demais grupos; e para a heurística SETUP, que domina para os grupos SSD_100 e SSD_125, em especial para os exemplos com maior número de máquinas e tarefas.

A próxima avaliação é através das tabelas de desvio relativo médio da média e o mínimo de 10 rodadas do programa e o tempo médio computacional de cada exemplo, apresentadas a seguir.

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	2,67%	2,86%	2,28%	1,87%	1,61%
		1,03%	1,07%	1,05%	1,12%	0,95%
20	10	3,41%	3,71%	3,36%	2,37%	2,74%
		1,38%	1,70%	1,37%	1,60%	1,68%
20	20	3,29%	2,84%	2,80%	2,28%	2,38%
		1,70%	1,50%	1,16%	1,57%	1,55%
50	5	2,64%	3,03%	2,55%	1,93%	1,93%
		1,54%	1,95%	1,71%	1,41%	1,44%
50	10	4,90%	4,54%	4,82%	3,58%	3,76%
		3,46%	3,36%	3,43%	2,84%	2,86%
50	20	4,75%	4,71%	4,78%	3,83%	3,86%
		3,42%	3,17%	3,44%	3,17%	2,79%
100	5	1,78%	1,96%	2,02%	1,75%	1,77%
		1,25%	1,46%	1,50%	1,36%	1,40%
100	10	3,15%	3,24%	3,36%	2,24%	2,25%
		2,34%	2,27%	2,44%	1,79%	1,79%
100	20	4,39%	4,33%	4,33%	3,01%	3,17%
		3,23%	3,28%	3,30%	2,50%	2,56%
200	10	2,17%	2,25%	2,41%	1,69%	1,56%
		1,61%	1,64%	1,87%	1,35%	1,19%
200	20	3,49%	3,52%	3,52%	1,98%	1,92%
		2,65%	2,77%	2,77%	1,64%	1,58%
500	20	2,18%	2,24%	2,44%	0,94%	0,75%
		1,68%	1,79%	2,01%	0,76%	0,55%
	Médias	3,24%	3,27%	3,22%	2,29%	2,31%
		2,11%	2,16%	2,17%	1,76%	1,70%

Tabela 5.17 – Desvio relativo médio em relação ao *BM* (média na primeira linha, em negrito e mínimo na segunda) para a busca local do grupo SSD_10

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	<i>0,0014</i>	<i>0,0012</i>	<i>0,0013</i>	<i>0,0009</i>	<i>0,0019</i>
20	10	<i>0,0020</i>	<i>0,0020</i>	<i>0,0021</i>	<i>0,0016</i>	<i>0,0016</i>
20	20	<i>0,0037</i>	<i>0,0038</i>	<i>0,0038</i>	<i>0,0026</i>	<i>0,0047</i>
50	5	<i>0,0106</i>	<i>0,0105</i>	<i>0,0102</i>	<i>0,0071</i>	<i>0,0158</i>
50	10	<i>0,0194</i>	<i>0,0193</i>	<i>0,0187</i>	<i>0,0139</i>	<i>0,0266</i>
50	20	<i>0,0337</i>	<i>0,0346</i>	<i>0,0340</i>	<i>0,0232</i>	<i>0,0469</i>
100	5	<i>0,0634</i>	<i>0,0621</i>	<i>0,0623</i>	<i>0,0474</i>	<i>0,1194</i>
100	10	<i>0,1028</i>	<i>0,1056</i>	<i>0,1039</i>	<i>0,0732</i>	<i>0,1885</i>
100	20	<i>0,1917</i>	<i>0,2052</i>	<i>0,1878</i>	<i>0,1265</i>	<i>0,3253</i>
200	10	<i>0,6882</i>	<i>0,7057</i>	<i>0,6838</i>	<i>0,4694</i>	<i>1,8173</i>
200	20	<i>1,4391</i>	<i>1,3715</i>	<i>1,3919</i>	<i>0,9679</i>	<i>3,2837</i>
500	20	<i>18,3457</i>	<i>18,5400</i>	<i>17,9513</i>	<i>13,8987</i>	<i>77,6574</i>

Tabela 5.18 – Tempo médio computacional para a busca local do grupo SSD_10

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	4,81%	5,15%	5,84%	4,34%	4,88%
		2,74%	2,73%	3,70%	2,92%	2,91%
20	10	4,47%	4,78%	4,92%	3,96%	3,49%
		2,33%	2,38%	2,77%	2,69%	2,61%
20	20	3,39%	3,56%	3,56%	2,95%	3,04%
		1,97%	1,94%	1,84%	2,07%	1,85%
50	5	4,92%	5,88%	7,25%	6,27%	6,06%
		3,62%	4,48%	5,33%	4,78%	4,81%
50	10	5,91%	6,28%	7,13%	5,57%	5,53%
		4,13%	4,78%	5,46%	4,45%	4,40%
50	20	5,10%	5,45%	5,67%	4,72%	4,36%
		3,93%	3,91%	4,07%	3,92%	3,35%
100	5	2,85%	3,82%	6,52%	5,89%	5,19%
		2,02%	2,99%	5,08%	4,97%	4,24%
100	10	4,40%	4,62%	5,70%	4,88%	4,51%
		3,29%	3,31%	4,12%	3,98%	3,71%
100	20	4,58%	4,74%	5,09%	4,14%	4,00%
		3,24%	3,68%	4,03%	3,36%	3,12%
200	10	1,71%	2,21%	4,71%	3,80%	2,99%
		1,01%	1,52%	3,66%	3,01%	2,48%
200	20	2,64%	2,72%	4,09%	2,97%	2,25%
		1,87%	1,97%	3,08%	2,40%	1,80%
500	20	-0,38%	0,07%	2,81%	1,78%	0,61%
		-0,84%	-0,42%	2,06%	1,40%	0,39%
Médias		3,70%	4,11%	5,27%	4,27%	3,91%
		2,44%	2,77%	3,77%	3,33%	2,97%

Tabela 5.19 – Desvio relativo médio em relação ao *BM* (média na primeira linha, em negrito e mínimo na segunda) para a busca local do grupo SSD_50

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	<i>0,0011</i>	<i>0,0011</i>	<i>0,0011</i>	<i>0,0008</i>	<i>0,0009</i>
20	10	<i>0,0017</i>	<i>0,0018</i>	<i>0,0019</i>	<i>0,0015</i>	<i>0,0024</i>
20	20	<i>0,0034</i>	<i>0,0034</i>	<i>0,0038</i>	<i>0,0028</i>	<i>0,0045</i>
50	5	<i>0,0095</i>	<i>0,0096</i>	<i>0,0109</i>	<i>0,0072</i>	<i>0,0161</i>
50	10	<i>0,0166</i>	<i>0,0159</i>	<i>0,0181</i>	<i>0,0117</i>	<i>0,0258</i>
50	20	<i>0,0318</i>	<i>0,0311</i>	<i>0,0315</i>	<i>0,0224</i>	<i>0,0492</i>
100	5	<i>0,0533</i>	<i>0,0551</i>	<i>0,0650</i>	<i>0,0478</i>	<i>0,1246</i>
100	10	<i>0,0904</i>	<i>0,0907</i>	<i>0,1034</i>	<i>0,0705</i>	<i>0,1878</i>
100	20	<i>0,1692</i>	<i>0,1788</i>	<i>0,1867</i>	<i>0,1110</i>	<i>0,3261</i>
200	10	<i>0,5543</i>	<i>0,5831</i>	<i>0,7209</i>	<i>0,4975</i>	<i>1,8463</i>
200	20	<i>1,2414</i>	<i>1,2858</i>	<i>1,3986</i>	<i>0,9433</i>	<i>3,3338</i>
500	20	<i>16,7280</i>	<i>16,3515</i>	<i>19,3676</i>	<i>13,1029</i>	<i>79,2586</i>

Tabela 5.20 – Tempo médio computacional para a busca local do grupo SSD_50

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	5,58%	6,13%	7,88%	7,72%	6,47%
		3,28%	4,36%	4,47%	5,80%	5,71%
20	10	5,07%	5,63%	6,04%	5,32%	4,85%
		3,14%	3,38%	3,12%	4,44%	3,28%
20	20	4,48%	4,40%	4,71%	3,49%	3,64%
		2,98%	2,78%	2,32%	2,41%	2,55%
50	5	5,73%	7,03%	10,07%	9,70%	8,48%
		4,04%	5,30%	7,66%	7,74%	7,27%
50	10	6,60%	6,89%	9,11%	7,83%	6,92%
		4,97%	5,19%	6,76%	6,16%	5,46%
50	20	6,27%	6,37%	7,13%	5,85%	5,61%
		4,63%	4,50%	5,42%	4,58%	4,73%
100	5	2,63%	3,59%	8,84%	8,65%	7,15%
		1,71%	2,55%	7,12%	7,05%	6,28%
100	10	3,78%	4,52%	7,49%	6,66%	5,23%
		2,58%	3,35%	5,80%	5,39%	4,47%
100	20	4,44%	4,68%	6,65%	5,20%	4,28%
		3,31%	3,48%	5,00%	4,33%	3,69%
200	10	-0,16%	0,53%	6,29%	5,54%	2,89%
		-0,92%	-0,18%	5,00%	4,49%	2,39%
200	20	1,48%	2,11%	5,09%	3,77%	2,20%
		0,72%	1,35%	4,07%	3,10%	1,78%
500	20	-2,31%	-1,81%	3,50%	2,35%	-0,08%
		-2,73%	-2,30%	2,63%	1,86%	-0,23%
Médias		3,63%	4,17%	6,90%	6,01%	4,81%
		2,31%	2,81%	4,95%	4,78%	3,95%

Tabela 5.21 – Desvio relativo médio em relação ao *BM* (média na primeira linha, em negrito e mínimo na segunda) para a busca local do grupo SSD_100

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	<i>0,0009</i>	<i>0,0008</i>	<i>0,0010</i>	<i>0,0010</i>	<i>0,0011</i>
20	10	<i>0,0016</i>	<i>0,0017</i>	<i>0,0017</i>	<i>0,0015</i>	<i>0,0021</i>
20	20	<i>0,0031</i>	<i>0,0031</i>	<i>0,0034</i>	<i>0,0026</i>	<i>0,0042</i>
50	5	<i>0,0085</i>	<i>0,0082</i>	<i>0,0098</i>	<i>0,0072</i>	<i>0,0166</i>
50	10	<i>0,0151</i>	<i>0,0149</i>	<i>0,0163</i>	<i>0,0123</i>	<i>0,0254</i>
50	20	<i>0,0278</i>	<i>0,0283</i>	<i>0,0310</i>	<i>0,0225</i>	<i>0,0487</i>
100	5	<i>0,0460</i>	<i>0,0457</i>	<i>0,0602</i>	<i>0,0444</i>	<i>0,1197</i>
100	10	<i>0,0823</i>	<i>0,0773</i>	<i>0,1005</i>	<i>0,0672</i>	<i>0,1867</i>
100	20	<i>0,1488</i>	<i>0,1661</i>	<i>0,1730</i>	<i>0,1108</i>	<i>0,3254</i>
200	10	<i>0,5248</i>	<i>0,5507</i>	<i>0,6654</i>	<i>0,4683</i>	<i>1,8578</i>
200	20	<i>1,1241</i>	<i>1,0885</i>	<i>1,4110</i>	<i>0,8859</i>	<i>3,2306</i>
500	20	<i>13,5636</i>	<i>14,6539</i>	<i>18,9306</i>	<i>13,2159</i>	<i>75,1891</i>

Tabela 5.22 – Tempo médio computacional para a busca local do grupo SSD_100

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	6,00%	5,45%	8,04%	7,06%	6,36%
		4,17%	4,72%	5,02%	4,92%	5,30%
20	10	5,78%	6,00%	6,36%	6,31%	5,59%
		3,25%	3,64%	3,30%	5,44%	4,27%
20	20	4,43%	4,82%	4,81%	3,93%	4,07%
		2,47%	3,03%	2,75%	2,90%	3,18%
50	5	5,30%	6,57%	10,77%	10,33%	8,41%
		3,95%	5,19%	8,43%	8,16%	6,80%
50	10	6,64%	6,97%	9,11%	8,35%	7,45%
		4,78%	5,16%	6,77%	7,21%	6,13%
50	20	6,01%	6,31%	7,34%	6,58%	5,76%
		4,48%	4,93%	5,62%	5,34%	4,60%
100	5	2,07%	3,40%	8,71%	9,61%	7,56%
		1,27%	2,29%	7,18%	7,65%	6,48%
100	10	3,96%	4,52%	8,68%	7,82%	5,75%
		2,80%	3,20%	6,84%	6,65%	4,92%
100	20	4,00%	4,33%	6,94%	5,65%	4,31%
		2,89%	3,19%	5,28%	4,90%	3,86%
200	10	-0,60%	0,47%	6,58%	6,09%	3,18%
		-1,23%	-0,26%	5,26%	4,92%	2,61%
200	20	1,11%	1,46%	5,42%	4,17%	2,03%
		0,38%	0,63%	4,27%	3,45%	1,66%
500	20	-2,91%	-2,55%	3,53%	2,72%	-0,47%
		-3,38%	-2,97%	2,51%	2,16%	-0,64%
Médias		3,48%	3,98%	7,19%	6,55%	5,00%
		2,15%	2,73%	5,27%	5,31%	4,10%

Tabela 5.23 – Desvio relativo médio em relação ao *BM* (média na primeira linha, em negrito e mínimo na segunda) para a busca local do grupo SSD_125

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	0,0009	0,0008	0,0010	0,0008	0,0009
20	10	0,0016	0,0017	0,0018	0,0014	0,0028
20	20	0,0032	0,0035	0,0035	0,0025	0,0031
50	5	0,0080	0,0084	0,0096	0,0074	0,0134
50	10	0,0141	0,0146	0,0170	0,0118	0,0247
50	20	0,0279	0,0272	0,0323	0,0225	0,0449
100	5	0,0450	0,0469	0,0592	0,0443	0,1154
100	10	0,0756	0,0775	0,0926	0,0603	0,1765
100	20	0,1506	0,1428	0,1723	0,1080	0,3128
200	10	0,4918	0,5105	0,6379	0,4676	1,7970
200	20	1,0477	1,0287	1,3521	0,8486	3,1890
500	20	13,9019	13,5961	18,8395	13,5093	74,6480

Tabela 5.24 – Tempo médio computacional para a busca local do grupo SSD_125

Figura 5.5: Desvio relativo médio em relação ao *BM* para a busca local (médias, mínimos e por classe de problema)

Figura 5.6: Tempos médios computacionais para a busca local

Estas tabelas evidenciam o equilíbrio entre os métodos após a etapa de busca local descendente. E permitem as seguintes conclusões:

- Como a estratégia de inserção utilizada na busca local é a mesma utilizada na etapa de construção da sequência dos métodos NEHT-RB e CNIT, a eficiência alcançada para os métodos que utilizam métodos construtivos diversos (como no caso do SETUP, TOTAL e UBX+S) é bem maior, chegando a obter resultados com desvio relativo médio em relação ao BM cerca de 4 vezes menores do que aqueles obtidos com o método construtivo sem a etapa de busca local, enquanto os melhores resultados alcançados para os métodos NEHT-RB e CNIT são cerca de 2 vezes menores.
- Com todas as soluções iniciais os resultados após a busca local são muito próximos, com destaque para o efeito observado nas tabelas anteriores: aqui nota-se que os desvios relativos médios crescem para a nova heurística CNIT à

medida que a distribuição do tempo de setup torna-se maior (ou seja, de SSD_10 para SSD_125), enquanto a heurística SETUP melhora à mesma medida para os problemas maiores (em especial $n \geq 100$).

- São estabelecidos novos limites inferiores para o banco de dados indicados pelos valores negativos dos desvios relativos médios, em especial pela heurística SETUP, já nos exemplos com $n = 500$ e $m = 20$ do grupo SSD_50.
- Mesmo tratando-se de um método sujeito à aleatoriedade estatística da escolha da vizinhança, nota-se alta acuracidade na busca local, observada pelas pequenas diferenças entre os valores de mínimo desvio médio relativo em relação ao BM e a média entre as 10 rodadas do programa.
- Em comparação com os tempos computacionais dos métodos após a aplicação da busca local, nota-se que ela torna-se “custosa” em termos computacionais para os métodos SETUP, TOTAL, UBX+S e NEHT-RB, tomando entre 5 e 20 vezes mais tempo de processamento. Já para o método CNIT, o acréscimo é de apenas 2% a 3%. Este baixo “custo” computacional relativo é explicado por 2 fatores: o alto tempo de computação do método original, limitado por mn^3 e a semelhança no método de inserção das tarefas, dando pouca margem de ganho, como observado nos itens anteriores.

4.2.3. A avaliação do efeito da solução inicial para a meta-heurística IG e sua comparação com a nova meta-heurística CNIT-IG

Ruiz e Stützle (2008) em sua experimentação computacional não avaliaram o efeito de diferentes soluções iniciais na qualidade da solução obtida depois de aplicar o método meta-heurístico IG. Neste trabalho esta avaliação é realizada utilizando as heurísticas construtivas que foram comparadas entre si. Nas tabelas de resultados, os métodos serão identificados pelas diferentes soluções iniciais (SETUP, TOTAL, UBX+S, NEHT-RB, CNIT). É importante observar que o método original utiliza a solução inicial obtida através do método NEHT-RB.

O critério de parada estabelecido aqui é o mesmo utilizado no trabalho original, ou seja, o tempo computacional limitado em $\left(n \cdot \frac{m}{2} \cdot \frac{t}{1000}\right)$ segundos. Os valores utilizados para t para a avaliação da solução inicial da meta-heurística IG foram $t = 30$ e $t = 60$, de maneira que, os problemas com $n = 500$ e $m = 20$ tiveram limites de 150.000 ms (ou 2,5 min) para $t = 30$ e 200.000 ms (ou 5 min) para $t = 60$, por exemplo.

Estes limites permitem o nivelamento das soluções iniciais. O tempo computacional médio do método CNIT é bastante superior aos demais, enquanto o método NEHT-RB possui o menor tempo computacional médio. Se o limite de tempo estabelecido for suficiente para a solução inicial, o restante do tempo determinará quantas vezes o método IG poderá realizar as etapas de Destruição e Construção.

Como nas duas seções anteriores primeiramente será apresentada a porcentagem de sucesso de cada um dos métodos e em seguida o desvio relativo médio. Cada exemplo foi rodado 5 vezes, já que este método está sujeito ao mesmo efeito aleatório observado na busca local. Por isso são apresentados os resultados da média dos 5 resultados e também os valores de mínimo alcançados.

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	0%	10%	20%	40%	30%
		20%	30%	10%	30%	40%
20	10	0%	0%	10%	40%	50%
		10%	0%	40%	10%	40%
20	20	0%	10%	20%	40%	30%
		20%	20%	50%	20%	20%
50	5	10%	0%	10%	50%	30%
		10%	0%	10%	60%	30%
50	10	10%	0%	20%	30%	40%
		10%	0%	10%	40%	40%
50	20	0%	10%	30%	20%	40%
		0%	20%	30%	20%	30%
100	5	40%	10%	0%	20%	40%
		20%	30%	10%	20%	20%
100	10	0%	0%	0%	60%	50%
		10%	0%	20%	40%	30%
100	20	0%	10%	10%	50%	30%
		10%	10%	10%	40%	40%
200	10	0%	0%	10%	60%	30%
		0%	10%	10%	50%	30%
200	20	0%	0%	0%	50%	50%
		0%	0%	0%	50%	50%
500	20	0%	0%	0%	20%	80%
		0%	0%	0%	40%	60%
Médias		5%	4%	11%	40%	42%
		9%	10%	17%	35%	36%

Tabela 5.25 – Porcentagem de sucesso para as diferentes soluções iniciais na meta-heurística IG com $t = 30$ para o grupo SSD_10, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	10%	30%	10%	30%	20%
		40%	10%	10%	30%	10%
20	10	10%	20%	0%	30%	40%
		0%	30%	20%	40%	20%
20	20	20%	20%	10%	20%	30%
		20%	20%	0%	30%	40%
50	5	70%	10%	0%	20%	0%
		40%	10%	0%	20%	30%
50	10	30%	0%	0%	40%	30%
		20%	20%	0%	40%	20%
50	20	20%	0%	20%	0%	60%
		10%	10%	40%	0%	40%
100	5	100%	0%	0%	0%	0%
		100%	0%	0%	0%	0%
100	10	40%	30%	0%	20%	10%
		60%	10%	0%	20%	10%
100	20	0%	20%	10%	20%	50%
		0%	30%	0%	30%	40%
200	10	90%	10%	0%	0%	0%
		90%	10%	0%	0%	0%
200	20	60%	20%	0%	0%	20%
		50%	20%	0%	0%	30%
500	20	80%	20%	0%	0%	0%
		90%	10%	0%	0%	0%
Médias		44%	15%	4%	15%	22%
		43%	15%	6%	18%	20%

Tabela 5.26 – Porcentagem de sucesso para as diferentes soluções iniciais na meta-heurística IG com $t = 30$ para o grupo SSD_50, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	20%	40%	0%	10%	30%
		20%	40%	0%	20%	20%
20	10	30%	20%	20%	20%	10%
		40%	20%	0%	30%	10%
20	20	10%	20%	10%	40%	30%
		20%	10%	20%	20%	30%
50	5	90%	10%	0%	0%	0%
		70%	10%	10%	0%	10%
50	10	30%	50%	0%	0%	20%
		20%	70%	0%	10%	0%
50	20	20%	20%	10%	30%	20%
		10%	40%	10%	30%	10%
100	5	80%	20%	0%	0%	0%
		90%	20%	0%	0%	0%
100	10	70%	20%	0%	0%	10%
		60%	40%	0%	0%	0%
100	20	30%	20%	20%	20%	10%
		40%	30%	10%	20%	10%
200	10	90%	10%	0%	0%	0%
		90%	10%	0%	0%	0%
200	20	80%	20%	0%	0%	0%
		60%	40%	0%	0%	0%
500	20	100%	0%	0%	0%	0%
		90%	10%	0%	0%	0%
Médias		54%	21%	5%	10%	11%
		51%	28%	4%	11%	8%

Tabela 5.27 – Porcentagem de sucesso para as diferentes soluções iniciais na meta-heurística IG com $t = 30$ para o grupo SSD_100, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	20%	70%	0%	10%	0%
		10%	30%	10%	20%	30%
20	10	30%	20%	30%	10%	10%
		30%	20%	10%	20%	20%
20	20	20%	20%	0%	50%	10%
		30%	20%	0%	40%	20%
50	5	80%	20%	0%	0%	0%
		70%	20%	0%	10%	0%
50	10	40%	40%	0%	0%	20%
		40%	30%	0%	20%	10%
50	20	10%	0%	10%	10%	70%
		10%	10%	20%	10%	50%
100	5	100%	0%	0%	0%	0%
		100%	0%	0%	0%	0%
100	10	80%	20%	0%	0%	0%
		70%	30%	0%	0%	0%
100	20	40%	40%	0%	0%	20%
		40%	40%	10%	10%	0%
200	10	100%	0%	0%	0%	0%
		100%	0%	0%	0%	0%
200	20	80%	20%	0%	0%	0%
		90%	10%	0%	0%	0%
500	20	90%	10%	0%	0%	0%
		100%	0%	0%	0%	0%
Médias		58%	22%	3%	7%	11%
		58%	18%	4%	11%	11%

Tabela 5.28 – Porcentagem de sucesso para as diferentes soluções iniciais na meta-heurística IG com $t = 30$ para o grupo SSD_125, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)

Figura 5.7: Porcentagem de sucesso para as diferentes soluções iniciais na meta-heurística IG com $t = 30$, considerando médias e mínimos

O que fica evidente na análise destas tabelas é a superioridade da heurística SETUP como solução inicial para a heurística IG, principalmente quando o limite da distribuição dos tempos de setup é igual ou superior a 50% do limite da distribuição dos tempos de processamento; e o equilíbrio entre as heurísticas NEHT-RB e a nova heurística CNIT.

As próximas tabelas mostram as médias e mínimos dos desvios relativos médios, que permitem uma interpretação mais aprofundada dos métodos:

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	1,56%	1,59%	1,42%	1,31%	1,19%
		0,83%	0,74%	0,78%	0,74%	0,75%
20	10	1,84%	1,93%	1,81%	1,54%	1,55%
		0,97%	1,00%	0,90%	0,93%	0,98%
20	20	1,59%	1,55%	1,42%	1,24%	1,22%
		0,65%	0,80%	0,62%	0,78%	0,49%
50	5	1,58%	1,92%	1,62%	1,36%	1,34%
		1,08%	1,45%	1,23%	1,07%	1,09%
50	10	3,00%	2,82%	2,89%	2,52%	2,54%
		2,07%	2,21%	2,34%	1,96%	1,95%
50	20	2,75%	2,73%	2,46%	2,55%	2,33%
		2,20%	2,15%	1,90%	1,92%	1,83%
100	5	1,09%	1,11%	1,18%	1,10%	1,07%
		0,84%	0,82%	0,94%	0,88%	0,82%
100	10	1,64%	1,60%	1,59%	1,23%	1,29%
		1,25%	1,29%	1,19%	1,00%	1,02%
100	20	2,12%	1,97%	2,12%	1,60%	1,73%
		1,62%	1,44%	1,56%	1,28%	1,30%
200	10	0,87%	0,94%	1,06%	0,66%	0,65%
		0,65%	0,61%	0,79%	0,48%	0,49%
200	20	1,49%	1,42%	1,48%	0,79%	0,79%
		1,18%	1,06%	1,09%	0,59%	0,59%
500	20	1,33%	1,42%	1,59%	0,38%	0,36%
		1,07%	1,17%	1,38%	0,26%	0,24%
Médias		1,74%	1,75%	1,72%	1,36%	1,34%
		1,20%	1,23%	1,23%	0,99%	0,96%

Tabela 5.29 – Desvio relativo médio em relação ao *BM* das soluções IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_10

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	3,80%	3,72%	3,99%	3,15%	3,50%
		2,05%	2,23%	2,35%	1,74%	2,32%
20	10	3,03%	2,69%	2,91%	2,51%	2,50%
		2,10%	1,63%	1,45%	1,26%	1,57%
20	20	1,76%	1,67%	1,80%	1,64%	1,58%
		0,96%	0,85%	1,15%	1,07%	0,80%
50	5	4,11%	4,72%	5,31%	4,71%	4,86%
		3,34%	3,76%	4,08%	3,74%	3,71%
50	10	4,08%	4,48%	4,67%	4,01%	4,05%
		3,34%	3,49%	3,61%	3,16%	3,46%
50	20	3,03%	3,20%	2,99%	3,15%	2,61%
		2,30%	2,56%	2,26%	2,73%	1,95%
100	5	2,16%	3,06%	4,56%	4,17%	4,12%
		1,61%	2,55%	3,86%	3,57%	3,61%
100	10	2,77%	3,02%	3,42%	3,05%	3,10%
		2,13%	2,51%	2,80%	2,54%	2,51%
100	20	2,55%	2,48%	2,54%	2,34%	2,31%
		2,03%	1,98%	1,99%	1,83%	1,85%
200	10	0,48%	0,79%	2,05%	1,80%	1,66%
		0,08%	0,43%	1,60%	1,40%	1,29%
200	20	0,86%	0,92%	1,44%	1,12%	0,95%
		0,47%	0,57%	1,09%	0,76%	0,63%
500	20	-1,10%	-0,66%	1,22%	0,66%	0,25%
		-1,30%	-0,88%	0,91%	0,50%	0,09%
Médias		2,29%	2,51%	3,08%	2,69%	2,62%
		1,59%	1,81%	2,26%	2,02%	1,98%

Tabela 5.30 – Desvio relativo médio em relação ao *BM* das soluções IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_50

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	5,09%	4,95%	6,02%	6,06%	5,15%
		3,28%	3,08%	4,34%	3,98%	3,48%
20	10	3,35%	4,02%	4,26%	4,07%	3,83%
		2,14%	2,73%	3,06%	2,62%	2,51%
20	20	2,30%	2,38%	2,35%	2,28%	2,35%
		1,26%	1,43%	1,24%	1,49%	1,38%
50	5	5,23%	6,20%	8,01%	7,49%	7,16%
		4,35%	5,12%	6,02%	6,23%	6,03%
50	10	5,08%	4,81%	6,28%	6,00%	5,54%
		4,26%	3,62%	5,19%	4,93%	4,79%
50	20	3,90%	4,03%	4,19%	3,86%	3,98%
		3,24%	3,21%	3,33%	3,20%	3,10%
100	5	2,19%	3,01%	6,70%	6,60%	5,92%
		1,43%	2,43%	5,67%	5,66%	5,21%
100	10	2,80%	3,31%	4,58%	4,65%	3,93%
		2,16%	2,44%	3,62%	3,89%	3,40%
100	20	2,66%	2,84%	3,38%	3,04%	2,90%
		2,03%	2,16%	2,74%	2,59%	2,42%
200	10	-0,85%	-0,33%	3,08%	2,88%	1,93%
		-1,33%	-0,76%	2,40%	2,28%	1,50%
200	20	0,14%	0,52%	1,79%	1,64%	1,16%
		-0,13%	0,07%	1,32%	1,23%	0,78%
500	20	-2,88%	-2,39%	1,56%	0,95%	-0,37%
		-3,12%	-2,68%	1,19%	0,64%	-0,52%
Médias		2,42%	2,78%	4,35%	4,13%	3,62%
		1,63%	1,90%	3,34%	3,23%	2,84%

Tabela 5.31 – Desvio relativo médio em relação ao *BM* das soluções IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_100

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	5,02%	4,22%	6,19%	5,35%	5,43%
		3,74%	3,10%	4,15%	3,90%	3,83%
20	10	3,68%	3,95%	4,41%	4,43%	3,86%
		2,06%	2,50%	2,68%	3,24%	2,72%
20	20	2,41%	2,40%	2,96%	2,20%	2,42%
		1,28%	1,51%	1,87%	1,24%	1,49%
50	5	4,87%	5,94%	8,87%	8,58%	7,35%
		3,99%	4,80%	6,94%	7,01%	5,88%
50	10	5,18%	5,51%	6,79%	6,38%	6,16%
		4,28%	4,42%	5,60%	5,25%	5,02%
50	20	4,20%	4,28%	4,54%	4,23%	3,94%
		3,35%	3,65%	3,46%	3,46%	3,23%
100	5	1,69%	2,81%	6,95%	7,74%	6,40%
		1,11%	2,07%	5,88%	6,72%	5,65%
100	10	2,95%	3,39%	5,38%	5,14%	4,54%
		2,06%	2,63%	4,41%	4,38%	3,74%
100	20	2,44%	2,54%	3,45%	3,09%	2,89%
		1,77%	1,81%	2,68%	2,49%	2,40%
200	10	-1,35%	-0,38%	3,29%	3,19%	2,11%
		-1,79%	-0,91%	2,65%	2,66%	1,56%
200	20	-0,18%	0,09%	2,01%	1,59%	1,03%
		-0,78%	-0,32%	1,49%	0,95%	0,64%
500	20	-3,47%	-3,14%	1,53%	1,19%	-0,71%
		-3,80%	-3,40%	0,99%	0,84%	-0,83%
Médias		2,29%	2,63%	4,70%	4,43%	3,79%
		1,44%	1,82%	3,57%	3,51%	2,94%

Tabela 5.32 – Desvio relativo médio em relação ao *BM* das soluções IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_125

Figura 5.8: Desvio relativo médio em relação ao *BM* das soluções IG com $t = 30$, considerando médias, mínimos e por classe de problemas

Sustentando as características observadas nas tabelas de porcentagem de sucesso, os menores desvios relativos médios são observados quando a solução inicial é CNIT para SSD_10 e para os exemplos menores de SSD_50 (com $n = 20$). A superioridade de SETUP como solução inicial de SSD_50, SSD_100 e SSD_125 é tão incontestável que se notam desvios negativos para $n = 500$ em SSD_50; $n = 200$, $m = 10$ e $n = 500$ em SSD_100; e $n = \{200, 500\}$ em SSD_125, estabelecendo novos limites inferiores.

Na sequência, o efeito do limite de tempo foi testado, trabalhando-se com $t = 60$, ou seja, o tempo permitido para o programa dobra. Seguem as tabelas:

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	0%	0%	10%	40%	60%
		30%	10%	50%	20%	30%
20	10	40%	0%	0%	40%	20%
		50%	10%	10%	30%	30%
20	20	10%	0%	10%	50%	30%
		30%	20%	20%	20%	30%
50	5	0%	0%	10%	60%	30%
		30%	10%	0%	30%	30%
50	10	20%	0%	10%	30%	40%
		20%	10%	10%	40%	30%
50	20	0%	20%	20%	10%	50%
		10%	40%	30%	0%	30%
100	5	30%	10%	10%	10%	40%
		40%	20%	10%	20%	20%
100	10	0%	0%	10%	50%	40%
		0%	10%	0%	40%	50%
100	20	0%	10%	10%	40%	40%
		10%	10%	0%	40%	40%
200	10	10%	0%	10%	20%	60%
		40%	0%	10%	30%	20%
200	20	0%	0%	0%	50%	50%
		0%	0%	0%	60%	40%
500	20	0%	0%	0%	30%	70%
		0%	0%	0%	30%	70%
Médias		9%	3%	8%	36%	44%
		22%	12%	12%	30%	35%

Tabela 5.33 – Porcentagem de sucesso para a solução IG com $t = 60$ do grupo SSD_10, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	20%	30%	20%	20%	20%
		10%	30%	20%	30%	10%
20	10	10%	10%	30%	40%	20%
		20%	20%	20%	30%	30%
20	20	20%	20%	10%	30%	20%
		30%	0%	30%	20%	40%
50	5	50%	20%	0%	20%	10%
		40%	50%	0%	0%	20%
50	10	20%	30%	10%	30%	10%
		50%	20%	10%	10%	10%
50	20	20%	20%	0%	10%	50%
		10%	40%	10%	10%	30%
100	5	100%	0%	0%	0%	0%
		80%	20%	0%	0%	0%
100	10	60%	30%	0%	10%	0%
		50%	20%	10%	10%	20%
100	20	10%	30%	10%	20%	30%
		20%	30%	10%	10%	30%
200	10	90%	10%	0%	0%	0%
		80%	20%	0%	0%	0%
200	20	30%	10%	10%	20%	30%
		40%	20%	0%	10%	30%
500	20	100%	0%	0%	0%	0%
		100%	0%	0%	0%	0%
Médias		44%	18%	8%	17%	16%
		44%	23%	9%	11%	18%

Tabela 5.34 – Porcentagem de sucesso para a solução IG com $t = 60$ do grupo SSD_50, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	40%	40%	0%	0%	20%
		40%	30%	0%	0%	30%
20	10	20%	30%	30%	0%	20%
		30%	30%	30%	0%	20%
20	20	10%	0%	30%	30%	30%
		10%	20%	20%	40%	20%
50	5	70%	30%	0%	0%	0%
		60%	30%	10%	0%	0%
50	10	50%	20%	10%	10%	10%
		20%	40%	0%	20%	20%
50	20	10%	10%	10%	10%	60%
		30%	0%	0%	0%	70%
100	5	90%	10%	0%	0%	0%
		90%	10%	0%	0%	0%
100	10	80%	20%	0%	0%	0%
		80%	20%	0%	0%	0%
100	20	30%	30%	10%	10%	20%
		30%	40%	0%	20%	10%
200	10	80%	20%	0%	0%	0%
		80%	20%	0%	0%	0%
200	20	100%	0%	0%	0%	0%
		90%	10%	0%	0%	0%
500	20	100%	0%	0%	0%	0%
		100%	0%	0%	0%	0%
Médias		57%	18%	8%	5%	13%
		55%	21%	5%	7%	14%

Tabela 5.35 – Porcentagem de sucesso para a solução IG com $t = 60$ do grupo SSD_100, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	20%	40%	20%	10%	10%
		20%	50%	10%	20%	10%
20	10	10%	30%	20%	0%	40%
		30%	20%	30%	10%	20%
20	20	10%	20%	0%	60%	10%
		30%	10%	10%	40%	30%
50	5	80%	20%	0%	0%	0%
		70%	30%	0%	0%	0%
50	10	60%	30%	0%	0%	10%
		30%	50%	10%	10%	0%
50	20	30%	0%	20%	0%	50%
		20%	10%	20%	0%	50%
100	5	100%	0%	0%	0%	0%
		100%	0%	0%	0%	0%
100	10	60%	40%	0%	0%	0%
		50%	50%	0%	0%	0%
100	20	40%	30%	0%	10%	20%
		60%	30%	0%	10%	0%
200	10	100%	0%	0%	0%	0%
		100%	0%	0%	0%	0%
200	20	70%	30%	0%	0%	0%
		80%	20%	0%	0%	0%
500	20	80%	20%	0%	0%	0%
		70%	30%	0%	0%	0%
Médias		55%	22%	5%	7%	12%
		55%	25%	7%	8%	9%

Tabela 5.36 – Porcentagem de sucesso para a solução IG com $t = 60$ do grupo SSD_125, considerando a média (1ª linha, em negrito) e o mínimo (2ª linha)

Figura 5.9: Porcentagem de sucesso para a solução IG com $t = 60$ considerado as médias e os mínimos

Nota-se que o desempenho do método para cada uma das diferentes soluções iniciais não se altera significativamente com o aumento no limite de tempo de computação. Em seguida pode-se avaliar seu efeito no desvio relativo médio.

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	1,76%	1,49%	1,26%	1,22%	1,10%
		0,77%	0,77%	0,63%	0,71%	0,64%
20	10	1,50%	1,61%	1,66%	1,21%	1,52%
		0,77%	0,88%	1,01%	0,75%	0,75%
20	20	1,16%	1,27%	1,11%	0,95%	0,96%
		0,57%	0,75%	0,52%	0,52%	0,47%
50	5	1,40%	1,60%	1,39%	1,22%	1,16%
		0,93%	1,17%	1,06%	0,89%	0,92%
50	10	2,48%	2,43%	2,55%	2,20%	2,27%
		1,95%	1,83%	2,04%	1,63%	1,86%
50	20	2,38%	2,21%	2,15%	2,25%	2,08%
		1,88%	1,59%	1,60%	1,79%	1,63%
100	5	0,99%	1,00%	1,03%	0,89%	0,92%
		0,74%	0,73%	0,78%	0,70%	0,73%
100	10	1,35%	1,39%	1,24%	1,04%	1,00%
		1,08%	1,02%	0,91%	0,77%	0,74%
100	20	1,76%	1,56%	1,73%	1,39%	1,44%
		1,31%	1,15%	1,31%	1,03%	1,01%
200	10	0,64%	0,70%	0,73%	0,53%	0,46%
		0,41%	0,47%	0,52%	0,36%	0,32%
200	20	1,16%	1,09%	1,15%	0,61%	0,56%
		0,82%	0,71%	0,81%	0,40%	0,32%
500	20	1,10%	1,18%	1,34%	0,26%	0,23%
		0,86%	1,00%	1,07%	0,15%	0,10%
Médias		1,47%	1,46%	1,44%	1,15%	1,14%
		1,01%	1,00%	1,02%	0,81%	0,79%

Tabela 5.37 – Desvio relativo médio em relação ao *BM* das soluções IG com $t = 60$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_10

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	3,32%	3,26%	3,47%	3,37%	3,06%
		2,19%	1,91%	2,15%	1,96%	2,14%
20	10	2,46%	2,46%	2,46%	2,29%	2,34%
		1,64%	1,52%	1,40%	1,35%	1,43%
20	20	1,57%	1,50%	1,48%	1,46%	1,29%
		0,82%	0,85%	0,71%	0,92%	0,66%
50	5	3,98%	4,26%	5,00%	4,44%	4,43%
		3,20%	3,17%	3,89%	3,67%	3,49%
50	10	3,61%	3,50%	3,80%	3,54%	3,62%
		2,76%	2,71%	2,86%	3,00%	2,65%
50	20	2,67%	2,61%	2,73%	2,71%	2,43%
		2,07%	1,87%	2,14%	2,15%	1,91%
100	5	2,03%	2,63%	4,09%	3,79%	3,73%
		1,60%	2,07%	3,27%	3,05%	3,16%
100	10	2,39%	2,50%	2,76%	2,76%	2,70%
		1,85%	1,92%	2,06%	2,20%	2,18%
100	20	2,03%	2,08%	2,12%	2,02%	1,96%
		1,50%	1,54%	1,62%	1,59%	1,49%
200	10	0,31%	0,67%	1,58%	1,35%	1,28%
		-0,02%	0,39%	1,15%	1,00%	0,88%
200	20	0,59%	0,67%	0,96%	0,69%	0,66%
		0,28%	0,34%	0,63%	0,42%	0,37%
500	20	-1,21%	-0,79%	0,80%	0,33%	0,01%
		-1,43%	-1,01%	0,58%	0,11%	-0,13%
Médias		1,98%	2,11%	2,60%	2,40%	2,29%
		1,37%	1,44%	1,87%	1,79%	1,68%

Tabela 5.38 – Desvio relativo médio em relação ao *BM* das soluções IG com $t = 60$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_50

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	4,14%	4,43%	5,25%	5,22%	4,71%
		2,34%	2,70%	3,49%	3,56%	3,12%
20	10	3,22%	2,96%	3,25%	3,58%	3,18%
		2,01%	1,92%	1,61%	2,58%	2,34%
20	20	2,08%	2,18%	1,94%	1,89%	1,95%
		1,31%	1,15%	1,06%	1,15%	1,22%
50	5	4,84%	5,77%	7,01%	6,96%	6,82%
		3,90%	4,54%	5,35%	5,71%	5,70%
50	10	4,62%	4,65%	5,39%	5,27%	4,97%
		3,77%	3,55%	4,35%	4,15%	4,03%
50	20	3,68%	3,74%	3,70%	3,62%	3,33%
		2,91%	2,99%	3,07%	3,05%	2,62%
100	5	1,77%	2,64%	6,24%	5,98%	5,29%
		1,14%	2,15%	5,22%	4,55%	4,46%
100	10	2,29%	2,81%	4,00%	3,87%	3,59%
		1,66%	2,09%	3,26%	3,04%	2,89%
100	20	2,32%	2,44%	3,02%	2,61%	2,47%
		1,82%	1,79%	2,44%	2,08%	1,88%
200	10	-1,04%	-0,59%	2,36%	2,26%	1,55%
		-1,46%	-0,97%	1,73%	1,69%	1,06%
200	20	-0,19%	0,22%	1,35%	1,16%	0,85%
		-0,58%	-0,21%	0,82%	0,66%	0,47%
500	20	-2,99%	-2,52%	0,96%	0,59%	-0,53%
		-3,22%	-2,77%	0,56%	0,30%	-0,73%
Médias		2,06%	2,39%	3,71%	3,58%	3,18%
		1,30%	1,58%	2,75%	2,71%	2,42%

Tabela 5.39 – Desvio relativo médio em relação ao *BM* das soluções IG com $t = 60$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_100

MÉTODOS						
n	m	SETUP	TOTAL	UBX+S	NEHT-RB	CNIT
20	5	4,74%	4,20%	5,25%	4,91%	5,02%
		3,37%	2,82%	3,03%	2,95%	3,78%
20	10	3,68%	3,63%	3,67%	3,89%	3,43%
		2,32%	2,40%	2,26%	2,76%	2,24%
20	20	1,92%	1,86%	2,38%	1,78%	2,02%
		0,97%	1,16%	1,61%	0,92%	1,09%
50	5	4,53%	5,37%	8,09%	7,69%	6,80%
		3,44%	4,32%	6,93%	5,87%	5,59%
50	10	4,87%	5,04%	5,94%	5,75%	5,86%
		3,63%	4,00%	5,01%	4,71%	5,05%
50	20	3,71%	3,98%	3,90%	4,02%	3,64%
		2,88%	3,20%	2,99%	3,48%	2,89%
100	5	1,47%	2,54%	6,20%	6,88%	6,18%
		0,84%	1,94%	5,22%	5,74%	5,36%
100	10	2,66%	2,79%	4,79%	4,61%	4,06%
		2,04%	2,03%	3,97%	3,66%	3,51%
100	20	1,96%	2,09%	2,91%	2,78%	2,31%
		1,41%	1,49%	2,26%	2,21%	1,79%
200	10	-1,51%	-0,66%	2,78%	2,82%	1,80%
		-1,84%	-1,15%	2,06%	2,17%	1,42%
200	20	-0,43%	-0,13%	1,35%	1,13%	0,80%
		-0,92%	-0,49%	0,84%	0,69%	0,36%
500	20	-3,57%	-3,25%	1,15%	0,64%	-0,84%
		-3,89%	-3,54%	0,67%	0,27%	-0,97%
Médias		2,00%	2,29%	4,04%	3,91%	3,42%
		1,19%	1,52%	3,07%	2,95%	2,68%

Tabela 5.40 – Desvio relativo médio em relação ao *BM* das soluções IG com $t = 60$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_125

Figura 5.10: Desvio relativo médio em relação ao *BM* das soluções IG com $t = 60$ considerando médias, mínimos e por classe de problemas

Percebe-se uma pequena melhoria nos resultados. Esta melhoria é mais proeminente para o método com as soluções iniciais NEHT-RB e UBX+S. Esta melhoria fica em torno de 0,5% e 0,6% em média. Para as demais soluções iniciais a melhoria é da ordem de 0,1% a 0,2% em média.

4.2.3.1. Análise dos resultados da nova meta-heurística CNIT-IG

O novo método meta-heurístico CNIT-IG, por suas semelhanças com o método original IG está também sujeito às escolhas dos parâmetros T , que define o padrão de temperatura constante baseado no método *simulated annealing*, e o parâmetro d , que define o tamanho do vetor de tarefas removidas da solução original e inseridas

novamente na etapa de construção. A nova meta-heurística difere da original principalmente no que se refere a esta etapa, já que utiliza o conceito introduzido pela nova heurística construtiva CNIT, em que, a cada etapa de inserção das tarefas, todas as tarefas ainda não programadas são testadas em todas as posições possíveis da sub-sequência atual. A sub-sequência de menor *makespan* é mantida para a próxima iteração e a tarefa escolhida é removida da lista de tarefas não programadas.

Assim, pode-se dizer que este novo método não obedece aos mesmos padrões do método IG no que se refere à etapa de destruição e construção, uma vez que todas as tarefas removidas são testadas, independentemente da ordem em que são removidas e, portanto, o valor utilizado para o parâmetro d não necessariamente obedece ao estudo realizado por Ruiz e Stützle (2008), no qual encontra o melhor valor para o parâmetro d para o problema do *flowshop* permutacional.

Aqui são avaliadas as melhores soluções iniciais apresentadas na seção anterior (CNIT para o grupo SSD_10 e SETUP para os outros grupos) com 5 diferentes valores possíveis para o parâmetro d :

- $d = 4$;
- $d = 8$;
- $d = 10\%$ do número de tarefas $(0,1\ n)$;
- $d = 20\%$ do número de tarefas $(0,2\ n)$;
- $d = 40\%$ do número de tarefas $(0,4\ n)$.

		MÉTODOS									
		Sol. Inicial SETUP					Sol. Inicial CNIT				
		d =	d =	d =	d =	d =	d =	d =	d =	d =	d =
n	m	4	8	0,1n	0,2n	0,4n	4	8	0,1n	0,2n	0,4n
20	5	1,82%	1,27%	2,50%	1,73%	0,97%	1,13%	0,95%	1,35%	1,26%	0,91%
		0,84%	0,66%	1,19%	0,78%	0,42%	0,61%	0,40%	0,83%	0,84%	0,51%
20	10	1,99%	1,38%	3,07%	1,99%	1,38%	1,63%	1,18%	2,27%	1,66%	1,25%
		0,92%	0,75%	1,60%	1,04%	0,65%	0,89%	0,56%	1,35%	0,95%	0,64%
20	20	1,52%	0,84%	2,26%	1,50%	0,88%	1,27%	0,79%	2,16%	1,32%	0,85%
		0,79%	0,37%	1,36%	0,74%	0,43%	0,70%	0,44%	1,34%	0,63%	0,43%
50	5	1,67%	1,37%	1,65%	1,43%	1,33%	1,33%	1,10%	1,24%	1,16%	1,30%
		1,15%	1,05%	1,16%	1,04%	1,03%	1,05%	0,82%	0,97%	0,91%	0,98%
50	10	2,94%	2,26%	2,71%	2,41%	2,43%	2,47%	1,99%	2,23%	2,16%	2,26%
		2,17%	1,78%	2,12%	1,71%	1,94%	1,89%	1,47%	1,82%	1,61%	1,91%
50	20	2,85%	2,05%	2,46%	2,02%	2,18%	2,40%	1,99%	2,03%	1,94%	2,10%
		2,12%	1,59%	1,94%	1,55%	1,74%	1,89%	1,56%	1,60%	1,48%	1,73%
100	5	1,08%	0,96%	0,91%	1,02%	1,17%	1,09%	0,92%	0,95%	1,03%	1,15%
		0,82%	0,72%	0,76%	0,83%	0,96%	0,86%	0,73%	0,74%	0,84%	0,97%
100	10	1,70%	1,33%	1,28%	1,23%	1,43%	1,22%	1,07%	1,02%	1,16%	1,34%
		1,28%	0,95%	0,96%	0,99%	1,20%	1,00%	0,84%	0,75%	0,95%	1,14%
100	20	2,06%	1,50%	1,58%	1,58%	1,69%	1,84%	1,28%	1,33%	1,41%	1,68%
		1,64%	1,04%	1,18%	1,25%	1,46%	1,44%	0,91%	1,06%	1,15%	1,42%
200	10	0,90%	0,70%	0,74%	0,86%	1,03%	0,62%	0,54%	0,65%	0,81%	0,99%
		0,64%	0,49%	0,58%	0,70%	0,85%	0,50%	0,36%	0,51%	0,65%	0,87%
200	20	1,63%	1,06%	0,98%	1,08%	1,24%	0,87%	0,62%	0,76%	1,00%	1,13%
		1,34%	0,76%	0,71%	0,87%	1,07%	0,65%	0,46%	0,56%	0,78%	0,94%
500	20	1,51%	1,28%	0,91%	0,83%	0,72%	0,45%	0,40%	0,49%	0,59%	0,59%
		1,27%	1,05%	0,70%	0,65%	0,53%	0,37%	0,29%	0,40%	0,43%	0,47%
Médias		1,80%	1,34%	1,75%	1,47%	1,37%	1,36%	1,07%	1,37%	1,29%	1,30%
		1,25%	0,93%	1,19%	1,01%	1,02%	0,99%	0,74%	0,99%	0,94%	1,00%

Tabela 5.41. Desvio relativo médio em relação ao *BM* das soluções CNIT-IG (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_10

MÉTODOS											
n	m	Sol. Inicial SETUP					Sol. Inicial CNIT				
		d =	d =	d =	d =	d =	d =	d =	d =	d =	d =
		4	8	0,1n	0,2n	0,4n	4	8	0,1n	0,2n	0,4n
20	5	3,42%	2,75%	4,58%	3,30%	2,77%	3,73%	2,78%	4,23%	3,55%	3,09%
		2,04%	1,55%	2,55%	2,13%	1,64%	2,56%	1,68%	2,89%	2,53%	2,12%
20	10	2,56%	2,17%	3,84%	2,70%	2,15%	2,51%	1,95%	3,08%	2,37%	1,93%
		1,41%	1,40%	2,53%	1,57%	1,22%	1,29%	1,29%	2,07%	1,53%	1,18%
20	20	1,93%	1,32%	2,71%	1,89%	1,24%	1,55%	1,29%	2,85%	1,79%	1,14%
		0,99%	0,79%	1,75%	1,11%	0,66%	1,15%	0,68%	1,97%	0,96%	0,67%
50	5	4,17%	3,83%	3,88%	3,73%	4,17%	4,87%	4,41%	4,86%	4,35%	4,32%
		3,25%	2,93%	3,19%	3,09%	3,42%	3,85%	3,57%	3,81%	3,67%	3,58%
50	10	4,11%	3,15%	3,76%	3,46%	3,68%	4,07%	3,43%	3,91%	3,59%	3,77%
		3,21%	2,40%	2,87%	2,90%	2,92%	3,31%	2,64%	3,28%	2,88%	3,22%
50	20	3,24%	2,75%	2,95%	2,53%	2,73%	2,89%	2,38%	2,69%	2,28%	2,54%
		2,31%	2,20%	2,20%	1,94%	2,27%	2,36%	1,98%	2,02%	1,71%	2,16%
100	5	2,22%	2,09%	2,05%	2,45%	2,72%	4,04%	3,62%	3,45%	3,83%	4,14%
		1,75%	1,65%	1,52%	2,07%	2,16%	3,41%	3,04%	2,85%	3,39%	3,68%
100	10	2,83%	2,40%	2,39%	2,73%	3,12%	3,18%	2,62%	2,61%	2,90%	3,16%
		2,11%	1,95%	1,99%	2,20%	2,57%	2,51%	2,19%	2,16%	2,44%	2,80%
100	20	2,63%	1,95%	1,93%	2,18%	2,41%	2,42%	1,83%	1,84%	2,17%	2,29%
		2,02%	1,47%	1,43%	1,79%	2,09%	1,84%	1,45%	1,38%	1,77%	1,98%
200	10	0,55%	0,32%	0,84%	1,28%	1,51%	1,69%	1,43%	1,79%	2,09%	2,45%
		0,08%	0,01%	0,55%	0,79%	0,97%	1,37%	1,09%	1,47%	1,79%	2,15%
200	20	0,97%	0,72%	1,17%	1,52%	1,80%	1,08%	0,85%	1,14%	1,51%	1,76%
		0,53%	0,29%	0,79%	1,27%	1,54%	0,76%	0,59%	0,84%	1,14%	1,51%
500	20	-0,92%	-0,97%	-0,55%	-0,36%	-0,39%	0,31%	0,28%	0,54%	0,57%	0,60%
		-1,13%	-1,19%	-0,81%	-0,68%	-0,64%	0,15%	0,15%	0,38%	0,40%	0,45%
Médias		2,31%	1,87%	2,46%	2,28%	2,33%	2,69%	2,24%	2,75%	2,58%	2,60%
		1,55%	1,29%	1,71%	1,68%	1,73%	2,05%	1,69%	2,09%	2,02%	2,13%

Tabela 5.42. Desvio relativo médio em relação ao *BM* das soluções CNIT-IG (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_50

		MÉTODOS									
		Sol. Inicial SETUP					Sol. Inicial CNIT				
m		d =	d =	d =	d =	d =	d =	d =	d =	d =	d =
n		4	8	0,1n	0,2n	0,4n	4	8	0,1n	0,2n	0,4n
20	5	4,35%	3,87%	5,13%	4,61%	4,26%	5,56%	4,82%	6,53%	5,79%	4,20%
		2,45%	2,22%	3,46%	2,83%	2,79%	4,20%	2,88%	5,83%	4,50%	2,44%
20	10	3,57%	2,64%	4,68%	3,69%	3,02%	3,60%	2,67%	4,68%	3,50%	2,92%
		2,08%	1,70%	2,67%	2,19%	1,98%	2,65%	1,73%	3,62%	2,47%	1,76%
20	20	2,35%	1,80%	3,64%	2,68%	1,72%	2,38%	1,64%	3,24%	2,52%	1,70%
		1,66%	0,94%	2,35%	1,78%	1,08%	1,62%	1,05%	2,53%	1,80%	1,04%
50	5	4,97%	4,74%	4,53%	4,62%	5,06%	7,35%	6,48%	7,09%	6,49%	6,79%
		3,93%	3,48%	3,32%	3,65%	4,04%	6,20%	4,84%	6,17%	5,16%	5,50%
50	10	5,41%	4,64%	4,95%	4,38%	4,81%	5,57%	4,69%	5,37%	4,93%	4,59%
		4,36%	3,49%	3,67%	3,53%	3,89%	4,56%	3,78%	4,56%	4,20%	3,45%
50	20	4,23%	3,27%	3,77%	3,35%	3,57%	4,00%	3,32%	3,96%	3,16%	3,40%
		3,38%	2,58%	2,98%	2,70%	2,92%	3,14%	2,46%	3,28%	2,51%	2,68%
100	5	2,31%	1,86%	1,87%	2,13%	2,40%	6,15%	5,18%	5,19%	5,43%	5,78%
		1,62%	1,32%	1,41%	1,48%	1,82%	5,42%	4,37%	4,13%	4,68%	4,87%
100	10	2,82%	2,36%	2,42%	2,71%	3,37%	4,11%	3,40%	3,37%	3,87%	3,89%
		2,25%	1,68%	1,86%	2,18%	2,67%	3,60%	2,68%	2,65%	3,33%	3,40%
100	20	2,82%	2,19%	2,16%	2,68%	3,12%	3,16%	2,53%	2,61%	2,91%	3,19%
		2,13%	1,68%	1,53%	2,14%	2,62%	2,57%	2,06%	2,12%	2,33%	2,75%
200	100	-0,77%	-0,96%	-0,61%	-0,13%	-0,07%	2,05%	1,51%	1,97%	2,47%	2,70%
		-1,16%	-1,45%	-1,13%	-0,63%	-0,69%	1,58%	1,08%	1,61%	2,08%	2,38%
200	20	0,29%	-0,05%	0,61%	1,00%	1,33%	1,30%	0,92%	1,36%	1,78%	1,92%
		-0,21%	-0,38%	0,27%	0,46%	0,93%	0,89%	0,56%	1,08%	1,50%	1,62%
500	20	-2,76%	-2,82%	-2,39%	-2,28%	-2,32%	-0,31%	-0,33%	-0,14%	-0,07%	-0,05%
		-3,00%	-3,03%	-2,66%	-2,70%	-2,69%	-0,46%	-0,48%	-0,29%	-0,18%	-0,15%
Médias		2,47%	1,96%	2,56%	2,45%	2,52%	3,74%	3,07%	3,77%	3,57%	3,42%
		1,62%	1,19%	1,64%	1,63%	1,78%	3,00%	2,25%	3,11%	2,86%	2,64%

Tabela 5.43. Desvio relativo médio em relação ao *BM* das soluções CNIT-IG (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_100

MÉTODOS											
n	m	Sol. Inicial SETUP					Sol. Inicial CNIT				
		d =	d =	d =	d =	d =	d =	d =	d =	d =	d =
		4	8	0,1n	0,2n	0,4n	4	8	0,1n	0,2n	0,4n
20	5	4,63%	4,58%	5,45%	4,66%	4,53%	5,67%	4,89%	5,96%	5,48%	4,63%
		2,85%	2,81%	3,26%	2,67%	3,09%	4,65%	2,93%	5,17%	4,12%	2,84%
20	10	3,86%	3,40%	4,91%	4,01%	3,04%	3,79%	3,45%	4,91%	3,68%	3,19%
		2,66%	2,17%	3,15%	2,25%	1,85%	2,66%	2,34%	3,57%	2,17%	1,98%
20	20	2,68%	1,85%	3,76%	2,76%	1,97%	2,45%	1,89%	3,53%	2,57%	1,91%
		1,33%	1,02%	2,14%	1,61%	1,21%	1,64%	1,11%	2,62%	1,64%	1,16%
50	5	4,80%	4,27%	4,68%	4,12%	4,91%	7,59%	6,72%	6,98%	6,35%	6,91%
		3,79%	2,87%	3,83%	2,95%	3,86%	6,60%	5,55%	5,90%	5,07%	5,73%
50	10	5,31%	4,56%	5,15%	4,76%	5,29%	6,07%	5,51%	6,06%	5,24%	5,44%
		4,30%	3,51%	3,91%	3,76%	4,30%	4,85%	4,69%	4,96%	4,40%	4,25%
50	20	4,35%	3,66%	3,96%	3,62%	3,91%	4,50%	3,53%	3,98%	3,51%	3,65%
		3,40%	3,00%	3,06%	2,93%	3,35%	3,58%	2,95%	3,18%	2,81%	3,08%
100	5	1,80%	1,44%	1,38%	1,76%	1,97%	6,77%	5,91%	5,66%	6,14%	6,46%
		1,11%	0,86%	0,82%	1,29%	1,35%	5,76%	5,02%	4,85%	5,11%	5,57%
100	100	2,95%	2,43%	2,43%	3,07%	3,61%	4,81%	3,74%	3,99%	4,22%	4,68%
		2,03%	1,59%	1,56%	2,30%	2,84%	4,19%	3,02%	3,42%	3,38%	4,08%
100	20	2,55%	2,06%	1,95%	2,59%	2,89%	3,09%	2,42%	2,39%	2,80%	3,30%
		1,96%	1,43%	1,56%	2,12%	2,31%	2,57%	1,81%	1,82%	2,27%	2,82%
200	10	-1,13%	-1,38%	-1,08%	-0,77%	-0,69%	2,27%	1,86%	2,38%	2,68%	3,10%
		-1,69%	-1,78%	-1,53%	-1,30%	-1,24%	1,80%	1,41%	2,03%	2,25%	2,79%
200	20	-0,01%	-0,34%	0,32%	0,83%	0,98%	1,22%	0,84%	1,39%	1,78%	1,90%
		-0,61%	-0,75%	-0,12%	0,31%	0,42%	0,92%	0,42%	1,06%	1,50%	1,62%
500	20	-3,34%	-3,48%	-2,96%	-2,96%	-2,99%	-0,63%	-0,69%	-0,50%	-0,47%	-0,49%
		-3,59%	-3,71%	-3,25%	-3,31%	-3,32%	-0,75%	-0,84%	-0,65%	-0,57%	-0,63%
Médias		2,37%	1,92%	2,50%	2,37%	2,45%	3,97%	3,34%	3,89%	3,66%	3,72%
		1,46%	1,08%	1,53%	1,47%	1,67%	3,21%	2,53%	3,16%	2,85%	2,94%

Tabela 5.44. Desvio relativo médio em relação ao *BM* das soluções CNIT-IG (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_125

Figura 5.11: Desvio relativo médio em relação ao *BM* das soluções CNIT-IG considerando as médias, mínimos e por classe de problemas

Algumas soluções se equivalem, como nos casos com $d = 4$ ou $d = 0,2 n$ para $n = 20$ e $d = 8$ ou $d = 0,4 n$ para $n = 20$. Ainda assim os experimentos foram conduzidos independentemente e apresentados separados. A relativa semelhança entre os resultados encontrados para esta classe de problemas demonstra esta equivalência.

A partir da tabela para o grupo SSD_10, pode-se observar o domínio da solução inicial CNIT com parâmetro $d = 8$, enquanto das tabelas para os grupos SSD_50, SSD_100 e SSD_125 observa-se o domínio da solução inicial SETUP com o parâmetro $d = 8$.

Com base nestas observações pode-se comparar os melhores resultados obtidos com a heurística IG (obtidos com as soluções iniciais SETUP e CNIT) com os resultados obtidos com a heurística CNIT-IG com as soluções iniciais SETUP e CNIT, utilizando, para este caso, o parâmetro $d = 8$.

Os métodos serão identificados na tabela da seguinte maneira:

- IG: método original de Ruiz e Stützle (2008), com solução inicial NEHT-RB.
- IG SETUP: método IG com solução inicial SETUP
- IG CNIT: método IG com solução inicial CNIT
- CNIT-IG SETUP: método CNIT-IG com solução inicial SETUP
- CNIT-IG CNIT: método CNIT-IG com solução inicial CNIT

MÉTODOS						
n	m	IG	IG SETUP	IG CNIT	CNIT-IG SETUP	CNIT-IG CNIT
20	5	1,31%	1,56%	1,19%	1,27%	0,95%
		0,74%	0,83%	0,75%	0,66%	0,40%
20	10	1,54%	1,84%	1,55%	1,38%	1,18%
		0,93%	0,97%	0,98%	0,75%	0,56%
20	20	1,24%	1,59%	1,22%	0,84%	0,79%
		0,78%	0,65%	0,49%	0,37%	0,44%
50	5	1,36%	1,58%	1,34%	1,37%	1,10%
		1,07%	1,08%	1,09%	1,05%	0,82%
50	10	2,52%	3,00%	2,54%	2,26%	1,99%
		1,96%	2,07%	1,95%	1,78%	1,47%
50	20	2,55%	2,75%	2,33%	2,05%	1,99%
		1,92%	2,20%	1,83%	1,59%	1,56%
100	5	1,10%	1,09%	1,07%	0,96%	0,92%
		0,88%	0,84%	0,82%	0,72%	0,73%
100	10	1,23%	1,64%	1,29%	1,33%	1,07%
		1,00%	1,25%	1,02%	0,95%	0,84%
100	20	1,60%	2,12%	1,73%	1,50%	1,28%
		1,28%	1,62%	1,30%	1,04%	0,91%
200	10	0,66%	0,87%	0,65%	0,70%	0,54%
		0,48%	0,65%	0,49%	0,49%	0,36%
200	20	0,79%	1,49%	0,79%	1,06%	0,62%
		0,59%	1,18%	0,59%	0,76%	0,46%
500	20	0,38%	1,33%	0,36%	1,28%	0,40%
		0,26%	1,07%	0,24%	1,05%	0,29%
Médias		1,36%	1,74%	1,34%	1,34%	1,07%
		0,99%	1,20%	0,96%	0,93%	0,74%

Tabela 5.45. Desvio relativo médio em relação ao *BM* das soluções selecionadas IG e CNIT-IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_10

MÉTODOS						
n	m	IG	IG SETUP	IG CNIT	CNIT-IG SETUP	CNIT-IG CNIT
20	5	3,15%	3,80%	3,50%	2,75%	2,78%
		1,74%	2,05%	2,32%	1,55%	1,68%
20	10	2,51%	3,03%	2,50%	2,17%	1,95%
		1,26%	2,10%	1,57%	1,40%	1,29%
20	20	1,64%	1,76%	1,58%	1,32%	1,29%
		1,07%	0,96%	0,80%	0,79%	0,68%
50	5	4,71%	4,11%	4,86%	3,83%	4,41%
		3,74%	3,34%	3,71%	2,93%	3,57%
50	10	4,01%	4,08%	4,05%	3,15%	3,43%
		3,16%	3,34%	3,46%	2,40%	2,64%
50	20	3,15%	3,03%	2,61%	2,75%	2,38%
		2,73%	2,30%	1,95%	2,20%	1,98%
100	5	4,17%	2,16%	4,12%	2,09%	3,62%
		3,57%	1,61%	3,61%	1,65%	3,04%
100	10	3,05%	2,77%	3,10%	2,40%	2,62%
		2,54%	2,13%	2,51%	1,95%	2,19%
100	20	2,34%	2,55%	2,31%	1,95%	1,83%
		1,83%	2,03%	1,85%	1,47%	1,45%
200	10	1,80%	0,48%	1,66%	0,32%	1,43%
		1,40%	0,08%	1,29%	0,01%	1,09%
200	20	1,12%	0,86%	0,95%	0,72%	0,85%
		0,76%	0,47%	0,63%	0,29%	0,59%
500	20	0,66%	-1,10%	0,25%	-0,97%	0,28%
		0,50%	-1,30%	0,09%	-1,19%	0,15%
Médias		2,69%	2,29%	2,62%	1,87%	2,24%
		2,02%	1,59%	1,98%	1,29%	1,69%

Tabela 5.46. Desvio relativo médio em relação ao *BM* das soluções seleccionadas IG e CNIT-IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_50

MÉTODOS						
n	m	IG	IG SETUP	IG CNIT	CNIT-IG SETUP	CNIT-IG CNIT
20	5	6,06%	5,09%	5,15%	3,87%	4,82%
		3,98%	3,28%	3,48%	2,22%	2,88%
20	10	4,07%	3,35%	3,83%	2,64%	2,67%
		2,62%	2,14%	2,51%	1,70%	1,73%
20	20	2,28%	2,30%	2,35%	1,80%	1,64%
		1,49%	1,26%	1,38%	0,94%	1,05%
50	5	7,49%	5,23%	7,16%	4,74%	6,48%
		6,23%	4,35%	6,03%	3,48%	4,84%
50	10	6,00%	5,08%	5,54%	4,64%	4,69%
		4,93%	4,26%	4,79%	3,49%	3,78%
50	20	3,86%	3,90%	3,98%	3,27%	3,32%
		3,20%	3,24%	3,10%	2,58%	2,46%
100	5	6,60%	2,19%	5,92%	1,86%	5,18%
		5,66%	1,43%	5,21%	1,32%	4,37%
100	10	4,65%	2,80%	3,93%	2,36%	3,40%
		3,89%	2,16%	3,40%	1,68%	2,68%
100	20	3,04%	2,66%	2,90%	2,19%	2,53%
		2,59%	2,03%	2,42%	1,68%	2,06%
200	10	2,88%	-0,85%	1,93%	-0,96%	1,51%
		2,28%	-1,33%	1,50%	-1,45%	1,08%
200	20	1,64%	0,14%	1,16%	-0,05%	0,92%
		1,23%	-0,13%	0,78%	-0,38%	0,56%
500	20	0,95%	-2,88%	-0,37%	-2,82%	-0,33%
		0,64%	-3,12%	-0,52%	-3,03%	-0,48%
Médias		4,13%	2,42%	3,62%	1,96%	3,07%
		3,23%	1,63%	2,84%	1,19%	2,25%

Tabela 5.47. Desvio relativo médio em relação ao *BM* das soluções selecionadas IG e CNIT-IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_100

MÉTODOS						
n	m	IG	IG SETUP	IG CNIT	CNIT-IG SETUP	CNIT-IG CNIT
20	5	5,35%	5,02%	5,43%	4,58%	4,89%
		3,90%	3,74%	3,83%	2,81%	2,93%
20	10	4,43%	3,68%	3,86%	3,40%	3,45%
		3,24%	2,06%	2,72%	2,17%	2,34%
20	20	2,20%	2,41%	2,42%	1,85%	1,89%
		1,24%	1,28%	1,49%	1,02%	1,11%
50	5	8,58%	4,87%	7,35%	4,27%	6,72%
		7,01%	3,99%	5,88%	2,87%	5,55%
50	10	6,38%	5,18%	6,16%	4,56%	5,51%
		5,25%	4,28%	5,02%	3,51%	4,69%
50	20	4,23%	4,20%	3,94%	3,66%	3,53%
		3,46%	3,35%	3,23%	3,00%	2,95%
100	5	7,74%	1,69%	6,40%	1,44%	5,91%
		6,72%	1,11%	5,65%	0,86%	5,02%
100	10	5,14%	2,95%	4,54%	2,43%	3,74%
		4,38%	2,06%	3,74%	1,59%	3,02%
100	20	3,09%	2,44%	2,89%	2,06%	2,42%
		2,49%	1,77%	2,40%	1,43%	1,81%
200	10	3,19%	-1,35%	2,11%	-1,38%	1,86%
		2,66%	-1,79%	1,56%	-1,78%	1,41%
200	20	1,59%	-0,18%	1,03%	-0,34%	0,84%
		0,95%	-0,78%	0,64%	-0,75%	0,42%
500	20	1,19%	-3,47%	-0,71%	-3,48%	-0,69%
		0,84%	-3,80%	-0,83%	-3,71%	-0,84%
	Médias	4,43%	2,29%	3,79%	1,92%	3,34%
		3,51%	1,44%	2,94%	1,08%	2,53%

Tabela 5.48. Desvio relativo médio em relação ao *BM* das soluções seleccionadas IG e CNIT-IG com $t = 30$ (média na primeira linha, em negrito e mínimo na segunda) para o grupo SSD_125

Figura 5.12: Desvio relativo médio em relação ao *BM* das soluções selecionadas IG e CNIT-IG com $t = 30$, considerando as médias, mínimos e por classe de problemas

Com base nas tabelas acima, é possível concluir que o método original só não possui o pior desempenho geral para o grupo SSD_10. Ainda assim, o equilíbrio apresentado pelas diferentes soluções iniciais com o método IG é superado pela nova meta-heurística CNIT-IG para os grupos SSD_50, SSD_100 e SSD_125.

5. Conclusões

Este trabalho estudou o problema do *flowshop* com tempos de *setup* separados e dependentes da sequência (SDST flowshop). Este problema possui diversas aplicações nos ambientes de fabricação e prestação de serviço.

A preocupação deste trabalho foi utilizar os melhores métodos encontrados na literatura em termos de métodos heurísticos construtivos e meta-heurísticas e desenvolver uma solução de boa qualidade para o problema *SDST flowshop*. Dentre as propriedades avaliadas, a utilização direta da propriedade UBX como proposta por Moccellin e Nagano (2007) não se mostrou diretamente efetiva, porém permitiu o desenvolvimento dos parâmetros *XR* e *QR* apresentados no capítulo 3.

A partir destas propriedades, foram desenvolvidas dois métodos: um heurístico denominado CNIT e outro meta-heurístico denominado CNIT-IG.

Na comparação com os métodos heurísticos construtivos da literatura, o método proposto CNIT apresentou porcentagem de sucesso de 48,75% contra 34,75% do segundo colocado (a heurística NEHT-RB). Seu desempenho foi tal que, para os exemplos com 500 tarefas dos grupos SSD_100 e SSD_125, o método foi capaz de encontrar novos limites inferiores sem quaisquer etapas melhorativas (identificadas pelos valores próximos de zero ou negativos dos desvios relativos médios). Em contrapartida, sua complexidade em termos computacionais é de ordem mn^3 , enquanto as demais heurísticas avaliadas possuem complexidade mn^2 ; assim, os problemas menores possuem tempo computacional médio duas vezes maiores que o segundo maior, enquanto para os problemas maiores o tempo computacional é 35 vezes maior que o segundo maior e cerca de 100 vezes maior que o menor.

Em seguida os métodos heurísticos construtivos heurísticos foram submetidos a uma busca local descendente. A busca local mostrou-se bastante efetiva para os métodos baseados nas características do problema SDST flowshop (SETUP, TOTAL e UBX+S), alcançando melhoria máxima dos desvios relativos médios em relação ao BM de 16,58% e 8,04% em média; enquanto para os métodos que já utilizam inserção em algum momento (NEHT-RB e CNIT), a melhoria máxima foi de 6,01% e a média foi de

3,03%. O método mais beneficiado com a busca local foi o UBX+S, com melhoria média de 12,67%, porém sua porcentagem média de sucesso permaneceu praticamente a mesma, mostrando a potência do método de busca local ainda assim limitada à qualidade da solução inicial. O tempo computacional exigido para os métodos SETUP, TOTAL e UBX+S foi aproximadamente 7 vezes o tempo médio computacional para a solução inicial, cerca de 11 vezes o para o NEHT-RB e no máximo 90% superior para o método CNIT. Isto se deve ao alto custo computacional para a solução inicial e também a sua boa qualidade na geração da solução inicial

A meta-heurística IG foi avaliada em termos de sua solução inicial e os resultados indicaram que o novo método construtivo CNIT fornece soluções melhores para o grupo de exemplos SSD_10, obtendo 41,67% de porcentagem média de sucesso, contra 40% do método original com solução inicial através do NEHT-RB; enquanto o método SETUP fornece soluções melhores para os demais grupos de exemplos, obtendo média de 51,94% de porcentagem média de sucesso contra 19,17% da solução que utiliza TOTAL como solução inicial e 14,44% da solução que utiliza CNIT. O método IG alcançou uma melhoria máxima de 17,6% em relação à solução inicial (pelo método TOTAL no grupo SSD_10). O método IG com solução inicial SETUP obteve melhoria média de 8,05% e o método CNIT obteve melhoria média de 2,83%.

Dobrar o limite de tempo computacional não afetou drasticamente o desempenho com relação à solução inicial: a porcentagem média de sucesso manteve-se estável entre as diferentes soluções iniciais. A melhoria no desvio relativo médio com o aumento no limite de tempo computacional foi de 0,52% em média, sendo 0,64% em média para a solução inicial SETUP e 0,49% para a solução inicial CNIT.

A nova meta-heurística CNIT-IG fornece resultados ainda melhores que os obtidos pela heurística IG. A comparação entre as meta-heurísticas IG e CNIT-IG com as duas diferentes soluções iniciais mostra que o desvio relativo médio obtido com a meta-heurística CNIT-IG e a solução inicial CNIT é 0,38% em média melhor que a solução obtida pelo método IG original para o grupo de exemplos SSD_10; para os demais grupos, a solução obtida por CNIT-IG com a solução inicial SETUP é, em média, 1,83% melhor que a solução obtida com o método IG original.

O desempenho obtido pela busca local em soluções iniciais baseadas nas propriedades do problema SDST flowshop sugere trabalhos futuros com o objetivo de desenvolver buscas locais baseadas nos parâmetros XR e QR , como o método 2 opt, por exemplo, para aplicação em soluções iniciais baseadas em métodos como o NEHT-RB ou CNIT; ou ainda a utilização dos parâmetros em combinação com o método UBX+S: a cada par de tarefas programadas os parâmetros podem ser utilizados para corrigir os custos da analogia ATSP. Por fim, o método CNIT-IG pode ser utilizado com diferentes medidas de desempenho, como tempo de fluxo médio por exemplo.

REFERÊNCIAS

ALLAHVERDI, A.; AL-ANZI, F. (2002). Using two-machine flowshop with maximum lateness objective to model multimedia data objects scheduling problem for WWW applications. **Computers and Operations Research** 29, 971-994

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**, Oxford, vol. 52, n. 4, p. 449-462.

ALLAHVERDI, A.; GUPTA, J. N. D.; ALDOWAISAN, T. (1999). **A review of scheduling research involving setup considerations**. Omega, Int. J. Mgmt Sci. V. 27 219-239.

ALLAHVERDI, A.; GUPTA, J. N. D.; ALDOWAISAN, T. (2008). A survey of scheduling problems with setup times or costs. **European Journal of Operational Research**. V. 187 985-1032.

ALLAHVERDI, A.; SOROUSH, H. M. (2008). The significance of reducing setup times/setup costs. **European Journal of Operational Research**, Amsterdam, vol. 187, n. 3, p. 978-984.

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

CAMPBELL, H.G.; DUDEK, R.A.; SMITH, M.L. (1970) A heuristic algorithm for the n job m machine sequencing problem. **Management Science**. Rhode Island, v. 16, n. 10, p. B630-B637, 1970.

CHANG, P. C.; HSIEH, J. C.; WANG, Y. W. (2003). Genetic algorithms applied in BOPP film scheduling problems: Minimizing total absolute deviation and setup times. **Applied Soft Computing** 3, 139-148.

CHENG, T. C. E.; GUPTA, J. N. D.; WANG G. (2000). A Review of Flowshop Scheduling Research With Setup Times. **Production and Operations Management** Vol. 9 no. 3 Fall 2000. 262-282.

CORWIN, B. D.; ESOGBUE, A. O. (1974). Two Machine Flow Shop Scheduling Problems with Sequence Dependent Setup Times: A Dynamic Programming Approach. **Naval Research Logistics Quarterly** V. 21 515-524.

CROWDER, B. **Minimizing the makespan in a flexible flowshop with sequence dependent setup times, uniform machines, and limited buffers.** 2006. 136f. Dissertação (Master of Science in Industrial Engineering) - Department of Industrial and Management Systems Engineering, West Virginia University, Morgantown, 2006.

DAS, S. R.; GUPTA, J. N. D.; KHUMAWALA, B. M. (1995). A Savings Index Heuristic Algorithm for Flowshop Scheduling with Sequence Dependent Set-up Times. **Journal of the Operational Research Society** V. 46 1365-1373.

DAUZÈRE-PÉRÈS, S.; LASSERE, J. (2002). On the Importance of sequencing decisions in production planning and scheduling. **International Transactions in Operational Research** v. 9, p. 779-793.

DONG, X.; HUANG, H.; CHEN, P. (2009). Study on Heuristics for the Permutation Flowshop with Sequence Dependent Setup Times. **IEEE IRI** V. 10 417-421.

DUDEK, R. A.; PANWALKAR, S. S.; SMITH, M. L. (1992). The lessons of flowshop scheduling research. **Operations Research**, Hannover, vol. 40, n. 1, p. 7-13.

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

GAITHER, N.; FRAZIER, G. (2005). **Administração da produção e operações.** 8ª Ed. São Paulo: Pioneira Thomsom Learning.

GAJPAL, Y.; RAJENDRAN, C.; ZIEGLER, H. (2005). An ant colony algorithm for scheduling in flowshops with sequence-dependent setup times of jobs. **Int. J. Manuf Technol** 30, 416-424.

GAREY, M. R.; JOHNSON, D. S. **Computers and intractability**: a guide to the theory of NP-completeness. W. H Freeman and Company, 1979.

GENDREAU, M.; LAPORTE, G.; GUIMARÃES, E. M. (2001). A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. **European Journal of Operational Research** 133, 183-189.

GUPTA, J. N. D. (1975). A search Algorithm for the Generalized Flowshop Scheduling Problem. **Comput. & Ops. Res.** V. 2 83-90.

GUPTA, J. N. D. (1986). Flowshop schedules with sequence dependent setup times. **Journal of the Operational Research Society of Japan**, Tokyo, vol. 29, n. 3, p. 206–219.

GUPTA, J. N. D.; DARROW, W. P. (1986). The two-machine sequence dependent flowshop scheduling problem. **European Journal of Operational Research** V. 24 439-446.

GUPTA, J. N. D.; DAS, S. R.; GHOSH, S. **Flowshop Scheduling with Sequence Dependent Setup Times**. Muncie: Department of Management - Ball State University, 1995. (Working paper, IN 47306).

GUPTA, J. N.D.; STAFFORD JR, E. F. (2006). Flowshop scheduling research after five decades. **European Journal of Operational Research** 169 (2006) 699–711.

HEJAZI, S. R.; SAGHAFIAN, S. (2005). Flowshop-scheduling problems with makespan criterion: a review. **International Journal of Production Research** V. 43, n. 14 2895-2929.

JOHNSON, L. A.; MONTGOMERY, D. C. (1974). **Operations Research in Production Planning: Scheduling and Inventory Control**. New York. Wiley

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

KIM, D. W.; KIM, K. H.; JANG, W.; CHEN, F. F. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. **Robotics and Computer-Integrated Manufacturing** 18, 223-231

KIM, Y. D.; LIM, H. G.; PARK, M. W. (1996). Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process. **European Journal of Operational Research** V. 91 124-143.

LAGUNA, M. (1999). A heuristic for production scheduling and inventory control in the presence of sequence-dependent setup times. **IIIE Transactions** 31, 125-134.

LIN, H. T.; LIAO, C. J. (2003). A case study in a two-stage hybrid flow shop with setup time and dedicated machines. **International Journal of Production Economics** 86, 133-143.

MADDUX III, H. S.; GUPTA, J. N. D. (2003). Scheduling intermediate and finished products in a two-stage flowshop with sequence dependent setup times. In: ANNUAL MEETING OF THE DECISION SCIENCES INSTITUTE, 34., 2003, Washington DC. **Proceedings...** Washington DC: The Decision Sciences Institute, 2003, p. 1579–1585.

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

MOCCELLIN, J. V.; NAGANO, M. S. Flow shop com maquinas paralelas genericas. In: SIMPOSIO BRASILEIRO DE PESQUISA OPERACIONAL, 35., 2003, Natal. **Anais...** Riode Janeiro: Sociedade Brasileira de Pesquisa Operacional, 2003, 1 CD-ROM.

MOCCELLIN, J. V.; NAGANO, M. S. (2007). Uma Propriedade Estrutural do Problema de Programação da Produção Flow Shop Permutacional com Tempos de Setup. **Pesquisa Operacional** V. 27, n. 3 487-515.

NAGANO, M. S. **Um novo método heurístico construtivo de alto desempenho para a programação de operações flow-shop permutacional**. 1999. 165 f. Tese (Doutorado em Engenharia de Produção) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos/SP, 1999.

NAWAZ, M.; ENSCORE JR, E. E.; HAM, I. (1983). A Heuristic Algorithm for the m-Machine, n-Job Flow-shop Sequencing Problem. **OMEGA The Int. J. of Mgmt Sci.** V. 11, N 1 91-95.

NORMAN, B. A. (1999). Scheduling flowshops with finite buffers and sequence dependent setup times. **Computers and Industrial Engineering**. 36: 163-177.

OSMAN, I. H.; POTTS, C. N.; (1989). Simulated Annealing for Permutation Flow-Shop Scheduling. **OMEGA Int. J. of Mgmt. Sci.** V. 17. n. 6 551-557.

PARTHASARATHY, S.; RAJENDRAN, C. (1997a). A simulated annealing heuristic for scheduling to minimize weighted tardiness in a flowshop with sequence dependent setup times of jobs - a case study. **Production Planning & Control**, Abington, vol. 8, n. 5, p. 475-483.

PARTHASARATHY, S.; RAJENDRAN, C. (1997b). An experimental evaluation of heuristics for scheduling in a real-life flowshop with sequence-dependent setup times of jobs. **International Journal of Production Economics** V. 49 255-263.

PINEDO, MICHAEL L. (2008). **Scheduling Theory, Algorithms, and Systems**. 3^a Ed. Springer Science+Business Media, LLC.

RAJENDRAN, C.; ZIEGLER, H. (1997a). Heuristics for scheduling in a flowshop with setup, processing and removal times separated. **Production Planning & Control**, Abington, vol. 8, n. 6, p. 568-576.

RAJENDRAN, C.; ZIEGLER, H. (1997b). A heuristic for scheduling to minimize the sum of wighted flowtime of jobs in a flowshop with sequence-dependent setup times of jobs. **Computers and Industrial Engineering** 33, 281-284.

RAJENDRAN, C.; ZIEGLER, H. (2003). Scheduling to minimize the sum of weighted flowtime and weighted tardiness of jobs in a flowshop with sequence-dependent setup times. **European Journal of Operational Research**, Amsterdam, vol. 149, n. 3, p. 513–522.

REEVES, C. R. (1995) A genetic algorithm for flowshop sequencing. **Computers & Opns Res.** V. 22 5-13.

RICKEL, J. An intelligent tutoring framework for task-oriented domains. In: CONFERENCE ON INTELLIGENT TUTORING SYSTEMS, 1988, Montreal. **Proceedings...** Montreal: Lecture notes in computer science, 1988, p. 109-115.

RÍOS-MERCADO, R. Z.; BARD, J .F. **The flowshop scheduling polyhedron with setup times**. Austin: University of Texas at Austin, 1996. (Technical Report ORP96-07)

RÍOS-MERCADO, R. Z.; BARD, J. (1998a) Computational experience with a branch-and-cut algorithm for flowshop scheduling with setups. **Computers & Operations Research**, Oxford, vol. 25, n. 5, p. 351–366.

RÍOS-MERCADO, R. Z.; BARD, J. F. (1998b). Heuristics for the flow line problem with setup costs. **European Journal of Operational Research** V. 110 76-98.

RÍOS-MERCADO, R. Z.; BARD, J. F. (1999a). A branch-and-bound algorithm for permutation flow shops with sequence-dependent setup times. **IIE Transactions** V. 31, 721-73 1

RÍOS-MERCADO, R. Z.; BARD, J. F. (1999b). An Enhanced TSP-Based Heuristic for Makespan Minimization in a Flow Shop with Setup Times. **Journal of Heuristics** V. 5 53-70.

RÍOS-MERCADO, R. Z.; BARD, J. F. (2003). The Flow Shop Scheduling Polyhedron with Setup Times. **Journal of Combinatorial Optimization** V. 7, 291–318.

RUIZ, R.; MAROTO, C.; ALCARAZ, J. (2005) Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. **European Journal of Operational Research** V. 165 34-54.

RUIZ, R.; STÜTZLE, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. **European Journal of Operational Research** V. 177 2033-2049.

RUIZ, R.; STÜTZLE, T. (2008). An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. **European Journal of Operational Research** V. 187 1143-1159.

SCHALLER, J.; GUPTA, J. N. D.; VAKHARIA, A. J. (2000). Scheduling a flowline manufacturing cell with sequence dependent family setup times. **European Journal of Operational Research** 125, 324-339.

SIMMONS, J. V. (1992). Heuristics in Flow Shop Scheduling with Sequence Dependent Setup Times. **OMEGA Int. J. of Mgmt. Sci.** V. 20, n. 2 215-225.

SETHANAN, K. **Scheduling Flexible Flowshops with Sequence Dependent Setup Times**. 2001. 179f. Tese (Doctor of Philosophy in Decision Sciences and Production Systems) - Department of Industrial and Management Systems Engineering, West Virginia University, Morgantown, 2001.

SLACK, N. *et al* (1999). **Administração da Produção**. 1ª ed. 2. tir. São Paulo: Atlas. ed. compacta.

SOUZA, A.B.D.; MOCCELLIN, J.V. (2000). Metaheurística híbrida Algoritmo Genético Busca-tabu para a programação de operações *flow shop*. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, XXII, 2000, Viçosa. **Anais...** Rio de Janeiro: SOBRAPO, 2000. 1 CD-ROM

SRIKAR, B. R.; GHOSH, S. (1986). A MILP model for the n-job, m-stage flowshop with sequence dependent set-up times. **International Journal of Production Research** V.24, n.6 1459-1474.

STAFFORD, E. F.; TSENG, F. T. (1990). On the Srikar-Ghosh MILP model for the N X M SDST flowshop problem. **Int. J. Prod. Res.** V. 28, n. 10 1817-1830.

STAFFORD, E. F.; TSENG, F. T. (2001). Two models for a family of flowshop sequencing problems. **European Journal of Operational Research** V. 142 282-293

STAFFORD, E. F.; TSENG, F. T. (2002). Two MILP models for the N X M SDST Flowshop sequencing problem. **Int. J. Prod. Res.** V. 39, n. 8 1777-1809.

STINSON, J. P.; SMITH, A. W. (1982). A heuristic programming procedure for sequencing the static flowshop. **Int. J. Prod. Res.** V. 20, n. 6 753-764.

SZWARC, W.; GUPTA, J. N. D. (1987). A flow-shop problem with sequence-dependent additive setup times. **Naval Research Logistics**. v. 34, p. 619-627, 1987.

TAILLARD, E. (1990). Some efficient heuristic methods for the flow shop sequencing problem. **European Journal of Operational Research** V. 47, n. 1 65-74.

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

TSENG, F. T.; GUPTA, J. N. D.; STAFFORD, E. F. (2006). A penalty-based heuristic algorithm for the permutation flowshop scheduling problem with sequence-dependent set up times. **Journal of the Operational Research Society**, Oxford, vol. 57, n. 5, p. 541– 551.

USKUP, E.; SMITH, S. B. (1975). A branch-and-bound algorithm for two-stage production-sequencing problems. **Operations Research** V.23, n.1 118-136.

WIDMER, M.; HERTZ, A. (1989). A new heuristic method for the flow shop sequencing problem. **European Journal of Operational Research** V. 41 186-193.

YANG, W.; LIAO, C. (1999). Survey of scheduling research involving setup times. **International Journal of Systems Science**, v. 30, n. 2, p. 143-155.

YI, Y.; WANG, D. W. (2003). Soft computing for scheduling with batch setup times and earliness-tardiness penalties on parallel machines. **Journal of Intelligent Manufacturing** 14, 311-322

ZANDIEH, M.; FATHEMI GHOMI, S. M. T., MOATTAR HUSSEINI, S. M. (2006). An immune algorithm to hybrid flow shops scheduling with sequence-dependent setup times. **Applied Mathematics and Computation** 180, 111-127.

APÊNDICE - Código Fonte do programa computacional

PROGRAMA CNIT-IG.DEV

```

/*****

/*****

/*                                */

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

#include <time.h>

#include <math.h>

#undef max

#define max(a, b) ( a >= b ? a : b )

#undef min

#define min(a, b) ( a <= b ? a : b )

void leparametros(char *nome, int *maquinas, int *tarefas);

void leproblema(char *nome, int **proc, int ***setup, int m, int n);

void calculo_residuo(int **p, int ***s, int ***r, int ***UBX, int m, int n);

double GetTime();

int CNIT(int **p, int ***s, int ***r, int m, int n, int *ordem, int *tarefa, int d);

int buscalocal(int **p, int ***s, int ***r, int m, int n, int *ordem);

int insere_tarefa(int **p, int ***s, int ***r, int m, int n, int *ordem, int z, int h);

```



```
int repete = -1, salva_seq = -1;
```

```
/****** Parâmetros iniciais *****/
```

```
while
((MetodoTipo!=0)&&(MetodoTipo!=1)&&(MetodoTipo!=2)&&(MetodoTipo!=3)&&(
MetodoTipo!=4)){
```

```
    printf("Informe o método utilizado:\n");
```

```
    printf("TOTAL:  [0]\n");
```

```
    printf("SETUP:  [1]\n");
```

```
    printf("UBX+S:  [2]\n");
```

```
    printf("NEHT-RB: [3]\n");
```

```
    printf("CNIT:   [4]\n");
```

```
    scanf("%d", &MetodoTipo);
```

```
}
```

```
while ((b_local!=0)&&(b_local!=1)){
```

```
    printf("Informe se haverá busca local:\n");
```

```
    printf("NAO:  [0]\n");
```

```
    printf("SIM:  [1]\n");
```

```
    scanf("%d", &b_local);
```

```
}
```

```
while ((m_IG!=0)&&(m_IG!=1)&&(m_IG!=2)){
```

```

printf("Informe se haverá IG, CNIT-IG ou nenhum:\n");

printf("Nenhum:  [0]\n");

printf("IG:      [1]\n");

printf("CNIT-IG: [2]\n");

scanf("%d", &m_IG);

}

if ( m_IG > 0 ){

    printf("Valor do parâmetro de temperatura constante (T) :\n");

    scanf("%f", &T);

    printf("Valor do parâmetro destruição (d) :\n");

    printf("OBS: se inserido valor maior que 9 será considerado porcentagem\n");

    scanf("%d", &d_base);

    printf("Valor do parâmetro de limitação de tempo (t) :\n");

    scanf("%d", &t);

}

while
((ClasseInicial!=10)&&(ClasseInicial!=50)&&(ClasseInicial!=100)&&(ClasseInicial!=
125)){

    printf("Informe a classe inicial [10, 50, 100, 125]: ");

    scanf("%d", &ClasseInicial);

}

```

```

if ( ClasseInicial == 10 ) inicio = 0;

if ( ClasseInicial == 50 ) inicio = 1;

if ( ClasseInicial == 100 ) inicio = 2;

if ( ClasseInicial == 125 ) inicio = 3;

while
(ClasseFinal!=10&&ClasseFinal!=50&&ClasseFinal!=100&&ClasseFinal!=125||Classe
Final<ClasseInicial){

    printf("Informe a classe final [10, 50, 100, 125]: ");

    scanf("%d", &ClasseFinal);

}

if ( ClasseFinal == 10 ) fim = 0;

if ( ClasseFinal == 50 ) fim = 1;

if ( ClasseFinal == 100 ) fim = 2;

if ( ClasseFinal == 125 ) fim = 3;

while (ProblemaInicial<1||ProblemaInicial>120){

    printf("Informe o problema inicial [1 a 120]: ");

    scanf("%d", &ProblemaInicial);

}

while (ProblemaFinal<1||ProblemaFinal>120||ProblemaFinal<ProblemaInicial){

    printf("Informe o problema final [1 a 120]: ");

    scanf("%d", &ProblemaFinal);

```

```

}

while ( repete < 1 ){

    printf("Informe o número de repeticoes do experimento ");

    scanf("%d", &repete);

}

while ((salva_seq!=0)&&(salva_seq!=1)){

    printf("Informe se deseja salvar a sequencia final de cada problema:\n");

    printf("NAO:  [0]\n");

    printf("SIM:  [1]\n");

    scanf("%d", &salva_seq);

}

/*****
****/

for (int a = 0; a < repete; a++){

    for (int w = inicio; w <= fim; w++){

        if ( w == 0 ) classe = 10;

        if ( w == 1 ) classe = 50;

        if ( w == 2 ) classe = 100;

        if ( w == 3 ) classe = 125;

```



```

for (int z = ProblemaInicial; z <= ProblemaFinal; z++){

    char problema[80];

    int m, n, M, M_final, d;

    int **p, ***s, ***r, ***UBX, *ordem, *final, *atual, *tarefa, **matriz;

    double t1 = 0, t2 = 0;

    if ( z <= 9 ) sprintf(problema, "c:\\Users\\casa\\Documents\\Programação
C\\Problemas\\SDST%d_ta00%d.txt", classe, z);

        else      if      (      z      <=      99      )      sprintf(problema,
"c:\\Users\\casa\\Documents\\Programação C\\Problemas\\SDST%d_ta0%d.txt", classe,
z);

        else      sprintf(problema,      "c:\\Users\\casa\\Documents\\Programação
C\\Problemas\\SDST%d_ta%d.txt", classe, z);

    printf("Problema: SSD%d_%d\\n", classe, z);


    leparametros(problema, &m, &n);


    p = (int **) calloc (m, sizeof(int *));

    s = (int ***) calloc (m, sizeof(int **));

    r = (int ***) calloc (m, sizeof(int **));

    UBX = (int ***) calloc (m, sizeof(int **));

    for (int i = 0; i < m; i++){

        p[i] = (int *) calloc (n, sizeof(int));

```

```

s[i] = (int **) calloc (n, sizeof(int *));

r[i] = (int **) calloc (n, sizeof(int *));

UBX[i] = (int **) calloc (n, sizeof(int *));

for (int j = 0; j < n; j++){

    s[i][j] = (int *) calloc (n, sizeof(int));

    r[i][j] = (int *) calloc (n, sizeof(int));

    UBX[i][j] = (int *) calloc (n, sizeof(int));

}

}

ordem = (int *) calloc (n, sizeof(int));

final = (int *) calloc (n, sizeof(int));

atual = (int *) calloc (n, sizeof(int));

tarefa = (int *) calloc (n, sizeof(int));


leproblema(problema, p, s, m, n);

matriz = (int **) calloc (n+1, sizeof(int *));

for (int j = 0; j < n+1; j++)

    matriz[j] = (int *) calloc (n+1, sizeof(int));


t1 = GetTime();

```

```

if ( MetodoTipo < 3 ){

    if ( MetodoTipo == 0 ) TOTAL(matriz, p, s, m, n);

    if ( MetodoTipo == 1 ) SETUP(matriz, s, m, n);

    if ( MetodoTipo == 2 ){

        calculo_residuo(p, s, r, UBX, m, n);

        for (int j = 0; j < n; j++){

            for (int k = 0; k < n; k++)

                matriz[j][k] = max(UBX[m-1][j][k],0) + s[m-1][j][k];

            matriz[j][j] = -1;

        }

        VAM(matriz, ordem, n);

        M = Makespan(ordem, p, s, m, n);

        for (int j = 0; j < n; j++) atual[j] = ordem[j];

        for (int k = 0; k < n-1; k++){

            int C, aux;

            aux = atual[0];

            for (int j = 1; j < n; j++) atual[j-1] = atual[j];

            atual[n-1] = aux;

            C = Makespan(atual, p, s, m, n);

```

```

        if ( C < M){

            M = C;

            for (int j = 0; j < n; j++) ordem[j] = atual[j];

        }

    }

}

if ( MetodoTipo == 3 ) M = NEH(p, s, r, ordem, m, n);

if ( MetodoTipo == 4 ){

    calculo_residuo(p, s, r, UBX, m, n);

    for (int j = 0; j < n; j++) tarefa[j] = j;

    M = CNIT(p, s, r, m, n, ordem, tarefa, n);

}

if ( b_local ) M = buscalocal(p, s, r, m, n, ordem);

t2 = fabs(GetTime() - t1);

/*****          Rotina para o IG          *****/

if ( m_IG > 0 ){

    int C;

    float limite, Temperatura = 0.0;

```

```

for (int j = 0; j < n; j++) final[j] = ordem[j];

M_final = M;

if ( d_base < 10 ) d = d_base;

if ( d_base > 10 ) d = n * d_base / 10 ;

for (int i = 0; i < m; i++){

    for (int j = 0; j < n; j++) Temperatura += (float)p[i][j];

}

Temperatura = T * Temperatura / ((float)n * (float)m * 10.0);

limite = (float)n * ((float)m/2.0) * t / 1000.0;

if ( (m_IG == 2)&&(MetodoTipo!=2)&&(MetodoTipo!=2) )

    calculo_residuo(p, s, r, UBX, m, n);

do {

    float k;

    srand( time(NULL) );

    for (int j = 0; j < n; j++) atual[j] = ordem[j];

    if ( m_IG == 1 ) C = IG_original(p, s, m, n, d, atual);

    if ( m_IG == 2 ) C = CNIT_IG(p, s, r, m, n, d, atual);

    if ( b_local ) C = buscalocal(p, s, r, m, n, atual);

    k = (float)(rand()%101)/100;

    if ( C < M ){

```

```

    for (int j = 0; j < n; j++) ordem[j] = atual[j];

    M = C;

    if ( M < M_final ){

        for (int j = 0; j < n; j++) final[j] = ordem[j];

        M_final = M;

    }

    } else if ( k <= exp(-((float)C - (float)M)/Temperatura) ){

        for (int j = 0; j < n; j++) ordem[j] = atual[j];

        M = C;

    }

    t2 = fabs(GetTime() - t1);

    } while ( limite > t2 );

}

/*****

salvar(M_final, problema, t2, ordem, n, salva_seq);

for (int j = 0; j < n+1; j++) free(matriz[j]);

for (int i = 0; i < m; i++){

    for (int j = 0; j < n; j++){

        free(r[i][j]);

        free(UBX[i][j]);

    }

}

*****/

```



```
#include "principal.h"
```

```
void leparametros(char *nome, int *maquinas, int *tarefas)
```

```
{
```

```
    FILE *arquivo;
```

```
    int m, n;
```

```
    arquivo = fopen(nome, "r");
```

```
    fscanf(arquivo, "%d", &n);
```

```
    fscanf(arquivo, "%d", &m);
```

```
    *maquinas = m;
```

```
    *tarefas = n;
```

```
    fclose(arquivo);
```

```
}
```

```
/*
```

```
/*
```

```
/*
```

```
/*
```

```
/*                                leproblema.c                                */
```

```
#include "principal.h"
```

```
void leproblema(char *nome, int **proc, int ***setup, int m, int n)
```

```
{
```

```
    FILE *arquivo;
```

```
    int indice;
```



```

char aux[4];

arquivo = fopen(nome, "r");

fscanf(arquivo, "%d", &indice);

fscanf(arquivo, "%d", &indice);

for (int j = 0; j < n; j++){

    for (int i = 0; i < m; i++){

        fscanf(arquivo, "%d", &indice);

        fscanf(arquivo, "%d", &proc[i][j]);

    }

}

fscanf(arquivo, "%s", aux);

for (int i = 0; i < m; i++){

    fscanf(arquivo, "%s", aux);

    for (int j = 0; j < n; j++){

        for (int k = 0; k < n; k++)

            fscanf(arquivo, "%d", &setup[i][j][k]);

    }

}

fclose(arquivo);

}

/*****/

/*****/

/*****/

/*****/

```



```

        min1 = matriz[i][j];
        ind = j;
    }
    if ( matriz[i][j] < min2 && matriz[i][j] >= min1 && j != ind)
        min2 = matriz[i][j];
    }
}

matriz[i][n] = min2 - min1;
}
}

/* Encontra a diferença de cada coluna */
for (int j = 0; j < n; j++){
    if ( matriz[n][j] != -1 ){
        int ind;

        int min1 = INT_MAX;
        int min2 = INT_MAX;

        for (int i = 0; i < n; i++){
            if ( matriz[i][j] != -1 ){
                if ( matriz[i][j] < min1 ){
                    min2 = min1;
                    min1 = matriz[i][j];
                    ind = i;
                }
                if ( matriz[i][j] < min2 && matriz[i][j] >= min1 && i != ind)
                    min2 = matriz[i][j];
            }
        }
    }
}

```

```

        matriz[n][j] = min2 - min1;

    }

}

}

/*****/

int indices(int **matriz, int n, int *l, int *c)
{
    int dif_max = -1;
    int n_l = 0, n_c = 0;
    int linha[500], coluna[500];
    int ind_linha[500], ind_coluna[500];
    int aux_final = INT_MAX, l_final, c_final;

    /***** Encontra o maior valor entre a linha e a coluna *****/

    for (int i = 0; i < n; i++){
        if ( matriz[i][n] != -1 ){
            if ( matriz[i][n] > dif_max )
                dif_max = matriz[i][n];
        }

        if ( matriz[n][i] != -1 ){
            if ( matriz[n][i] > dif_max )
                dif_max = matriz[n][i];
        }
    }

    /***** Encontra todas as linhas e colunas empatadas *****/

    for (int i = 0; i < n; i++){

```

```

if ( matriz[i][n] == dif_max ){
    linha[n_l] = i;
    n_l++;
}

if ( matriz[n][i] == dif_max ){
    coluna[n_c] = i;
    n_c++;
}
}

/**** Entre as linhas e colunas empatadas, escolhe o menor valor *****/

for (int i = 0; i < n_l; i++){
    int aux = INT_MAX;
    for ( int j = 0; j < n; j++){
        if ( matriz[linha[i]][j] != -1 ){
            if ( matriz[linha[i]][j] < aux ){
                aux = matriz[linha[i]][j];
                ind_linha[i] = j;
            }
        }
    }

    if ( matriz[linha[i]][ind_linha[i]] < aux_final ){
        aux_final = matriz[linha[i]][ind_linha[i]];
        l_final = linha[i];
        c_final = ind_linha[i];
    }
}

for (int j = 0; j < n_c; j++){

```

```

int aux = INT_MAX;

for (int i = 0; i < n; i++){

    if ( matriz[i][coluna[j]] != -1 ){

        if ( matriz[i][coluna[j]] < aux ){

            aux = matriz[i][coluna[j]];

            ind_coluna[j] = i;

        }

    }

}

if ( matriz[ind_coluna[j]][coluna[j]] < aux_final ){

    aux_final = matriz[ind_coluna[j]][coluna[j]];

    l_final = ind_coluna[j];

    c_final = coluna[j];

}

}

/***** Saída por referência *****/

*l = l_final;

*c = c_final;

return dif_max;

}

/*****/

void VAM(int **matriz, int *ordem, int n)

{

    int c = 0, inicio = 0;

    int DE[500], PARA[500];

```



```

for (int i = 0; i < n; i++)

    DE[i] = PARA[i] = -1;


for (int k = 0; k < n-1; k++){

    int linha, coluna;

    int dif_max, aux1, aux2;


    diferenca(matriz, n);

    dif_max = indices(matriz, n, &linha, &coluna);

    DE[linha] = coluna;

    PARA[coluna] = linha;


    for (int i = 0; i <= n; i++){

        matriz[linha][i] = -1;

        matriz[i][coluna] = -1;

    }

    /***** Esta nova sub-rotina elimina os sub-tours *****/

    aux1 = coluna;

    while ( DE[aux1] != -1 ) aux1 = DE[aux1];


    aux2 = linha;

    while ( PARA[aux2] != -1 ) aux2 = PARA[aux2];


    matriz[aux1][aux2] = -1;

}

while ( PARA[inicio] != -1 ) inicio = PARA[inicio];

ordem[0] = inicio;

```

```

while ( DE[inicio] != -1 ) {

    c++;

    ordem[c] = DE[inicio];

    inicio = DE[inicio];

}

}

/*****

/*****

/*****

/*****

/*                      Makespan.c                      */

#include "principal.h"

int Makespan(int *ordem, int **p, int ***s, int m, int n)

{

    int C[20][500];

    C[0][0] = p[0][ordem[0]];

    for (int i = 1; i < m; i++)

        C[i][0] = C[i-1][0] + p[i][ordem[0]];

    for (int j = 1; j < n; j++){

        C[0][j] = C[0][j-1] + s[0][ordem[j-1]][ordem[j]] + p[0][ordem[j]];

        for (int i = 1; i < m; i++)

            C[i][j] = max(C[i-1][j], C[i][j-1] + s[i][ordem[j-1]][ordem[j]]) + p[i][ordem[j]];

```



```

int CNIT(int **p, int ***s, int ***r, int m, int n, int *ordem, int *tarefa, int d)
{
    int d_aux, h, M, aux2 = INT_MAX;

    int tentativa[500], final[500];

    /* Tarefa inicial se estiver inserindo todas */
    if ( d == n ){
        for (int j = 0; j < n; j++){
            int aux1 = 0;

            for (int i = 0; i < m; i++) aux1 += p[i][j];

            if ( aux1 < aux2 ){
                ordem[0] = j;
                aux2 = aux1;
            }
        }

        for (int j = 0; j < n; j++){
            if ( tarefa[j] == ordem[0] ) tarefa[j] = -1;
        }
    }

    if ( d == n ) d_aux = d - 1;
    else d_aux = d;

    /* Inserção das outras tarefas */
    for (int z = (n-d_aux); z < n; z++){
        /* Cálculo para a ordem atual das matrizes X, Q */
        int X[20][500], Q[20][500], e[20][500], q[20][500], f[20][501], Fi[501];

        for (int j = 0; j < z; j++){

```

```

    X[0][j] = 0;

    Q[m-1][j] = 0;

}

for (int i = 1; i < m; i++){

    X[i][0] = X[i-1][0] + p[i-1][ordem[0]];

    Q[m-1-i][z-1] = Q[m-i][z-1] + p[m-i][ordem[z-1]];

}

for (int j = 1; j < z; j++){

    for (int i = 1; i < m; i++){

        X[i][j] = r[i][ordem[j-1]][ordem[j]] + min(X[i][j-1], 0) + max(X[i-1][j], 0);

        Q[m-1-i][z-1-j] = -r[m-i][ordem[z-1-j]][ordem[z-j]] + min(Q[m-1-i][z-j], 0) +
max(Q[m-i][z-1-j], 0);

    }

}

/* Cálculo das matrizes e, q */

for (int i = 0; i < m; i++){

    e[i][0] = X[i][0] + p[i][ordem[0]];

    q[m-1-i][z-1] = Q[m-1-i][z-1] + p[m-1-i][ordem[z-1]];

}

for (int i = 0; i < m; i++){

    for (int j = 1; j < z; j++){

        e[i][j] = s[i][ordem[j-1]][ordem[j]] + max(X[i][j], 0) + p[i][ordem[j]] + e[i][j-
1];

        q[m-1-i][z-1-j] = max(Q[m-1-i][z-1-j], 0) + s[m-1-i][ordem[z-1-j]][ordem[z-
j]] + p[m-1-i][ordem[z-1-j]] + q[m-1-i][z-j];

    }

}

/* Tentativa com cada uma das tarefas */

```

```

M = INT_MAX;

for (int y = 0; y < d; y++){
    if ( tarefa[y] == -1 ) continue;

    for (int k = 0; k <= z; k++){

        int Fi_aux;

        tentativa[k] = tarefa[y];

        for (int j = 0; j < k; j++) tentativa[j] = ordem[j];

        for (int j = (k+1); j <= z; j++) tentativa[j] = ordem[j-1];


        if (k == 0){

            f[0][0] = p[0][tentativa[0]];

            for (int i = 1; i < m; i++) f[i][0] = f[i-1][0] + p[i][tentativa[0]];

        } else {

            f[0][k] = e[0][k-1] + s[0][tentativa[k-1]][tentativa[k]] +
p[0][tentativa[k]];

            for (int i = 1; i < m; i++)

                f[i][k] = max(f[i-1][k], e[i][k-1] + s[i][tentativa[k-1]][tentativa[k]] +
p[i][tentativa[k]]);

        }


        Fi[k] = 0;

        if (k < z){

            for (int i = 0; i < m; i++){

                Fi_aux = f[i][k] + s[i][tentativa[k]][tentativa[k+1]] + q[i][k];

                if (Fi_aux > Fi[k]) Fi[k] = Fi_aux;

            }

        } else {

            for (int i = 0; i < m; i++){

```



```

int buscalocal(int **p, int ***s, int ***r, int m, int n, int *ordem)
{
    int M = INT_MAX, melhoria = 1;
    int atual[500], tarefa[500];

    while (melhoria){
        int k, Cmax;
        melhoria = 0;
        for (int j = 0; j < n; j++) tarefa[j] = j;
        for (int z = 0; z < n; z++){
            int w, h, l = 0;
            int controle = 0;

            srand( time(NULL) );

            k = rand() % (n-z);
            for (w = 0; w < k; w++) while ( tarefa[w+l] == -1 ) l++;
            while ( tarefa[w+l] == -1 ) l++;
            h = tarefa[w+l];
            tarefa[h] = -1;

            for (int j = 0; j < (n-1); j++){
                if ( ordem[j] == h || controle ){
                    atual[j] = ordem[j+1];
                    controle = 1;
                } else atual[j] = ordem[j];
            }
        }
    }
}

```

```

    Cmax = insere_tarefa(p, s, r, m, n, atual, n-1, h);

    if (Cmax < M){

        M = Cmax;

        for (int j = 0; j < n; j++) ordem[j] = atual[j];

        melhora = 1;

    }

}

return(M);

}

/*****

/*****

/*****

/*****

/*                                inseretarefa.c                                */

#include "principal.h"

int insere_tarefa(int **p, int ***s, int ***r, int m, int n, int *ordem, int z, int h)
{

    int X[20][500], Q[20][500], e[20][500], q[20][500], f[20][501], Fi[501];

    int final[500];

    int C = INT_MAX;

    for (int j = 0; j < z; j++){

```

```

    X[0][j] = 0;

    Q[m-1][j] = 0;

}

for (int i = 1; i < m; i++){

    X[i][0] = X[i-1][0] + p[i-1][ordem[0]];

    Q[m-1-i][z-1] = Q[m-i][z-1] + p[m-i][ordem[z-1]];

}

for (int j = 1; j < z; j++){

    for (int i = 1; i < m; i++){

        X[i][j] = r[i][ordem[j-1]][ordem[j]] + min(X[i][j-1], 0) + max(X[i-1][j], 0);

        Q[m-1-i][z-1-j] = -r[m-i][ordem[z-1-j]][ordem[z-j]] + min(Q[m-1-i][z-j], 0) +
max(Q[m-i][z-1-j], 0);

    }

}

/* Cálculo das matrizes e, q */

for (int i = 0; i < m; i++){

    e[i][0] = X[i][0] + p[i][ordem[0]];

    q[m-1-i][z-1] = Q[m-1-i][z-1] + p[m-1-i][ordem[z-1]];

}

for (int i = 0; i < m; i++){

    for (int j = 1; j < z; j++){

        e[i][j] = s[i][ordem[j-1]][ordem[j]] + max(X[i][j], 0) + p[i][ordem[j]] + e[i][j-1];

        q[m-1-i][z-1-j] = max(Q[m-1-i][z-1-j], 0) + s[m-1-i][ordem[z-1-j]][ordem[z-j]] +
p[m-1-i][ordem[z-1-j]] + q[m-1-i][z-j];

    }

}

for (int k = 0; k <= z; k++){

```

```

int Fi_aux, tentativa[500];

tentativa[k] = h;

for (int j = 0; j < k; j++) tentativa[j] = ordem[j];

for (int j = (k+1); j <= z; j++) tentativa[j] = ordem[j-1];

/* Matriz f[i][j] */

if (k == 0){

    f[0][0] = p[0][tentativa[0]];

    for (int i = 1; i < m; i++)

        f[i][0] = f[i-1][0] + p[i][tentativa[0]];

} else {

    f[0][k] = e[0][k-1] + s[0][tentativa[k-1]][tentativa[k]] + p[0][tentativa[k]];

    for (int i = 1; i < m; i++)

        f[i][k] = max(f[i-1][k], (e[i][k-1] + s[i][tentativa[k-1]][tentativa[k]])) +
p[i][tentativa[k]];

}

/* Cálculo do Fi */

Fi[k] = 0;

if (k < z){

    for (int i = 0; i < m; i++){

        Fi_aux = f[i][k] + s[i][tentativa[k]][tentativa[k+1]] + q[i][k];

        if (Fi_aux > Fi[k]) Fi[k] = Fi_aux;

    }

} else {

    for (int i = 0; i < m; i++){

        Fi_aux = f[i][k];

        if (Fi_aux > Fi[k]) Fi[k] = Fi_aux;

    }

}

```



```

    atual[j] = ordem[j];
}

for (int z = 0; z < d; z++){
    int w, l = 0;
    int k, controle = 0;

    srand( time(NULL) );
    k = rand() % (n-z);
    for (w = 0; w < k; w++) while ( tarefa[w+l] == -1 ) l++;
    while ( tarefa[w+l] == -1 ) l++;
    destroi[z] = tarefa[w+l];
    tarefa[destroi[z]] = -1;

    for (int j = 0; j < (n-z-1); j++){
        if ( atual[j] == destroi[z] || controle ){
            atual[j] = atual[j+1];
            controle = 1;
        }
    }
}

for (int z = 0; z < d; z++)

    M = insere_tarefa(p, s, m, n, atual, n-d+z, destroi[z], 0);

for (int j = 0; j < n; j++) ordem[j] = atual[j];

return M;
}

```

```

/*****

```

```

/*****

/*****

/*****

/*                                CNIT-IG.c                                */

#include "principal.h"

int CNIT_IG(int **p, int ***s, int ***r, int m, int n, int d, int *ordem)
{
    int tarefa[500], atual[500];
    int M, *destroi;

    destroi = (int *) calloc (d, sizeof(int));
    for (int j = 0; j < n; j++){
        tarefa[j] = j;
        atual[j] = ordem[j];
    }

    for (int z = 0; z < d; z++){
        int w, l = 0;
        int k, controle = 0;

        srand( time(NULL) );
        k = rand() % (n-z);
        for (w = 0; w < k; w++) while ( tarefa[w+l] == -1 ) l++;
        while ( tarefa[w+l] == -1 ) l++;
    }
}

```

```

    destroi[z] = tarefa[w+1];
    tarefa[destroi[z]] = -1;

    for (int j = 0; j < (n-z-1); j++){
        if ( atual[j] == destroi[z] || controle ){
            atual[j] = atual[j+1];
            controle = 1;
        }
    }
}

M = CNIT(p, s, r, m, n, atual, destroi, d);
for (int j = 0; j < n; j++) ordem[j] = atual[j];
return M;
}

/*****

/*****

/*****

/*****

/*                                salvar.c                                */

#include "principal.h"

void salvar(int C, char problema[80], double tempo2, int *seq, int n, int salva_seq)
{
    FILE *arquivo;

```



```
arquivo = fopen("saidai.xls", "a");  
fprintf(arquivo, "%s\t", problema);  
fprintf(arquivo, "%d\t", C);  
fprintf(arquivo, "%0.3f\n", tempo2);  
if ( salva_seq ){  
    for (int i = 0; i < n; i++) fprintf(arquivo, "%d ", seq[i]);  
}  
fclose(arquivo);  
}
```