MÉTODOS HEURÍSTICOS CONSTRUTIVOS PARA O PROBLEMA DE PROGRAMAÇÃO DA PRODUÇÃO EM SISTEMAS *FLOW SHOP* HÍBRIDOS COM TEMPOS DE PREPARAÇÃO DAS MÁQUINAS ASSIMÉTRICOS E DEPENDENTES DA SEQÜÊNCIA

HÉLIO YOCHIHIRO FUCHIGAMI

Dissertação apresentada à Escola de Engenharia de São Carlos – Universidade de São Paulo, como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Produção.

Orientador: Prof. Titular João Vitor Moccellin

São Carlos

Fevereiro/2005

Dedico este trabalho,
concluído no ano
do cinqüentenário da
publicação do
pioneiro Johnson (1954)

Ao meu mestre da vida,
Daisaku Ikeda,
por ensinar o caminho direto
para a felicidade absoluta.

Aos meus pais,
Lourdes e Yochio,
pelo inestimável
legado: o estudo.

Ao meu melhor amigo,
Rafael, meu irmão preferido,
pela minha admiração
e pela sua confiança.

Agradecimentos

O ano 2004 foi o mais feliz da minha vida. Além de concluir este trabalho com muito orgulho, realizei o tão sonhado treinamento no Japão (que parecia impossível!) e encontrei-me com meu mestre Daisaku Ikeda.

Nunca gostei tanto de empreender um trabalho como neste ano. Quem me fez enxergar a minha paixão pela Pesquisa Operacional foi o professor Moccellin, um grande exemplo de profissional e um excelente orientador tanto na parte técnica como nos conselhos pessoais. Não sei o que é maior: o orgulho ou o privilégio de ser seu orientado. Deixar de valorizar é falta de gratidão.

Mas devo o que sou aos meus pais. É uma dívida que jamais conseguirei saldar. Eles não têm nem o ensino fundamental completo. Meus avós maternos apenas sabem ler. Os paternos, aprenderam o idioma português apenas para se comunicarem no Brasil. Com isto, quero mostrar minha origem humilde e o fato de que sou a pessoa mais graduada em minha família. Isto não me torna melhor que ninguém, mas responsável por incentivar todos a estudarem e a vencerem.

Agradeço aos meus amigos Ana Elisa, Fernando, Ana Laura, Edna, Alessandro, Valéria, Ava e todos os outros, que me viram "reflexivo" na fase de inspiração. Agradeço à minha família, que sentiu minha "falta de tempo" na fase de transpiração. Agradeço aos professores (principalmente ao Nagano) e funcionários do departamento, que me viram instrospectivo pela preocupação. E agradeço aos companheiros da BSGI de São Carlos, Ribeirão Preto e S.J.Rio Preto (e todas as outras cidades), bem como aos amigos do Grupo Sekai Kofu do Brasil, que acompanharam a minha vitória.

É claro que, sem a ótima infra-estrutura do departamento de Engenharia de Produção e a bolsa da CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) e do CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), não teríamos concluído esta pesquisa. Também agradeço à banca examinadora e à coordenação da pós-graduação.

Muito obrigado a todos!

Em 2005, continuarei trilhando a mesma estrada...

"Coragem! Vocês pertencem à geração do futuro.

Vocês realizarão grandes feitos."

Victor Hugo (1802-1885)

RESUMO

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

Este trabalho trata do problema de programação de operações no ambiente *flow shop* com máquinas múltiplas, com seus tempos de preparação (*setup*) assimétricos e dependentes da seqüência de processamento das tarefas. Este ambiente de produção é comum em indústrias gráficas, químicas, têxteis, de papel e de tinta, caracterizadas por sistemas com amplo *mix* de produtos. Qualquer processo produtivo requer um gerenciamento eficaz por meio do Planejamento e Controle da Produção (PCP). Esta atividade inclui a programação da produção, ou seja, a alocação de recursos para a execução de tarefas em uma base de tempo. A atividade de programação é uma das tarefas mais complexas no gerenciamento de produção, pois há a necessidade de lidar com diversos tipos diferentes de recursos e atividades simultaneamente. Além disso, o número de soluções possíveis cresce exponencialmente em várias dimensões, de acordo com a quantidade de tarefas, operações ou máquinas, conferindo uma natureza combinatorial ao problema. No ambiente estudado neste trabalho as operações de cada tarefa são executadas em múltiplos estágios de produção, podendo variar a quantidade de máquinas em cada um deles. Cada operação é processada por apenas uma máquina em cada estágio. Os tempos de preparação das máquinas possuem uma variabilidade relevante em função da ordem de execução das tarefas nas máquinas. A função-objetivo considerada é a minimização da duração total da programação (*makespan*). Foram desenvolvidos quatro **métodos heurísticos construtivos** com base em algoritmos reportados na literatura para solução de problemas *flow shop* permutacional e máquinas paralelas no ambiente cujo tempo de *setup* é dependente da seqüência. Como não foram encontrados na literatura métodos para programação no ambiente tratado neste trabalho, os algoritmos construídos foram comparados entre si. O foco da pesquisa foi o estudo da influência da relação entre as ordens de grandeza dos tempos de processamento e de *setup* em cada método de solução. Os resultados obtidos na experimentação computacional foram analisados e discutidos com base na porcentagem de sucesso, desvio relativo (%), desvio-padrão do desvio relativo e tempo médio de computação.

Palavras-chave: programação da produção, *flow shop* híbrido, *setup* dependente, métodos heurísticos.

ABSTRACT

FUCHIGAMI, H.Y. (2005). ***Construtive heuristic methods for hybrid flow shop scheduling problem with asymmetric sequence dependent setup times***. *M. Sc. Dissertation* – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos. 2005.

This work adressess the hybrid flow shop scheduling problem with asymmetric sequence dependent setup times. This environment of production system is common in graphical, chemical, fabric, paper and ink industries. It's characterized by systems with large mix of products. Any productive process requires an efficient management by means of Production Planning and Control. This activity includes scheduling, i.e., the resources allocation for the execution of jobs in a time base. Scheduling is one of the tasks most complex in production management, since it deals simultaneously with different types of resources and activities. Moreover, the number of possible solutions grows exponentially in some dimensions, in accordance with the number of jobs, operations or machines, conferring a combinatorial nature to the problem. In the environment studied in this work, the operations of each job are processed in multiple production stages. The number of machines in each stage can be different. Each operation is processed by only one machine in each stage. The setup times have a significant variability in function of the sequence of job processing on the machines. The objective is minimizing the total time to complete the schedule (makespan). Four ***constructive heuristic methods*** *were developed on the basis of algorithms reported in the literature for solving permutation flow shop and parallel machine problems with sequence dependent setup times. The proposed heuristic methods have been compared between themselves, since no constructive heuristics have been found in the literature for the scheduling problem considered in this work. The focus of the research was the study of the influence of the relations among the range of the times processing and setup times in each method. The statistics used in order to evaluate the heuristic performances were the percentage of success (in finding the best solution), relative deviation, standard deviation of relative deviation and average computation time. Results from computational experience are discussed.*

Keywords: *production scheduling, hybrid flow shop, sequence dependent setup times, heuristics.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CDS	Algoritmo de Campbell, Dudek e Smith (1970)
CPM	*Critical Path Method*
DR	Desvio Relativo
DRM	Desvio Relativo Médio
EDD	*Earliest Due Date*
FS	*Flow Shop*
FSH	*Flow Shop* Híbrido
FSP	*Flow Shop* Permutacional
FIFO	*First In First Out*
GRASP	*Greedy Randomized Adaptive Search* de Ríos-Mercado e Bard (1998)
LPST	*Longest Processing-Setup Time*
LPT	*Longest Processing Time*
NEH	Algoritmo de Nawaz, Enscore Jr. e Ham (1983)
N&M	Método Heurístico de Nagano e Moccellin (2002)
MAV	Método de Aproximação de Vogel (REINFELD e VOGEL, 1958)
MRP	*Material Requirements Planning*
MRP II	*Manufacturing Resources Planning*
MST	*Minimum Slack Time*
OPT	*Optimized Production Tecnology*
PCP	Planejamento e Controle da Produção
PERT	*Program Evaluation and Review Technique*
SCT	*Shortest Completion Time*
SPT	*Shortest Processing Time*
SRD	*Shortest Release Date*
TSP	*Traveling Salesman Problem* (Problema do Caixeiro Viajante)

LISTA DE SÍMBOLOS

B	número de lotes (*batches*)
C_i	data de término da tarefa J_i
C_{max}	data de término máxima das tarefas (*makespan*)
C_{ik}	data de término da tarefa J_i no estágio k
$C_{E_{ik}}$	data mais cedo de término da tarefa J_i no estágio k
d_i	data de entrega (*due date*) da tarefa J_i
D_h	*makespan* obtido pelo método h
D^*	melhor *makespan* obtido pelos métodos
E_i	*earliness* da tarefa J_i
E_{max}	*ealiness* máximo de uma seqüência de tarefas
F_i	tempo de fluxo ou tempo de permanência da tarefa J_i
F_{max}	tempo máximo de fluxo das tarefas
$\overline{F}$	tempo médio de fluxo de uma seqüência de tarefas
J	conjunto das n tarefas $\{J_1, J_2, ..., J_n\}$
J_i	tarefa de índice i
$J_{[i]}$	tarefa que ocupa a i-ésima posição numa seqüência
J'	conjunto das tarefas ainda não programadas
k	estágio de produção
K	número de estágios de produção
L_i	*lateness* da tarefa J_i
L_{max}	atraso máximo
m_i	máquina de índice i
M	número total de máquinas do problema
M_k	número de máquinas no estágio k
n	número de tarefas
op_{ij}	operação j da tarefa J_i
$O(x)$	ordem de grandeza da variável x
p_{ik}	tempo de processamento da operação da tarefa J_i no estágio k

p_{max} tempo máximo de processamento da tarefa

$\overline{p}_m$ tempo médio de processamento das tarefas na máquina m

r_i data de liberação (*release date*) da tarefa J_i

s_{ij} tempo de preparação (*setup*) da tarefa J_i para a tarefa J_j

s_{ijk} tempo de preparação (*setup*) da tarefa J_i para a tarefa J_j no estágio k

S_x desvio-padrão da variável x

T_i *tardiness* da tarefa J_i

T_{max} *tardiness* máximo

u_m índice da última tarefa alocada na máquina m

w peso de uma variável

w_i peso da tarefa J_i

m_k conjunto dos índices das máquinas do estágio k

π seqüência de tarefas

σ_m subconjunto de tarefas alocadas à máquina m

$|\sigma_m|$ cardinalidade do subconjunto de tarefas σ_m

$\cdot>$ dominância (exemplo: $m_1 \cdot> m_2$ significa que m_1 *domina* m_2)

SUMÁRIO

1 INTRODUÇÃO

1.1 Planejamento e controle da produção

Conforme Harding (1981, p.24), um **sistema de produção** "é um conjunto de partes inter-relacionadas, as quais quando ligadas atuam de acordo com padrões estabelecidos sobre *inputs* (entradas) no sentido de produzir *outputs* (saídas)". Pode-se classificar os sistemas de produção de várias formas. Erdmann (2000, p.19) define como **determinísticos** os sistemas exatamente previsíveis em suas operações e **probabilísticos** aqueles com previsões das atividades em termos de probabilidade. Johnson e Montgomery (1974) sugerem uma classificação em três categorias, segundo o processo de fluxo de materiais:

i) **Sistema de grande projeto**: produção de itens complexos e/ou de grande porte, na maioria dos casos com lotes unitários;

ii) **Sistema contínuo**: produção em larga escala de produtos padronizados e com pouca diversificação; e

iii) **Sistema intermitente**: caracterizado pela flexibilidade, ou seja, a capacidade de produzir uma grande variedade de produtos. Este sistema é subdividido em dois tipos:

- ***Flow shop***: a maioria dos itens fabricados em uma linha de produção ou célula de manufatura tem a mesma seqüência de operações nas diversas máquinas; e
- ***Job shop***: a seqüência de execução das operações se modifica de um produto para o outro.

Pode-se dizer que um sistema de produção é formado por três subsistemas interativos:

i) **Estrutural**: constituído pela parte física do sistema, como matéria-prima, instalações etc.;

ii) **Social**: formado pelas relações sociais entre as pessoas da empresa; e

iii) **Organizacional**: composto pela hierarquia de decisão, departamentos etc.

Dentro do terceiro subsistema encontra-se o **Planejamento e Controle da Produção** (PCP), que se constitui de um conjunto de várias funções com o objetivo de comandar e gerenciar o processo produtivo. Essencialmente, tais funções são: planejamento de recursos de longo prazo, planejamento agregado da produção, plano mestre de produção, planejamento das necessidades de materiais, controle de estoques, planejamento da capacidade, liberação de ordens e *programação da produção.*

Qualquer operação produtiva requer o gerenciamento das suas atividades de modo a satisfazer a demanda dos consumidores, ou seja, precisa realizar o **planejamento e controle da produção**, cujo propósito é garantir que a produção ocorra eficazmente. Isto requer que os recursos produtivos estejam disponíveis na *quantidade*, no *momento* e no *nível de qualidade* adequados (SLACK *et al.*, 1999, p.230).

Segundo Slack *et al.* (1999, p.232), a divisão entre planejamento e controle não é clara nem na teoria nem na prática, em muitas situações. Entretanto, há algumas características gerais que os distinguem. Um **plano** é uma formalização do que se pretende que aconteça em determinado momento no futuro e não garante que um evento realmente acontecerá. Há muitas diferentes variáveis que podem contribuir para que um plano torne-se não executável. E o **controle** é o processo de lidar com essas variáveis, pois realiza os ajustes que permitem que a operação atinja os objetivos que o plano estabeleceu, mesmo que as suposições feitas pelo plano não se confirmem.

1.2 Programação da produção

A **programação da produção** pode ser definida como: (1) a determinação de *quando* e *onde* cada operação necessária para a fabricação de um produto deve ser realizada ou (2) a determinação de *datas* nas quais iniciar e/ou completar cada *evento* ou

operação que compõe um procedimento. Portanto, pode-se dizer basicamente que a programação da produção consiste na alocação de *recursos* para a execução de *tarefas* em uma *base de tempo.*

Segundo Slack *et al.* (1999, p.245), a atividade de programação é uma das mais complexas tarefas no gerenciamento de produção. Primeiro, os programadores precisam lidar com diversos tipos diferentes de recursos simultaneamente. As máquinas terão diferentes capacidades e o pessoal terá diferentes habilidades. De maneira mais importante, o número de programações possíveis cresce rapidamente à medida que o número de atividades e processos aumenta. Ou seja, para n tarefas há $n!$ (n fatorial) maneiras diferentes de programação dos trabalhos em um processo simples. Considerando mais que uma máquina ($M > 1$), o número de programações possíveis passa para $(n!)^M$.

Assim, a primeira característica que torna os problemas de programação difíceis de resolver é a sua natureza combinatorial, o que significa que o número de soluções possíveis cresce exponencialmente em várias dimensões, de acordo com a quantidade de tarefas, operações ou máquinas.

Uma ferramenta comumente usada na programação é o **Gráfico de Gantt**, inventado por H. L. Gantt em 1917 e que representa graficamente o tempo como uma barra. Os momentos de início e fim de atividades podem ser indicados no gráfico e algumas vezes o progresso real do trabalho também é indicado. As vantagens dos gráficos de Gantt é que eles proporcionam uma representação visual simples do que deve ocorrer em cada operação (SLACK *et al.*, 1999, p.244-245).

A Figura 1.1 mostra um exemplo de Gráfico de Gantt com a execução das operações de uma tarefa em três máquinas diferentes.

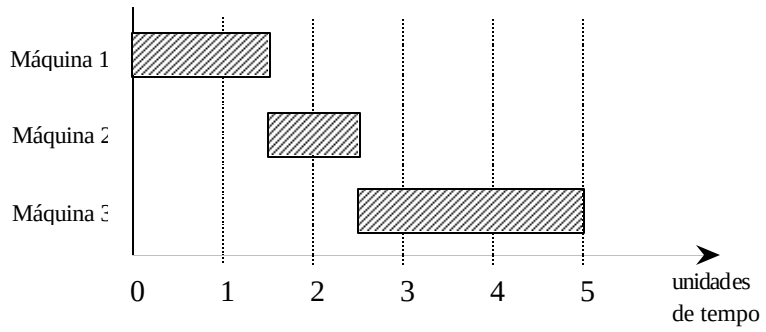


FIGURA 1.1 – Exemplo de Gráfico de Gantt

De acordo com Erdmann (2000, p.46), podem ser citadas como técnicas de programação:

- MRP (*Material Requirement Planning*, ou planejamento das necessidades de material);
- MRP II (*Manufacturing Resources Planning*, ou planejamento de recursos de manufatura);
- Kanban (técnica de comando da produção por sinalização visual);
- Software incorporado ao OPT (*Optimized Production Tecnology*);
- Programação por redes (PERT e CPM);
- Programação orientada pela carga dos recursos de produção;
- Ativação da produção pelo estoque;
- Programação da produção por períodos de tempo;
- Programação por tamanhos de lote;
- Programação para atendimentos de pedidos.

1.3 Problemas de programação de operações em máquinas

Conforme consta em Pinedo (1995, p.9), um problema de programação de operações em máquinas é descrito pela notação padrão de três campos **a | b | g**. O primeiro campo, **a**, representa o ambiente de máquinas. Por exemplo, para máquina única, o **a** é representado por "1", no *flow shop* é "*Fm*" e o ambiente de máquinas paralelas idênticas é denotado por "*Pm*".

O campo **b** fornece detalhes das características de processamento e restrições, podendo estar vazio, conter um ou múltiplos parâmetros. Alguns exemplos são: restrições de precedência (*prec*), de eligibilidade de máquina (M_j), permutacional (*prmu*), com datas de liberação ou *release dates* (r_j), tempos de preparação dependentes da seqüência (s_{jk}). E o campo **g** contém a função-objetivo, que pode ser *makespan* (C_{max}), *lateness* máximo (L_{max}), soma ponderada das datas de término ($\Sigma w_i C_i$), entre outras (PINEDO, 1995, p.10-14).

Os problemas geralmente também se tornam complexos pelo grande número de restrições relacionando uma atividade a outra, recursos a atividades, um recurso ao outro e um recurso ou atividade a eventos externos ao sistema. Por exemplo, pode haver

uma **restrição de precedência** que especifica quais atividades devem preceder outras. Também pode não ser possível usar dois recursos simultaneamente durante um certo período de tempo ou em alguma atividade, ou então um recurso pode não estar disponível durante um intervalo de tempo específico devido a manutenção. Como estes complexos inter-relacionamentos podem tornar muito difícil a busca da solução exata ou mesmo aproximada de um grande problema, é natural resolver primeiro versões mais simples. Então, a sensibilidade da solução pode ser testada quanto a sua complexidade e soluções aproximadas podem ser encontradas para problemas difíceis (MORTON e PENTICO, 1993, p.6)

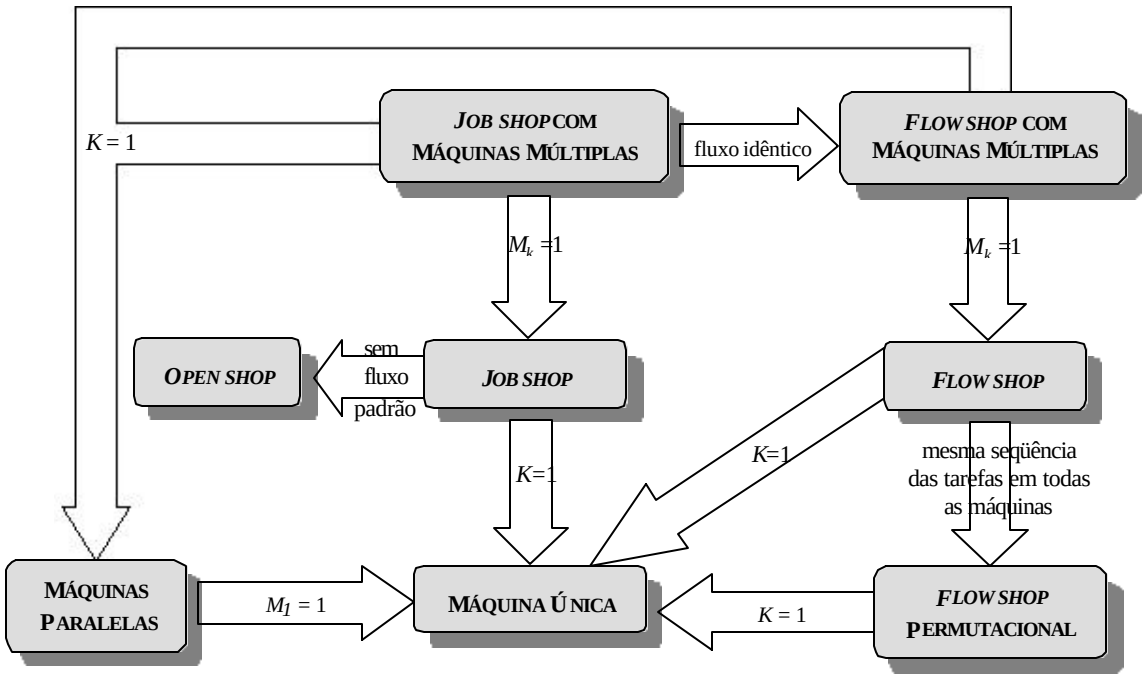
O problema de minimização do ***makespan***, ou duração total da programação, tem sido bastante estudado na literatura. Isto porque este critério de desempenho é simples e útil para ambientes muito carregados, onde a utilização dos recursos em longo prazo precisa ser otimizada. O *makespan* é também a única função-objetivo suficientemente simples para fornecer resultados analíticos para problemas de múltiplas máquinas e para tornar métodos *Branch and Bound* praticáveis em problemas de tamanho médio (MORTON e PENTICO, 1993, p.302).

Os modelos de programação são aplicados em ambientes específicos, podendo inclusive ser uma adaptação de um modelo já existente ou então um procedimento híbrido. As restrições tecnológicas são determinadas principalmente pelo fluxo das tarefas nas máquinas, levando a uma classificação dos problemas conforme segue:

- ***Job shop***: cada tarefa tem sua própria seqüência de processamento nas máquinas;
- ***Flow shop***: todas as tarefas possuem o mesmo fluxo de processamento nas máquinas;
- ***Open shop***: não há uma seqüência preestabelecida para as tarefas;
- ***Flow shop* permutacional**: *flow shop* em que a ordem de processamento das tarefas em cada máquina é estritamente a mesma;
- **Máquina única**: existe apenas uma única máquina disponível para o processamento das tarefas;
- **Máquinas paralelas**: em um mesmo estágio de produção, há duas ou mais máquinas disponíveis que podem executar qualquer tarefa;

- ***Job shop* com máquinas múltiplas**: *job shop* em que existem duas ou mais máquinas paralelas em cada estágio, sendo que cada tarefa é processada por somente uma máquina em cada um dos estágios;
- ***Flow shop* com máquinas múltiplas**: as tarefas são processadas em múltiplos estágios e em cada um deles há máquinas paralelas, podendo variar a quantidade por estágio. As tarefas são processadas por apenas uma máquina em cada estágio.

A Figura 1.2 ilustra a relação entre as classes de problemas descritas.



K: número de estágios de produção
M_k: número de máquinas do estágio k (com k = 1, 2, ..., K)

FIGURA 1.2 – Relação entre as classes de problemas de programação de operações em máquinas (adaptado de MACCARTHY e LIU, 1993)

1.3.1 Conceito de tempos de preparação dependentes da seqüência

O **tempo de preparação** da máquina ou **tempo de *setup*** inclui o trabalho de preparar a máquina, o processo ou a oficina para a fabricação de produtos

(OSTWALD[1], 1992 *apud* ALLAHVERDI, GUPTA e ALDOWAISAN, 1999). Isto inclui o tempo para obtenção das ferramentas, posicionamento dos materiais a serem usados no trabalho, processos de limpeza, preparação e ajuste das ferramentas e inspeção de materiais (ALLAHVERDI, GUPTA e ALDOWAISAN, 1999).

Liaee e Emmons (1997) apresentaram uma classificação por critério de desempenho dos problemas de processamento de famílias de tarefas com tempos de *setup*. Allahverdi, Gupta e Aldowaisan (1999) fizeram uma revisão da literatura de problemas de programação da produção envolvendo tempos de preparação. Os problemas foram classificados em *batch* e *non-batch* e também como *setup* dependente e independente da seqüência de tarefas. Foram considerados os diversos ambientes de fabricação como máquina única, máquinas paralelas, *flow shops* e *job shops*. O artigo salientou os resultados das pesquisas em diferentes tipos de problemas, e ainda forneceu direcionamentos para futuras pesquisas.

O exemplo clássico para este ambiente é o **Problema do Caixeiro Viajante** (*Traveling Salesman Problem* – TSP), proposto em 1934 por Hassler Whitney em um seminário na Universidade de Princeton. O nome do problema deve-se ao fato de representar a viagem de um caixeiro que deseja percorrer o caminho mais curto de sua moradia a cada uma das cidades que deve visitar e retornar para o ponto de origem. Conhecendo as distâncias entre cada par de cidades da viagem, pode-se utilizar um método de otimização para encontrar o caminho desejado (FLOOD, 1956). A relação com o problema de programação da produção ocorre quando se deseja determinar uma seqüência de tarefas a ser processada por uma máquina, com seus tempos de preparação dependentes da ordem de execução, de tal forma a minimizar o tempo total de preparação desta máquina. Os tempos de preparação da máquina são análogos às distâncias entre as cidades no Problema do Caixeiro Viajante.

Muitas pesquisas em programação da produção desconsideram os tempos de preparação (*setup*) das máquinas ou então os incluem nos tempos de processamento das tarefas. Isto simplifica a análise das aplicações, porém afeta a qualidade da solução quando tais tempos têm uma variabilidade relevante em função da ordenação das tarefas nas máquinas.

Existem dois tipos de problemas que requerem que o tempo de *setup* seja explícito ou **separado** do tempo de processamento das tarefas: tempos de *setup*

[1] OSTWALD, P.F. (1992). Cost estimating. **Handbook of Industrial Engineering**, 2nd ed., p.1263-1288.

independentes da seqüência, em que o *setup* depende somente da tarefa a ser processada, e tempos de *setup* **dependentes da seqüência**, cujo *setup* depende tanto da tarefa a ser processada quanto daquela que foi processada imediatamente antes (ALLAHVERDI, GUPTA e ALDOWAISAN, 1999).

Neste trabalho, foi considerado o ambiente em que os tempos de *setup* são dependentes da seqüência de execução das tarefas. Isto significa que o tempo de *setup* para a tarefa J_c, tendo a máquina previamente realizada a tarefa J_a, é diferente do tempo de *setup* para a mesma tarefa J_c, porém tendo anteriormente executada a tarefa J_b. Pinedo (1995, p.48) mostrou que este problema é classificado como fortemente *NP-hard*.

Esta abordagem é necessária em sistemas de produção como indústrias químicas, na produção de tinta por exemplo, onde o processo de limpeza é diferenciado dependendo da cor que estava sendo produzida e daquela que será fabricada em seguida.

Como o *setup* é uma atividade que não agrega valor, muitas pesquisas nas últimas décadas têm se voltado para técnicas de seqüenciamento que minimizem o tempo total de *setup* (REDDY e NARENDRAN, 2003). Como pode ser visto em Burbidge (1975), Robinson (1990) e Shingo (1996), há várias vantagens em reduzir o *setup* no processo produtivo. Além disso, a variação na matriz de tempos de *setup* das máquinas influencia diretamemente o *makespan*.

1.3.2 Conceito de tempos de preparação assimétricos

Os tempos de *setup*, além de serem dependentes da seqüência, também são **assimétricos**, ou seja,

$$s_{ab} \neq s_{ba}\,.$$

Isto indica que o tempo de preparação entre as operações das tarefas J_a e J_b (s_{ab}) é diferente do tempo de preparação entre operações das tarefas J_b e J_a (s_{ba}).

Estas características de assimetria e dependência trazem uma variação na soma total dos tempos de *setup* em função do seqüenciamento das tarefas, levando à variação no valor do critério de desempenho, aqui considerado o *makespan* ($C_{\max}$).

1.4 Métodos de solução

Nas últimas décadas, um considerável esforço de pesquisa tem sido dedicado à solução dos problemas de programação de operações em máquinas. Algumas **técnicas para solução ótima** têm sido empregadas tais como as de Programação Matemática, por exemplo os modelos de Programação Linear Inteira de Selen e Hott (1986) e Wilson (1989), e técnicas de enumeração *Branch and Bound* de Ignall e Schrage (1965) e Potts (1980). Entretanto, tais técnicas não são eficientes computacionalmente em problemas de grande e médio porte. Por este motivo, muitos métodos heurísticos têm sido propostos para a solução do problema.

Um **método heurístico** é um processo de solução de problema apoiado em critérios racionais ou computacionais para escolher um caminho entre vários possíveis, sem a preocupação de percorrer todas as possibilidades ou atingir a melhor opção. Esta busca por um determinado objetivo visa encontrar uma solução viável, pelo menos próxima da ótima, cujo tempo de computação seja aceitável. Em geral, a pequena diferença entre a solução heurística e a ótima não justifica o grande esforço computacional requerido para atingir esta última.

Os métodos heurísticos podem ser classificados de diversas formas, sendo em geral divididos em dois grupos, conforme apresentado por Souza e Moccellin (2000):

- **Métodos construtivos**: a seqüência adotada como solução do problema é obtida:

  i) diretamente a partir da ordenação das tarefas segundo índices de prioridade calculados em função dos tempos de processamento das tarefas, como por exemplo em Palmer (1965) e Gupta (1971); ou

  ii) escolhendo-se a melhor seqüência das tarefas a partir de um conjunto de seqüências também obtidas utilizando-se índices de prioridade associados às tarefas como em Campbell, Dudek e Smith (1970) e Hundal e Rajgopal (1988); ou ainda

  iii) a partir da geração sucessiva de seqüências parciais (subseqüências) das tarefas até a ordenação de uma seqüência completa através de algum critério de inserção de tarefas, como por exemplo em NEH (NAWAZ, ENSCORE JR. e HAM, 1983) e N&M (NAGANO e MOCCELLIN, 2002).

- **Métodos melhorativos**: obtém-se uma solução inicial e posteriormente através de algum procedimento iterativo (geralmente envolvendo trocas de posições das tarefas na seqüência de processamento das máquinas) busca-se encontrar uma programação das tarefas melhor que a atual quanto à medida de desempenho adotada.

  Nesta categoria, destacam-se os procedimentos de busca em vizinhança, como por exemplo em Dannenbring (1977), considerado um método de busca simples.

  Outros métodos de maior complexidade como Busca Tabu, *Simulated Annealing* e Algoritmo Genético têm gerado aplicações bem sucedidas reportadas na literatura. Tais métodos, conhecidos como **metaheurísticas**, consistem de procedimentos de busca no espaço de soluções, definidos por estratégias que exploram apropriadamente a topologia de tal espaço.

  O sucesso das metaheurísticas deve-se a fatores como:

  i) alusão a mecanismos de otimização da natureza (nos casos do Algoritmo Genético e do *Simulated Annealing*);

  ii) aplicabilidade geral da abordagem;

  iii) facilidade de implementação; e

  iv) qualidade da solução aliada a um esforço computacional relativamente baixo.

1.5 Objetivos e estrutura do trabalho

Este trabalho investiga o problema de *flow shop* com máquinas múltiplas e tempos de preparação das máquinas assimétricos e dependentes da seqüência, e propõe métodos heurísticos construtivos de solução para a minimização da duração total da programação (*makespan*). Este ambiente tem sido pouco estudado, conforme o exame da literatura apresentado no capítulo 2. Esta foi uma primeira motivação para realização desta pesquisa.

Esse ambiente de produção é comum em indústrias gráficas, químicas, têxteis, de papel e de tinta, caracterizadas por sistemas com amplo *mix* de produtos. Nestes

ambientes, a criação de ferramentas aplicáveis é fundamental para maximizar a utilização dos recursos. Além disso, com a atual disponibilidade de recursos tecnológicos nas empresas, sistemas eficazes para programação de tarefas tornam-se muito valiosos, pois em geral as decisões são tomadas com base na experiência e desconsiderando processos de otimização.

Desde o início da década de 1990, os estudos referentes à área de Programação de Operações em Máquinas na Escola de Engenharia de São Carlos – USP resultaram em uma tese de livre docência, uma de doutorado e cinco dissertações de mestrado (ver apêndice A). O desenvolvimento sucessivo de trabalhos com o ambiente *flow shop* tratou de modelos com complexidade crescente e procurou utilizar cumulativamente os resultados obtidos.

Do estudo do problema de programação *flow shop* tradicional passou-se ao desenvolvimento de metaheurísticas para *flow shop* permutacional, incluindo métodos heurísticos híbridos com Algoritmo Genético, *Simulated Annealing* e Busca Tabu. Modelos híbridos, ou seja, agregando *flow shop* e máquinas paralelas também foram abordados. E em seguida, foram tratados modelos com estágios gargalos, com um deles considerando os tempos de preparação das máquinas dependentes da seqüência.

Com base nos resultados encorajadores das pesquisas, este trabalho objetiva dar continuidade à investigação e estudo do problema de programação da produção em ambiente *flow shop* híbrido. Este foi uma segunda motivação para o desenvolvimento desta pesquisa.

O texto foi estruturado da seguinte forma: o primeiro capítulo apresenta uma introdução sobre Planejamento e Controle da Produção (PCP), enfatizando o problema de programação de operações em máquinas. O segundo capítulo traz uma revisão bibliográfica de vários ambientes *flow shop* e máquinas paralelas. A contextualização do problema tratado e a proposição dos métodos heurísticos construtivos para solução, incluindo as regras de prioridade e de alocação, constam no terceiro capítulo. O quarto capítulo apresenta o delineamento da experimentação computacional e a análise dos resultados obtidos. As conclusões do trabalho são discorridas no quinto capítulo.

Na área de programação da produção há muitos termos técnicos em inglês que não encontram tradução fiel em português e uma aproximação do significado poderia comprometer a sua interpretação. Por este motivo, neste trabalho algumas vezes optou-se por manter os termos em inglês e apresentar um glossário no final do texto.

2 PROGRAMAÇÃO DE OPERAÇÕES EM AMBIENTES *FLOW SHOP* E MÁQUINAS PARALELAS

2.1 *Flow shop* permutacional

Desde o trabalho pioneiro de Johnson (1954) que aborda *flow shop* permutacional com duas máquinas, muitas pesquisas têm sido conduzidas na busca de métodos exatos e heurísticos para o problema *flow shop* permutacional. Devido à natureza do problema ser *NP-completo*, os pesquisadores têm enfocado principalmente o desenvolvimento de heurísticas e metaheurísticas. Entre as heurísticas mais conhecidas estão a de Campbell, Dudek e Smith (1970), denominada método **CDS**, e a de Nawaz, Enscore Jr. e Ham (1983), o método **NEH**.

Algumas metaheurísticas relevantes são: *Simulated Annealing* de Osman e Potts (1989), Busca Tabu de Widmer e Hertz (1989) e Algoritmo Genético de Reeves (1995).

Gangadharan e Rajendran (1994) consideraram o problema de programação em ambiente *flow shop* com bicritério de otimização – minimização do *makespan* e do tempo total de fluxo (*flowtime*). A heurística proposta foi baseada no método *Simulated Annealing* de Kirkpatrick (1983). Também foram apresentados dois algoritmos para fornecer as seqüências sementes para cada critério. Experimentos computacionais mostraram a superioridade dos métodos propostos em relação aos existentes anteriormente.

Moccellin (1995) propôs um novo método heurístico semelhante à Busca Tabu de Widmer e Hertz (1989), com um desempenho superior a este último. A nova heurística forneceu resultados superiores em termos de qualidade da solução para problemas com até 50 tarefas e 30 máquinas em comparação com o método de Widmer e Hertz (1989) e também com o NEH.

Dois casos especiais de programação *flow shop* permutacional foram examinados por Ho e Gupta (1995). O primeiro problema considerou uma série crescente de **máquinas dominantes**; e o segundo, uma série decrescente. Uma máquina m_a domina uma outra máquina m_b (denotado por $m_a \cdot\!> m_b$) se $\min_{J_i \in J}\{p_{ia}\} \geq \max_{J_i \in J}\{p_{ib}\}$, onde $J = \{J_1, J_2, ..., J_n\}$ é o conjunto de todas as n tarefas. Uma **série crescente** de máquinas dominantes é uma ordenação de máquinas tal que $m_1 <\!\cdot\, m_2 <\!\cdot\, ... <\!\cdot\, m_m$. Analogamente, uma **série decrescente** de máquinas dominantes é uma ordenação da seguinte forma: $m_1 \cdot\!> m_2 \cdot\!> ... \cdot\!> m_m$. Foram desenvolvidos procedimentos para encontrar a programação permutacional ótima para várias medidas de desempenho como *makespan*, tempo médio de fluxo, tempo médio de utilização das máquinas, número de tarefas em atraso e atraso máximo.

Vários trabalhos com programação de operações *flow shop* permutacional são citados na tese de Nagano (1999).

Suliman (2000) propôs um método de solução para o problema *flow shop* permutacional em duas fases. Na primeira, uma seqüência inicial de tarefas é gerada usando algoritmos construtivos eficientes computacionalmente e, na segunda fase, a seqüência gerada é melhorada para o critério da minimização do *makespan* por meio do mecanismo de troca de pares de tarefas com direção restrita. Os resultados forneceram boas soluções para problemas de pequeno porte em termos de tempo computacional e desempenho similar ao método NEH.

Moccellin e Santos (2000) desenvolveram um método heurístico híbrido para minimizar o *makespan* da programação *flow shop* permutacional, combinando os algoritmos de Busca Tabu e *Simulated Annealing*. Souza e Moccellin (2000) utilizaram a mesma idéia para associar Algoritmo Genético e Busca Tabu. Completando as três combinações de pares de metaheurísticas Busca Tabu, *Simulated Annealing* e Algoritmo Genético, Buzzo e Moccellin (2000) apresentaram um método híbrido com as duas últimas. Cada procedimento híbrido proposto foi comparado com os métodos puros. Os resultados experimentais obtidos mostraram a possibilidade de agregar características vantajosas dos métodos metaheurísticos puros para o desenvolvimento de um método híbrido mais eficaz do que qualquer um dos seus componentes isoladamente.

O problema com séries crescentes e decrescentes de máquinas dominantes também foi tratado por Xiang, Tang e Cheng (2000). Foram apresentados cinco

algoritmos de solução polinomial para resolver problemas com medidas de desempenho regulares, ou seja, soma ponderada das datas de término (representado por Ów_iC_i, onde w_i é peso da tarefa J_i e C_i é a data de término da tarefa J_i), *lateness* máximo, *tardiness* máximo, número de tarefas em atraso e *makespan*.

Braun *et al*. (2002) examinaram o problema de minimização do *makespan* em um ambiente de programação *flow shop* com duas máquinas e a presença de intervalos de não-disponibilidade (SCHMIDT, 2000). Se não houver tais intervalos, o problema pode ser facilmente resolvido usando o algoritmo de Johnson (1954). Foi utilizada a análise de estabilidade, que demonstra o quão estável é uma programação ótima se houver mudanças independentes no tempo de processamento das tarefas.

Um novo método heurístico construtivo de alta qualidade para minimização do *makespan*, denominado **N&M**, foi introduzido por Nagano e Moccellin (2002). O método foi comparado com a melhor heurística construtiva até então reportada na literatura – a NEH. A heurística NEH prioriza tarefas com maior tempo de processamento em todas as máquinas. O método N&M penaliza a prioridade NEH de acordo com um limite inferior para o tempo de espera de uma tarefa entre o fim de sua operação em uma máquina qualquer e o início da operação na próxima. Os resultados computacionais demonstraram que para problemas com até 10 máquinas e 100 tarefas, o novo método supera a heurística NEH. Não há nenhuma diferença significativa no esforço computacional de ambas as heurísticas.

Uma generalização do algoritmo NEH foi realizada por Framinan, Leisten e Rajendran (2003), com objetivo de minimização do *makespan*, do tempo ocioso de máquina e do tempo de fluxo. Foram construídas 177 ordenações iniciais diferentes de tarefas e seu desempenho foi avaliado pelo método de inserção-NEH.

Uma revisão e avaliação comparativa de heurísticas e metaheurísticas para problemas *flow shop* permutacionais com o critério *makespan* foi realizada por Ruiz e Maroto (2004). A pesquisa compara 25 métodos, desde o clássico algoritmo de Johnson (1954) e regras de prioridade até as mais recentes metaheurísticas, incluindo Busca Tabu, *Simulated Annealing*, Algoritmos Genéticos, Busca Local Iterativa e técnicas híbridas. O trabalho classifica heurísticas construtivas e de melhoria e metaheurísticas, além de fazer uma avaliação comparativa usando os problemas-teste de Taillard (1993).

2.2 Máquinas paralelas

O problema de máquinas paralelas tem sido extensivamente estudado. Quando há somente um único recurso (máquina), ou um fluxo de produção unidirecional, a programação é determinada pelo **seqüenciamento** das atividades a serem executadas. No caso das máquinas paralelas, além da ordem de execução das tarefas, é necessário também definir a **alocação** dos recursos. Uma base ferramental sobre programação de operações em máquinas paralelas pode ser encontrada em Baker (1974), Morton e Pentico (1993) e Pinedo (1995). Os principais resultados de pesquisas nesta área foram apresentados no estado da arte publicado por Cheng e Sin (1990). Em seguida, serão descritos sucintamente os trabalhos mais recentes.

Piersma e VanDijk (1996) examinaram o problema de programação em máquinas paralelas **não-relacionadas**, com o critério de minimização do *makespan*. Algoritmos de busca local foram propostos utilizando o método de Busca Tabu. O mesmo ambiente foi estudado por Bank e Werner (2001), com o objetivo de minimizar a soma ponderada dos valores da antecipação relativa ao prazo de término (*earliness*) e as penalidades dos atrasos (*tardiness*). No problema, todas as tarefas possuíam a mesma data de entrega, porém diferentes datas de liberação. Vários algoritmos construtivos e melhorativos foram sugeridos e comparados.

Chiu, Fang e Lee (1999) propuseram um modelo de programação inteira mista para seqüenciamento de operações em máquinas paralelas. Um procedimento baseado no Algoritmo Genético e dois novos operadores genéticos foram apresentados para gerar soluções descendentes factíveis. A experimentação computacional mostrou que o algoritmo genético proposto é capaz de encontrar soluções eficientes e de alta qualidade.

Koulamas e Kyparisis (2000) consideraram o problema de máquinas paralelas **uniformes**, ou **proporcionais**, com o objetivo de minimizar o atraso máximo. O trabalho mostrou que uma extensão da regra EDD (*Earliest Due Date*) para máquinas paralelas resulta em um valor do atraso máximo que não excede a solução ótima em mais do que $p_{\max}$, onde $p_{\max}$ é o maior dos tempos de processamento das tarefas.

Sarin e Hariharan (2000) estudaram o problema bicritério de programar n tarefas em duas máquinas paralelas, com o objetivo de minimizar o atraso máximo, prioritariamente, e o número de tarefas em atraso. Algoritmos para minimizar cada um

dos critérios e propriedades de otimização do atraso máximo foram desenvolvidos. O procedimento para minimização do número de tarefas em atraso é baseado no método *Branch and Bound* e pode ser usado para resolver problemas com número de tarefas superior a 100.

Gupta e Ruiz-Torres (2000) enfocaram um problema de minimização do *makespan* sujeito a um tempo total de fluxo ótimo, em um ambiente de programação de máquinas paralelas **idênticas**. Em vista da natureza *NP-hard* do problema, limites inferiores para o valor do *makespan* ótimo e algoritmos heurísticos eficientes foram definidos e avaliados experimentalmente. Gupta e Ho (2001) propuseram um algoritmo para determinar a solução ótima do problema de programação em duas máquinas paralelas idênticas. A função-objetivo foi a minimização do *makespan* entre um conjunto de programações com tempo total de fluxo ótimo.

O problema de máquinas paralelas idênticas com o critério de minimização da antecipação máxima relativa ao prazo de término (*earliness* máximo) foi abordado por Mandel e Mosheiov (2001). Uma eficiente heurística baseada na regra MST (*Minimum Slack Time*) foi introduzida, assim como dois limitantes superiores para o valor ótimo da função-objetivo. Foi apresentado um extenso estudo numérico indicando que a heurística possui bom desempenho para problemas com várias configurações de tarefas e máquinas. Os resultados computacionais mostraram que a heurística gera valores próximos da solução ótima.

Liao e Lin (2003) estudaram o problema de programação com duas máquinas paralelas uniformes e critério de minimização do *makespan*. O problema pode ser transformado em um caso especial de duas máquinas paralelas idênticas, do ponto de vista da carga de trabalho ao invés de tempo de término. Apesar do algoritmo proposto possuir uma complexidade de tempo de computação exponencial, os resultados mostraram que ele pode fornecer solução ótima para problemas de grande porte com um esforço computacional relativamente baixo.

2.3 *Flow shop* com máquinas múltiplas

O ambiente *flow shop* com máquinas múltiplas pode ser visto como uma combinação do *flow shop* clássico e o problema de máquinas paralelas, que têm sido

intensivamente estudados. Na literatura, este problema também é conhecido como *flow shop* **flexível**, *flow shop* **híbrido** ou *flow shop* com **máquinas paralelas**.

Sriskandarajah e Sethi (1989) discutiram o desempenho de algoritmos de programação para um ambiente com dois centros de fabricação compostos por uma ou mais máquinas paralelas idênticas. As operações de cada tarefa podem ser processadas em qualquer máquina paralela do centro de trabalho. A função-objetivo era a minimização do *makespan*.

O caso restrito do problema *flow shop* com múltiplos processadores, caracterizado por uma limitação preestabelecida no número total de tarefas no sistema, foi abordado por Hunsucker e Shah (1995). Uma simulação experimental foi desenvolvida para investigar o comportamento de seis regras de prioridade e três medidas de desempenho para o ambiente descrito. Os resultados comprovaram que o procedimento *Shortest Processing Time* (SPT) obteve desempenho superior para os critérios *makespan* e tempo médio de fluxo.

Guinet e Solomon (1996) propuseram alguns algoritmos heurísticos para o problema de programação no ambiente *flow shop* híbrido com gargalos de produção, tendo como objetivo a minimização do atraso máximo ou do *makespan*. Foram definidos limitantes inferiores da solução ótima para avaliar o desempenho das heurísticas. As heurísticas possuem duas fases: na primeira, três algoritmos conhecidos (CDS, NEH e algoritmo de Townsend, 1977) são utilizados para criar uma lista de prioridades e na segunda, as tarefas são associadas às máquinas em cada centro de trabalho. Os resultados computacionais indicaram que a heurística baseada no algoritmo NEH forneceu os melhores resultados entre os métodos testados.

A classe de problemas *flow shops* paralelos proporcionais foi enfocada por Sundararaghavan, Kunnathur e Viswanathan (1997). O problema consiste em um ambiente *flow shop* em que as tarefas podem ser processadas em um *flow shop* mais lento ou em outro mais rápido. O tempo de processamento no primeiro é múltiplo do tempo de processamento no segundo. O objetivo é designar as tarefas aos *flow shops* de modo que o *makespan* seja minimizado. Para problemas de pequeno porte (com 15 tarefas), os resultados foram comparados com soluções exatas e para problemas com 30 e 50 tarefas, com um limite inferior fornecido por um modelo de programação linear.

O problema *flow shop* híbrido com dois estágios e função-objetivo de minimizar o número total de tarefas em atraso foi examinado por Gupta e Tunc (1998). O caso estudado contém somente uma máquina no primeiro estágio e *m* máquinas paralelas

idênticas no segundo. Como o problema é classificado fortemente como *NP-hard*, vários algoritmos heurísticos foram desenvolvidos para encontrar a melhor programação. O ambiente com dois estágios também foi estudado por Lee e Vairaktarakis (1994), Haouari e M'Hallah (1997), Li (1997) e Kim, Kang e Lee (1997).

Riane, Artiba e Elmaghraby (1998) analisaram um problema de seqüenciamento em um ambiente *flow shop* híbrido com três estágios, sendo uma máquina no primeiro e no terceiro estágios e o segundo contendo duas máquinas dedicadas, ou seja, com designação de tarefas pré-estabelecida. O objetivo era a minimização do *makespan*. O primeiro método proposto para resolução deste problema foi baseado na programação dinâmica e o segundo combinou um processo construtivo e um limitante superior. Este último baseou-se no método *Branch and Bound*.

Linn e Zhang (1999) propuseram uma classificação das pesquisas em *flow shop* híbrido em três categorias: problemas com dois estágios, três estágios e *K* estágios ($K > 3$). Os autores também discutiram a complexidade, o critério de desempenho e o método de solução dos problemas e citaram a distância existente entre a teoria de programação da produção e as aplicações práticas.

Vignier, Billaut e Proust (1999) apresentaram para aquele momento o estado da arte de problemas *flow shops* híbridos. A proposta, organizada em duas partes, inclui uma notação para identificar um problema precisa e rapidamente. A primeira parte é dedicada a *flow shops* híbridos com dois estágios e a segunda, ao caso geral com *K* estágios.

Um algoritmo *Branch and Bound* usando um novo esquema de ramificação foi introduzido por Moursli e Pochet (2000) para resolver o problema com *K* estágios de produção. Várias estratégias de limitantes superiores e inferiores foram consideradas, tanto para acelerar a busca da solução na árvore (*branching*) como para melhorar a sua qualidade. Quando há um estágio gargalo, o método mostrou-se convergente e o algoritmo fornece boa solução para problemas de 2 a 5 máquinas e de 10 a 20 tarefas, a partir do limite superior.

Seis heurísticas foram introduzidas por Botta-Genoulaz (2000) para resolver o problema de minimização do atraso máximo em *flow shop* híbrido com *K* estágios compostos por máquinas paralelas idênticas. O problema era sujeito às restrições de precedência em árvore *out-tree* (cada tarefa tem no máximo uma predecessora) e mínimo tempo de transporte (*time lag*).

Riane, Artiba e Elmaghraby (2002) estudaram o problema com dois estágios sendo uma máquina no primeiro estágio e duas máquinas paralelas não-relacionadas no segundo. A função-objetivo era a minimização do *makespan*. Foi demonstrado que o problema é *NP-completo*. Três métodos de solução foram propostos: o primeiro é baseado em programação dinâmica, o segundo é uma heurística *sequence-and-merge* e o terceiro é uma heurística "gulosa" (*greedy*). Os dois primeiros são baseados no algoritmo de Johnson (1954). Os resultados das heurísticas foram comparados com os do método CDS e com a solução ótima fornecida por um modelo de programação linear mista e por programação dinâmica. Foram testados problemas de seis tamanhos de 20 a 120 tarefas (passos de 20 em 20). Quatro relações nos tempos médios de processamento, denotado por $\overline{p}_m$, das tarefas em cada uma das três máquinas (m = 1, 2, 3) envolvem todas as possibilidades de relacionamento entre os tempos de processamento: i) Tempos médios iguais: $\overline{p}_1 = \overline{p}_2 = \overline{p}_3$; ii) Estágio 1 dominado: $\overline{p}_1 \leq \min(\overline{p}_2, \overline{p}_3)$; iii) Estágio 1 intermediário: $\overline{p}_2 \leq \overline{p}_1 \overline{p}_3$; iv) Estágio 1 dominante: $\overline{p}_1 \geq \max(\overline{p}_2, \overline{p}_3)$. A comparação dos resultados fornecidos com o limite inferior estabelecido mostrou a grande proximidade com a solução ótima.

Moccellin e Nagano (2003a) trataram do problema de programação *flow shop* híbrido com a existência de um estágio gargalo, no qual o número de máquinas paralelas é menor que nos demais. Foi considerada como função-objetivo a minimização do *makespan*. Moccellin e Nagano (2003b) estudaram o problema com duas máquinas paralelas genéricas em cada estágio de produção, tendo como objetivo minimizar o *makespan*. Foram propostos métodos heurísticos alternativos com o objetivo de avaliar a eficácia de um movimento condicional de tarefas utilizando um limitante inferior para o tempo de espera das tarefas entre o final de sua operação no estágio k e o início no estágio (k +1).

Oguz *et al.* (2003) estudaram o problema *flow shop* híbrido com multiprocessamento de tarefas, que relaxam a limitação do modelo clássico de máquinas paralelas permitindo que as operações sejam processadas simultaneamente. Os autores apresentaram algoritmos construtivos para a minimização do *makespan* em problemas com dois estágios. Foram consideradas regras de seqüenciamento de tarefas conhecidas na literatura, como SPT e LPT, e também regras compostas. Alguns limites inferiores para o problema foram estabelecidos para a análise do desempenho dos algoritmos. Os resultados sugerem que os algoritmos propostos são eficazes e eficientes.

Para o mesmo ambiente, Oguz *et al.* (2004) mostraram que, sem restrições de precedência e assumindo que todos os tempos de processamento das tarefas tenham um limite superior, a minimização do *makespan* é alcançada em tempo polinomial, enquanto a introdução de restrições de precedência torna o problema *NP-hard* mesmo na sua mais simples versão. Algumas das restrições estudadas foram: (1,2)-*jobs*, em que cada tarefa necessita de um processador no estágio 1 e dois processadores no estágio 2, e (2,1)-*jobs*, analogamente. Foi apresentado um algoritmo que melhora uma solução inicial, baseado na idéia da Busca Tabu, cujos resultados indicam que o desempenho do método não depende da distribuição dos tempos de processamento das tarefas.

2.4 *Flow shop* permutacional com tempos de preparação dependentes da seqüência

A pesquisa em problemas com tempos de *setup* dependentes da seqüência iniciou com os modelos desenvolvidos para máquina única, destacando o trabalho de Gavett (1965), em que três heurísticas construtivas (*NB*, *NB'* e *NB"*) são desenvolvidas a partir do Problema do Caixeiro Viajante com o objetivo de minimizar o *makespan*.

Simons Jr. (1992) propôs quatro métodos heurísticos construtivos, denominados MINIT, MINICOT, TOTAL e SETUP. As duas primeiras heurísticas são aplicadas à última máquina do sistema e a seqüência obtida é considerada para as demais. TOTAL e SETUP consistem na aplicação da Heurística de Stinson (STINSON e SMITH, 1982) para as matrizes dos tempos de *setup* das tarefas do *flow shop*. Os resultados foram comparados com os de métodos de pesquisas anteriores. As heurísticas TOTAL e SETUP forneceram os melhores resultados em todos os critérios de comparação (como o número de vezes que o método forneceu melhor solução, desempenho por tamanho de problema, desvio-padrão do índice de desempenho). E em alguns casos, a heurística TOTAL superou a SETUP.

Das, Gupta e Khumawala (1995) desenvolveram um algoritmo heurístico construtivo com um índice que determina as economias no tempo associado a uma seqüência particular. Tanto a eficácia como a eficiência do algoritmo foram demonstrados através das experimentações computacionais. Com exceção de quatro problemas, os resultados fornecidos pelos 240 problemas analisados (variando o número

de tarefas de 4 a 20 e o número de máquinas de 2 a 20) ficaram a uma margem média menor que 6% da solução ótima. O algoritmo também foi testado com problemas de grande porte (acima de 50 tarefas e 50 máquinas), fornecendo a solução em tempos computacionais relativamente pequenos.

Parthasarathy e Rajendran (1997) apresentaram um estudo de caso de uma indústria com o ambiente *flow shop* com tempos de *setup* dependentes da seqüência de execução das tarefas. A função-objetivo foi a minimização do atraso médio ponderado. Foi desenvolvido um algoritmo heurístico baseado no método *Simulated Annealing* e apresentado um novo esquema de geração da vizinhança chamado *Random Insertion Perturbation Scheme* (RIPS) com o objetivo de reduzir o atraso médio total.

Rajendran e Ziegler (1997) propuseram uma heurística para minimizar a soma dos tempos de fluxo ponderados das tarefas. Um esquema de melhoria foi utilizado para aumentar a qualidade da solução heurística. Os resultados demonstraram rapidez computacional e solução eficaz. Uma interessante observação é que a heurística proposta também forneceu melhor solução para o critério de minimização da soma ponderada das datas de término.

Duas heurísticas com o objetivo de minimizar o *makespan* foram apresentadas por Ríos-Mercado e Bard (1998). A primeira é uma versão modificada da heurística NEH e a outra, denominada *Greedy Randomized Adaptive Search Procedure* (GRASP), é um método heurístico para problemas de otimização combinatorial. Ambos os algoritmos foram comparados com as heurísticas de Simons Jr. (1992) – TOTAL e SETUP. Um procedimento de busca local e um esquema de limitante inferior com duas fases também foram apresentados.

Ríos-Mercado e Bard (1999) propuseram um método heurístico melhorativo para minimização do *makespan*. O procedimento considera o ambiente como um Problema do Caixeiro Viajante, introduzindo uma função de custo que penaliza programações com grandes tempos de *setup* e altos valores do critério de desempenho. Esta função de custo híbrida é uma melhoria de métodos anteriores que penalizam somente o tempo de *setup*, ignorando o aspecto de fluxo do problema. Foi apresentada uma versão melhorada das heurísticas de Simons Jr. (1992), denominada HYBRID, com a correção de algumas imperfeições e adicionando uma fase de busca local. A heurística HYBRID possui melhor desempenho do que a GRASP quando o número de máquinas é pequeno ou quando a variação no tempo de *setup* é grande.

O problema de programação *flow shop* permutacional com duas máquinas e agrupamento de tarefas foi estudado por Yang e Chern (2000). Em cada grupo, as tarefas são processadas sucessivamente, despendendo tempo de *setup* e de liberação em ambas as máquinas. Também é necessário um tempo de transporte para mover as tarefas da primeira máquina para a segunda. Um algoritmo polinomial é proposto para resolver o problema, cujo objetivo é a minimização do *makespan.*

Rajendran e Ziegler (2003) enfocaram o problema de programação *flow shop* com o critério de minimização da soma do tempo de fluxo ponderado e do atraso ponderado das tarefas. Dados os tempos de *setup* dependentes da seqüência, foram sugeridas duas heurísticas para construir boas soluções. Um esquema de melhoria foi implementado, aumentando a qualidade da solução. A heurística de Gelders e Sambandam (1978) e os procedimentos de busca randômica e busca local "gulosa" (*greedy*) foram usados para avaliar as heurísticas propostas. A análise do desempenho revelou que o método de Rajendran e Ziegler (2003) é mais rápido computacionalmente e mais efetivo na busca e na qualidade da solução.

Barros e Moccellin (2003) apresentaram uma metaheurística *Simulated Annealing* que analisa a possível mudança no gargalo do sistema decorrente da ordenação das tarefas. A medida de desempenho utilizada foi o *makespan.* O método proposto foi comparado com o algoritmo TOTAL, de Simons Jr. (1992), apresentando uma superioridade relevante.

Reddy e Narendran (2003) propuseram heurísticas para programar um fluxo linear de tarefas que compõem uma família em células de manufatura, com o objetivo de aumentar a utilização das máquinas e reduzir o tempo de atraso e o número de tarefas em atraso. A configuração do problema considera os tempos de *setup* dependentes da seqüência de processamento das famílias em cada célula. As programações permutacionais mantêm a mesma ordem de execução entre as famílias e entre as tarefas em cada estágio.

Ruiz, Maroto e Alcaraz (2004) consideraram a minimização do *makespan* aplicando um algoritmo genético avançado e uma versão híbrida. O procedimento incorporou uma inicialização especial de população e um novo esquema de geração de soluções descendentes que evita convergência prematura. Foram usados quatro conjuntos de 120 problemas cada, baseados nos problemas-teste de Taillard (1993), com a adição dos tempos de *setup* dependentes da seqüência. Várias adaptações foram feitas em metaheurísticas já existentes como a *Simulated Annealing* de Osman e Potts (1989),

a Busca Tabu de Widmer e Hertz (1989), o Algoritmo Genético de Reeves (1995) e também o recente algoritmo para *no-wait flow shop* de Aldowaisan e Allahverdi (2003). Foram codificados vários métodos específicos para *flow shop* com tempos de *setup* dependentes da seqüência como as heurísticas SETUP e TOTAL de Simons Jr. (1992) e a metaheurística GRASP de Ríos-Mercado e Bard (1998). Os resultados obtidos mostraram que em todos os tipos e tamanhos de problemas analisados, os algoritmos genéticos propostos forneceram melhor desempenho.

2.5 Máquinas paralelas com tempos de preparação dependentes da seqüência

O problema de programação de tarefas em máquinas paralelas idênticas com tempos de *setup* dependentes da seqüência foi estudado por Schutten (1996), Lee e Pinedo (1997) e Balakrishnan, Kanet e Sridharan (1999).

Lancia (2000) enfocou o problema de alocação de tarefas com *release dates* e *tails* (tempo gasto por uma tarefa em uma máquina após o seu processamento) em duas máquinas paralelas não-relacionadas. O objetivo foi a minimização do *makespan*. Um procedimento baseado no método *Branch and Bound* foi descrito para a solução deste problema.

O ambiente de máquinas paralelas não-relacionadas foi analisado por Weng, Lu e Ren (2001). O critério utilizado para otimização foi a minimização da data média de término ponderada (denotada por $\sum_{i=1}^{n} \frac{1}{n} w_i C_i$). O estudo é proveniente de um problema industrial real. Sete heurísticas foram propostas e testadas com simulação. Foi observado que as diferenças entre o desempenho dos algoritmos diminui com a redução da amplitude dos tempos de processamento e tempos de *setup*.

Kravchenko e Werner (2001) consideraram o problema de máquinas paralelas idênticas com o objetivo de minimizar a soma das datas de término ($\sum C_i$), que equivale à minimização do tempo médio de fluxo ($\overline{F}$), para o caso de tempos de *setup* unitários e tempos de processamento arbitrários. O problema foi classificado como fortemente *NP-hard*. Um algoritmo heurístico foi apresentado com um erro absoluto limitado pelo produto do número de tarefas com tempo de processamento menor que (m-1) e (m-2).

Hurink e Knust (2001) estudaram o problema com restrições de precedência e tempos de *setup* dependentes da seqüência de tarefas com o critério de minimização do *makespan*. As restrições representam uma ordenação linear que define qual a próxima tarefa a ser realizada imediatamente após a atual. O trabalho aplicou a técnica de programação de lista, que representa basicamente um procedimento que, para uma dada ordenação de tarefas (uma "lista"), fornece uma programação correspondente. Em geral, este procedimento considera cada tarefa individualmente em uma ordenação e forma uma programação parcial com base nas tarefas previamente programadas.

Kim *et al.* (2002) sugeriram uma técnica baseada no método *Simulated Annealing* para minimizar o atraso total em processamento de lotes. Os tempos de *setup* não dependiam da máquina mas unicamente da seqüência de processamento das tarefas. O método sugerido utilizou seis técnicas para gerar a vizinhança das soluções.

Kim, Na e Chen (2003) apresentaram várias heurísticas de busca para programação de máquinas paralelas não-relacionadas. O processamento foi feito em lotes (*batches*) de tarefas iguais ou similares com o objetivo de reduzir o tempo de *setup*. Todas as tarefas do mesmo lote possuíam tempos de processamento iguais e o mesmo prazo de entrega. O *setup* era requerido entre os diferentes lotes mas não dependiam das máquinas. O objetivo do estudo foi minimizar o *tardiness* total ponderado, representado por $\sum_{i=1}^{B} w_i \max\{C_i - d_i, 0\}$, onde B é o número de lotes.

2.6 *Flow shop* com máquinas múltiplas e tempos de preparação dependentes da seqüência

Huang e Li (1998) apresentaram um ambiente *flow shop* híbrido com dois estágios e tempos de *setup* dependentes da seqüência de famílias de produtos, baseado em um problema real. O primeiro estágio continha somente uma máquina e o segundo, várias máquinas uniformes. O objetivo foi a minimização do *makespan*. Para a solução do problema, foram construídas duas heurísticas e definidas oito regras de seqüenciamento. Também foi apresentado um modelo matemático de programação linear considerando o *trade-off* entre o custo de máquinas rápidas e o *makespan*.

Um estudo de caso para o problema *flow shop* híbrido com dois estágios e tempos de *setup* dependentes da seqüência somente no primeiro estágio, grupos de máquinas idênticas no segundo estágio e dois prazos de entrega (*due dates*) foi realizado por Lin e Liao (2003). O objetivo foi minimizar o atraso máximo ponderado. O resultado da heurística proposta foi comparado com a solução ótima de problemas de pequeno porte (7 a 10 tarefas) e com a solução do método de programação utilizado na fábrica. A heurística ordena as tarefas no primeiro estágio minimizando o tempo total de *setup* e emprega a regra *First In First Out* (FIFO) no segundo estágio.

Ruiz e Maroto (2003) construíram uma heurística, baseada no algoritmo genético, para problemas complexos de programação *flow shop* híbrido. O procedimento resulta da composição de máquinas paralelas não-relacionadas em cada estágio, tempos de preparação dependentes da seqüência e restrições de elegibilidade. Como nem todos os produtos podem ser processados em todas as máquinas disponíveis existe a **elegibilidade de máquina**. O número de estágios varia de dois, nos sistemas de produção mais rígidos, a cinco ou mais, nas grandes empresas. O número de máquinas paralelas não-relacionadas a cada estágio é de um a três. Este problema é comum em produção de tecidos e telhas de cerâmica. O experimento computacional indicou superioridade entre 53% e 135% do algoritmo proposto com relação a adaptações de metaheurísticas conhecidas. O algoritmo também foi testado com dados reais e os resultados mostraram uma redução no *makespan* de quase 9% em relação aos métodos utilizados na empresa objeto do estudo de caso.

É este o ambiente de produção tratado neste trabalho. Como pode ser observado pelo exame da literatura, existem poucos trabalhos reportados até o momento.

3 OS MÉTODOS HEURÍSTICOS CONSTRUTIVOS PROPOSTOS

3.1 Definição do problema

O problema tratado neste trabalho integra uma linha de pesquisa do Departamento de Engenharia de Produção da Escola de Engenharia de São Carlos – USP denominada "Pesquisa Operacional Aplicada aos Sistemas de Produção". O principal tema de pesquisa no Programa de Pós-Graduação em Engenharia de Produção para esta linha foca a "Programação da Produção em Sistemas de Produção Intermitentes". Este tema consiste no estudo de modebs e desenvolvimento de novos métodos de solução para problemas de programação de operações em máquinas.

O problema pode ser definido como:

1) *Flow shop* híbrido composto por múltiplos estágios de produção, ou seja, $K \quad 2$;

2) Em cada estágio k, existem M_k máquinas paralelas idênticas, onde $M_k \quad 2$;

3) Os tempos de preparação das máquinas são assimétricos e dependentes da seqüência de execução das tarefas.

O ambiente descrito é ilustrado na Figura 3.1.

As principais hipóteses consideradas no problema são as seguintes:

1) Os tempos de processamento das tarefas nas diversas máquinas são determinados e fixos;

2) As tarefas têm a mesma data de liberação, a partir da qual qualquer uma pode ser programada e executada. Esta data de liberação pode ser considerada igual a zero, sem perda de generalidade;

3) Uma vez iniciadas, as operações de cada tarefa não podem ser interrompidas nem subdivididas em sub-operações simultâneas;

4) Uma tarefa só pode começar a ser executada em uma máquina após a execução completa da sua operação no estágio anterior e desde que a máquina já esteja preparada;

5) O *setup* de uma máquina, para determinada tarefa, pode ser executado antes da operação dessa tarefa estar concluída no estágio anterior e considera-se que o *setup* da primeira operação em cada máquina já esteja realizado (no problema, o tempo de *setup* da primeira operação é considerado igual a zero).

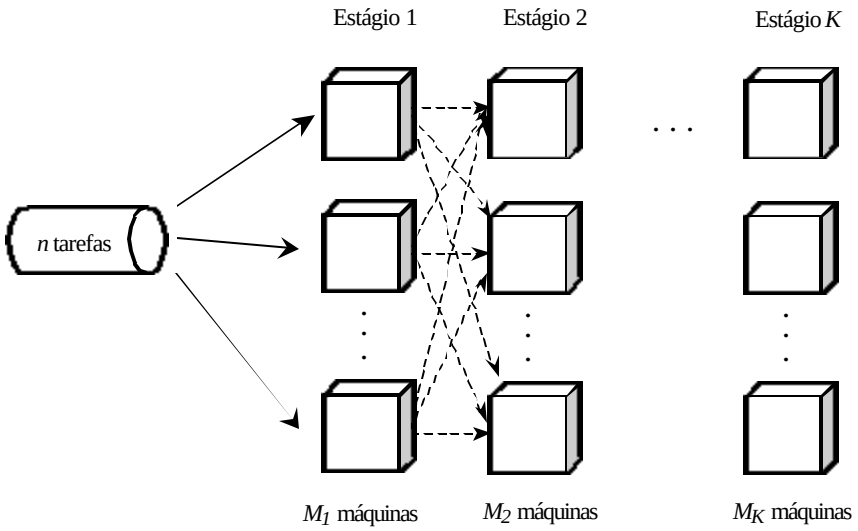


FIGURA 3.1 – Ilustração do ambiente de produção

O problema consiste em programar um conjunto de n tarefas, definido como $J = \{J_1, ..., J_n\}$, onde cada tarefa possui necessariamente uma única operação em cada estágio de produção. As tarefas devem ser realizadas seqüencialmente e suas operações passam por todos os estágios. O objetivo do problema é minimizar o *makespan* como medida de desempenho.

3.2 Proposição de Métodos Heurísticos Construtivos

Para obter a solução do problema descrito anteriormente, foram desenvolvidos e avaliados quatro **métodos heurísticos construtivos** com base em algoritmos reportados

na literatura para solução de problemas *flow shop* permutacional e máquinas paralelas no ambiente cujo tempo de *setup* é dependente da seqüência.

Foram definidos procedimentos para programação das tarefas estágio a estágio, como solução iterativa de K problemas relacionados. O primeiro estágio ($k = 1$) é programado como se fosse um problema tradicional de M_1 máquinas paralelas idênticas com a mesma data de liberação, convencionada igual a zero ($r_i = r = 0$). Os estágios seguintes ($k \geq 2$) consistem de $K - 1$ problemas consecutivos de M_k máquinas paralelas idênticas com as seguintes particularidades:

- tempos de preparação das máquinas assimétricos e dependentes da seqüência de processamento das tarefas; e
- diferentes datas de liberação das tarefas, correspondentes às datas de término das operações das respectivas tarefas no estágio imediatamente anterior ($k - 1$).

Os algoritmos dos quatro métodos de solução – dois procedimentos e duas regras que cada um utiliza – são detalhados a seguir.

Procedimento 1 – estabelecimento de ordenação inicial

O Procedimento 1 define uma ordenação inicial de tarefas e utiliza duas regras de prioridade: a *Longest Processing Time* (LPT) e um método baseado no algoritmo TOTAL de Simons Jr. (1992). Desta forma, as regras de prioridade deste procedimento são denominadas LPT e TOTAL, respectivamente.

A seguinte expressão, para o cálculo das datas de término das tarefas J_i em um determinado estágio k, foi adaptada do trabalho de Ruiz e Maroto (2003):

$$C_{E_{ik}} = \min_{m=1}^{M_k} \left\{ \max \left\{ C_{u_m k} + s_{u_m ik} ; C_{i,k-1} \right\} + p_{ik} \right\} \tag{3.1}$$

com $i = 1, ..., n$ e $k = 1, ..., K$.

$C_{E_{ik}}$ representa a menor data de término da tarefa J_i no estágio k, M_k é o número de máquinas no estágio k, u_m é o índice da última tarefa alocada na máquina m (J_{u_m}), $C_{u_m k}$ denota a data de término da tarefa J_{u_m} no estágio k, $s_{u_m ik}$ é o tempo de

setup da tarefa J_i no estágio k após o processamento de J_{u_m}, $C_{i,k-1}$ denota a data de término da tarefa J_i no estágio k-1 e p_{ik} é o tempo de processamento da tarefa J_i no estágio k.

Regra de Prioridade LPT

Para estabelecer a ordenação inicial no Procedimento 1, a regra de prioridade LPT considera a soma dos tempos de processamento de cada tarefa em todos os estágios. O seqüenciamento das tarefas é feito pela ordem não-crescente destas somas.

Regra de Prioridade TOTAL

Esta regra de prioridade baseia-se na heurística TOTAL de Simons Jr. (1992) para programação de *flow shop* permutacional com tempos de *setup* dependentes da seqüência. Por sua vez, a heurística TOTAL utiliza o Método de Aproximação de Vogel, conhecido por fornecer boas soluções iniciais para problemas de transporte (SHORE, 1970).

O Método de Aproximação de Vogel (MAV) seleciona células em uma matriz de transporte examinando o aumento da diferença entre os dois menores valores de cada linha e coluna. A célula selecionada em cada iteração é aquela que produz a maior diferença em comparação com a célula de menor valor na linha ou coluna. Este método é aplicado ao Problema do Caixeiro Viajante onde cada combinação linha/coluna representa uma subseqüência possível (por exemplo, linha 5, coluna 3 representa a seqüência parcial 5 – 3). Então, para o problema *flow shop* permutacional, cada célula representa um possível par consecutivo de tarefas e cada iteração seleciona uma seqüência parcial de duas tarefas (SIMONS JR., 1992).

O algoritmo TOTAL aplica o MAV em uma matriz $n \times n$ composta pela soma dos tempos de processamento e de *setup* em todas as máquinas. Neste trabalho, a matriz utilizada na regra de ordenação inicial TOTAL é formada pela soma dos tempos de processamento e de *setup* de todos os estágios de produção.

O algoritmo do MAV, adaptado de Reinfeld e Vogel (1958), é apresentado a seguir.

Método de Aproximação de Vogel

PASSO 1

Calcule a diferença entre os dois menores elementos de cada linha e cada coluna da matriz formada pela soma dos tempos de processamento e de *setup* de cada tarefa.

PASSO 2

Identifique a linha ou coluna com a maior diferença (desempates são arbitrários).

PASSO 3

Selecione o elemento da matriz com menor valor correspondente à linha ou coluna com a maior diferença.

PASSO 4

Se a seqüência de tarefas está completa, PARE.

Caso contrário, vá para o PASSO 5.

PASSO 5

Recalcule as diferenças entre os dois menores elementos de cada linha e cada coluna, desconsiderando os elementos correspondentes à subseqüência programada, ou seja, a linha referente à primeira tarefa da subseqüência, a coluna referente à segunda tarefa da subseqüência e o elemento correspondente ao inverso da subseqüência (por exemplo, se a subseqüência é $J_5 - J_3$, desconsiderar a linha 5, a coluna 3 e a célula correspondente a $J_3 - J_5$). Linhas e colunas somente com elemento zero também devem ser desconsideradas. Vá para o PASSO 2.

Exemplo numérico:

Considere-se a seguinte matriz, onde cada elemento é a soma dos tempos de processamento (em todos os estágios) e de *setup* de cada tarefa:

	J_1	J_2	J_3	J_4
J_1	-	12	25	18
J_2	26	-	33	36
J_3	17	13	-	31
J_4	29	28	11	-

PASSO 1 (1ª ITERAÇÃO)

	J_1	J_2	J_3	J_4	
J_1	-	12	25	18	**6**
J_2	26	-	33	36	**7**
J_3	17	13	-	31	**4**
J_4	29	28	11	-	**17**
	9	**1**	**14**	**13**	

PASSO 2

	J_1	J_2	J_3	J_4	
J_1	-	12	25	18	**6**
J_2	26	-	33	36	**7**
J_3	17	13	-	31	**4**
J_4	29	28	11	-	**(17)**
	9	**1**	**14**	**13**	

PASSO 3

Menor elemento da linha: 11

Subseqüência: $J_4 - J_3$

PASSO 4

$J_4 - J_3$: Seqüência incompleta

PASSO 5

	J_1	J_2	J_3	J_4	
J_1	-	12	--	18	**6**
J_2	26	-	--	36	**10**
J_3	17	13	--	31	**4**
J_4	--	--	--	--	--
	9	**1**	--	**13**	

PASSO 2 (2ª ITERAÇÃO)

	J_1	J_2	J_3	J_4	
J_1	-	12	--	18	**6**
J_2	26	-	--	36	**10**
J_3	17	13	--	31	**4**
J_4	--	--	--	--	--
	9	**1**	--	**(13)**	

PASSO 3

Menor elemento da coluna: 18

Subseqüência: $J_1 - J_4$

PASSO 4

$J_1 - J_4 - J_3$: Seqüência incompleta

PASSO 5

	J_1	J_2	J_3	J_4	
J_1	---	---	---	---	---
J_2	26	-	--	---	**26**
J_3	17	13	--	---	**4**
J_4	--	--	--	---	--
	9	**13**	--	---	

PASSO 2 (3ª ITERAÇÃO)

	J_1	J_2	J_3	J_4	
J_1	---	---	---	---	---
J_2	26	-	--	---	**(26)**
J_3	17	13	--	---	**4**
J_4	--	--	--	---	--
	9	**13**	--	---	

PASSO 3

Menor elemento da linha: 26

Subseqüência: $J_2 - J_1$

PASSO 4

$J_2 - J_1 - J_4 - J_3$

Seqüência completa: **PARE**

ALGORITMO DO PROCEDIMENTO 1 – REGRAS LPT E TOTAL

Seja J um conjunto de n tarefas a serem programadas, J' o conjunto das tarefas ainda não programadas, $J_{[j]}$ a tarefa que ocupa a j-ésima posição de J', M o número total de máquinas considerando todos os estágios de produção ($M = \sum_{k=1}^{K} M_k$, onde M_k é o número de máquinas no estágio k) e σ_m o subconjunto de tarefas alocadas à máquina m. Seja r_i a data de liberação da tarefa J_i e SRD (*Shortest Release Date*) a ordenação das tarefas segundo os valores não-decrescentes de r_i.

PASSO 1 (INICIALIZAÇÃO)

$\sigma_m = \varnothing$, para $m = 1, 2, ..., M$

$J' \leftarrow J$

Ordene as tarefas do conjunto J' de acordo com a regra de prioridade (LPT ou TOTAL). Vá para o PASSO 3.

PASSO 2

$J' \leftarrow J$

Ordene as tarefas do conjunto J' de acordo com a ordenação SRD, considerando as datas de término das tarefas no estágio anterior como as datas de liberação do estágio atual.

PASSO 3

A primeira tarefa de J' será alocada à máquina x, cuja programação possuirá a data mais cedo de término.

PASSO 4 (ATUALIZAÇÃO DOS CONJUNTOS E TESTE DE PARADA)

$\sigma_x \leftarrow \sigma_x \cup \{ J'_{[1]} \}$

$J' \leftarrow J' - \{ J'_{[1]} \}$

Se $J' \neq \varnothing$, vá para o PASSO 3.

Caso contrário, programação do estágio concluída.

Se não for o último estágio, vá para o PASSO 2.

Caso contrário PARE, programação concluída. Saída: σ_m para $m = 1, 2... M$.

Procedimento 2 – sem estabelecimento de ordenação inicial

O Procedimento 2 não utiliza ordenação inicial para as tarefas e uma das regras de alocação emprega uma adaptação do melhor entre sete algoritmos para máquinas paralelas com *setup* dependente apresentados por Weng, Lu e Ren (2001). A outra regra de alocação deste procedimento também utiliza a adaptação do algoritmo citado, mas quando todas as tarefas estiverem liberadas respeita a ordenação pela regra LPST (*Longest Processing-Setup Time*), envolvendo a soma do tempo de processamento com o tempo de *setup*. As regras de alocação deste procedimento são designadas SCT (*Shortest Completion Time*) e SCT/LPST, respectivamente.

O algoritmo de Weng, Lu e Ren (2001) associa uma tarefa de cada vez e escolhe o par tarefa-máquina com a data de término mais cedo. Da mesma forma que o procedimento 1, o procedimento 2 é aplicado sucessivamente em todos os estágios, considerando as datas de liberação das tarefas em cada estágio.

Regra de Alocação SCT

Como não há estabelecimento de uma ordenação inicial de tarefas, a alocação é feita por meio de uma expressão que analisa todas as possibilidades de combinação tarefa-máquina e escolhe o par com a data de término mais cedo. Em cada estágio k, a seguinte expressão é calculada:

$$C_{E_{ik}} = \min\left\{C_{u_m k} + s_{u_m ik} + p_{ik} \mid m \in m_k\right\} \tag{3.2}$$

com $i = 1,..., n$ e $k = 1, ..., K$, onde m_k é o conjunto dos índices das máquinas do estágio k e as variáveis são as mesmas da expressão 3.1.

Regra de Alocação SCT/LPST

Esta regra de alocação segue o mesmo procedimento da regra SCT, porém quando todas as tarefas estiverem liberadas, respeita a ordenação pela regra LPST.

Considerando a máquina de menor carga, a tarefa alocada é aquela com o maior valor da soma dos tempos de *setup* e de processamento ($s_{u_m ik} + p_{ik}$).

ALGORITMO DO PROCEDIMENTO 2 – REGRA SCT

PASSO 1 (INICIALIZAÇÃO)

$\sigma_m = \varnothing$, para $m = 1, 2, ..., M$

PASSO 2

$J' \leftarrow J$

PASSO 3

A partir da primeira data com tarefas liberadas, calcule a data de término de cada uma das tarefas de J' já liberadas, em cada uma das máquinas do estágio, ou seja, analise todas as possibilidades de associação tarefa-máquina, e escolha o par tarefa J_i e máquina x com a data de término mais cedo.

PASSO 4 (ATUALIZAÇÃO DOS CONJUNTOS E TESTE DE PARADA)

$\sigma_x \leftarrow \sigma_x \cup \{J_i\}$

$J' \leftarrow J' - \{J_i\}$

Se $J' \neq \varnothing$, vá para o PASSO 3.

Caso contrário, programação do estágio concluída.

Se não for o último estágio, vá para o PASSO 2.

Caso contrário PARE, programação concluída. Saída: σ_m para $m = 1, 2... M$.

ALGORITMO DO PROCEDIMENTO 2 – REGRA SCT/LPST

PASSO 1 (INICIALIZAÇÃO)

$\sigma_m = \varnothing$, para $m = 1, 2, ..., M$

PASSO 2

$J' \leftarrow J$

PASSO 3

Se todas as tarefas estiverem liberadas, vá para o PASSO 4.

Caso contrário, a partir da primeira data com tarefas liberadas, calcule a data de término de cada uma das tarefas de J' já liberadas, em cada uma das máquinas do estágio, ou seja, analise todas as possibilidades de associação tarefa-máquina, e escolha o par tarefa J_i e máquina x com a data de término mais cedo. Vá para o PASSO 5.

PASSO 4

Na máquina de menor carga, associe a tarefa J_i com o maior valor de $(s_{u_m ik} + p_{ik})$, ou seja, observe a regra LPST.

PASSO 5 (ATUALIZAÇÃO DOS CONJUNTOS E TESTE DE PARADA)

$\sigma_x \leftarrow \sigma_x \cup \{J_i\}$

$J' \leftarrow J' - \{J_i\}$

Se $J' \neq \varnothing$, vá para o PASSO 3.

Caso contrário, programação do estágio concluída.

Se não é o último estágio, vá para o PASSO 2.

Caso, contrário PARE, programação concluída. Saída: σ_m para $m = 1, 2... M$.

Observação: nos algoritmos dos dois procedimentos, considera-se que uma máquina qualquer, ao receber a primeira tarefa, já esteja preparada para sua execução.

4 EXPERIMENTAÇÃO COMPUTACIONAL

4.1 Delineamento do experimento

4.1.1 Relações entre as ordens de grandeza dos tempos de processamento e de *setup*

O foco da experimentação computacional foi a análise da influência da relação entre as ordens de grandeza dos tempos de processamento e de *setup* das tarefas, denotada por $O(p_i)/O(s_{ij})$, no desempenho dos métodos desenvolvidos. Por este motivo, foram definidas seis relações:

Relação I: $O(p_i)/O(s_{ij}) = 1$

Os tempos de processamento e de *setup* possuem a mesma ordem de grandeza, ou seja, foram definidos no mesmo intervalo, conforme ilustra a figura 4.1.

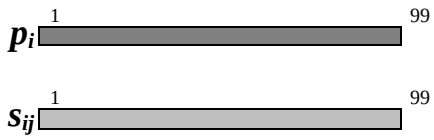


FIGURA 4.1 – Ilustração da relação I

Relação II: $O(p_i)/O(s_{ij}) < 1$

As ordens de grandeza dos tempos de processamento e de *setup* não são comuns, ou seja, os tempos foram definidos em intervalos distintos. Nesta relação, os valores dos tempos de processamento são sempre menores do que os tempos de *setup*. A figura 4.2 ilustra este caso.

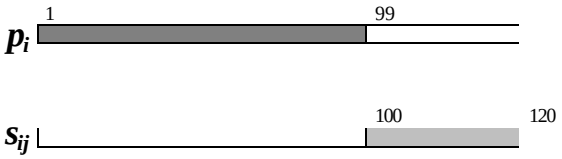


FIGURA 4.2 – Ilustração da relação II

Relações III e IV: $O(p_i)/O(s_{ij}) > 1$

Foram definidas duas alternativas cujas ordens de grandeza não são comuns e os tempos de *setup* são menores do que os tempos de processamento. As relações III e IV são ilustradas nas figuras 4.3 e 4.4, respectivamente.

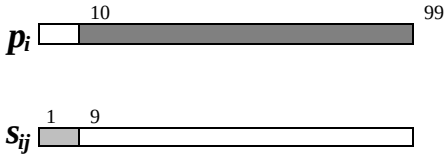


FIGURA 4.3 – Ilustração da relação III

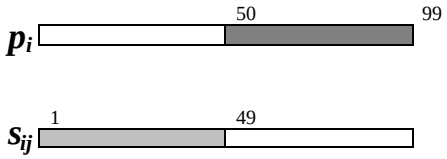


FIGURA 4.4 – Ilustração da relação IV

Relação V: $O(p_i)/O(s_{ij}) \leq 1$

As ordens de grandeza dos tempos de processamento e de *setup* não são iguais, porém existem intervalos comuns. Nesta relação, o intervalo dos tempos de *setup* é maior, mas contém todos os valores do intervalo dos tempos de processamento, como pode ser visto na figura 4.5.

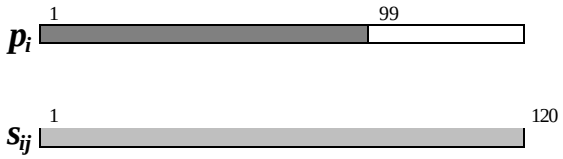


FIGURA 4.5 – Ilustração da relação V

Relação VI: $O(p_i)/O(s_{ij}) > 1$

Neste caso também as ordens de grandeza não são iguais, porém é o oposto da relação V, pois o intervalo dos tempos de processamento é maior e contém todos os valores do intervalo dos tempos de *setup*. Esta relação é ilustrada na figura 4.6.

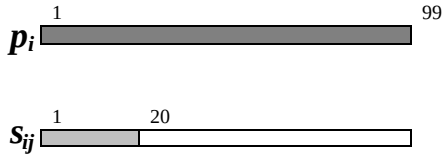


FIGURA 4.6 – Ilustração da relação VI

A geração dos tempos de processamento e de *setup* dentro de cada intervalo foi feita de forma aleatória com distribuição uniforme. Os limites dos intervalos das relações foram definidos e padronizados com base em trabalhos reportados na literatura e resumidos na tabela 4.1.

Simons Jr. (1992), Ríos-Mercado e Bard (1998) e Rajendran e Ziegler (2003) utilizaram intervalos comuns de tempos de processamento e de *setup*, com valores de 1 a 99, definindo os valores da relação I.

As relações II, III e IV são variações de ordens não-comuns. De forma semelhante a Das, Gupta e Khumawala (1995), valores dos tempos de processamento menores (1 a 99) do que os tempos de *setup* (100 a 120) são utilizados na relação II. Com base no trabalho de Ríos-Mercado e Bard (1999), valores dos tempos de processamento maiores (10 a 99) do que os tempos de *setup* (1 a 9) foram definidos na relação III e intervalos de tempos de processamento (50 a 99) e de *setup* (1 a 49) com amplitudes próximas são considerados na relação IV.

TABELA 4.1 – Parâmetros de trabalhos publicados

TRABALHO	AMBIENTE	FUNÇÃO-OBJETIVO	NÚMERO DE PROBLEMAS	NÚMERO DE TAREFAS	NÚMERO DE MÁQUINAS	INTERVALO DE p_i	INTERVALO DE s_{ij}	DISTRIBUIÇÃO
Simons Jr. (1992)	FS *setup* dep.	C_{max}	180 (20/classe)	5, 10, 15	5, 10, 15	1-99	1-99	Uniforme
Das, Gupta e Khumawala (1995)	FSP *setup* dep.	C_{max}	240	4-20	2-20	1-99	100-500	Uniforme
Huang e Li (1998)	FSH 2 estágios *setup* dep.	C_{max}	800 (100/categoria)	Fam. *i*: 3-5, 5-10; Parte *j*: 2-5, 5-15	Estágio 1: 1; Estágio 2: 2-5, 5-8	p_i: 5-20; p_{ij}: 1-2p_i	1-2p_i; p_i-$\sum p_{ij}$	Uniforme
Ríos-Mercado e Bard (1998)	FS *setup* dep.	C_{max}	360 (20/classe)	20, 50, 100	2, 4, 6	1-99 1-99	1-10 1-99	(não cita)
Ríos-Mercado e Bard (1999)	FS *setup* dep.	C_{max}	900 (20/classe)	20, 50, 100	2, 4, 6, 8, 10	10-100 20-100 50-100	1-10 20-40 1-50	Uniforme
Weng, Lu e Ren (2001)	Máq. Paralelas não-rel. *setup* dep.	$\sum_{j=1}^{n} \frac{1}{n} w_j C_j$	7000 (1000/heur.) (20/classe)	40, 60, 80, 100, 120	4, 6, 8, 10, 12	1-100 1-50	1-20	Uniforme
Barros (2002)	FSP *setup* dep.	C_{max}	180 (20/classe)	5, 10, 15	5, 10, 15	10-200	10-200	Uniforme
Lin e Liao (2003)	FSH 2 estágios *setup* dep.	wT_{max}	146 dias	24-40 / dia	Estágio 1: 1; Estágio 2: 4	(variável)	6-46 (min.)	Empírica / Uniforme
Rajendran e Ziegler (2003)	FS *setup* dep.	$w_j C_j + w_j T_j$	480 / 1200 — 30/classe	7, 8 10, 15, 20, 25, 30	5, 10, 15, 20	1-99	1-99 51-149	Uniforme
Ruiz e Maroto (2003)	FSH 5,10, 20 estágios *setup* dep.	C_{max}	1320 (110 probl. x 12 classes)	20, 50, 100, 200	1-3/estágio 2/estágio 3/estágio	1-99	10% p_i 50% p_i 100% p_i 125% p_i	Uniforme

Legenda:

FS: *Flow Shop*

FSP: *Flow Shop* Permutacional

FSH: *Flow Shop* Híbrido

C_{max}: *Makespan*

$\sum_{j=1}^{n} \frac{1}{n} w_j C_j$: tempo médio de fluxo ponderado

wT_{max}: *tardiness* máximo ponderado

$w_j C_j + w_j T_j$: soma do tempo de fluxo ponderado e do *tardiness* ponderado das tarefas

Intervalos comuns de tempos de processamento e de *setup*, mas com amplitudes diferentes, foram estabelecidos nas relações V e VI. Intervalos de tempos de processamento menores (1 a 99) do que os de *setup* (1 a 120) são considerados na relação V. E, como definido por Weng, Lu e Ren (2001), intervalos de tempos de processamento maiores (1 a 99) do que os tempos de *setup* (1 a 20) foram utilizados na relação VI.

4.1.2 Definição da amostragem

Na experimentação computacional foram testados 14.400 problemas, divididos em 144 classes definidas pelo número de tarefas (n), número de estágios de produção (K) e pelas relações entre as ordens de grandeza dos tempos de processamento e de *setup* ($O(p_i)/O(s_{ij})$).

Para cada classe, foram gerados aleatoriamente 100 problemas com $n \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120\}$, $K \in \{4, 7\}$ e uma das seis relações de grandeza entre os tempos. Assim, 12 (alternativas do número de tarefas) $\times$ 2 (alternativas do número de estágios) $\times$ 6 (relações $O(p_i)/O(s_{ij})$) = 144 classes. A quantidade de problemas gerados em cada classe objetiva reduzir o erro amostral.

Em cada estágio, o número de máquinas paralelas idênticas varia de 2 a 5, ou seja, $M_k \in \{2, 3, 4, 5\}$. Como M_k é gerado aleatoriamente com distribuição uniforme dentro deste conjunto, sua variação não altera a quantidade de classes.

O número de estágios e o número de máquinas paralelas idênticas em cada estágio definem o *lay-out* do sistema de produção.

4.1.3 Obtenção dos dados

De acordo com os parâmetros definidos anteriormente, todos os problemas foram gerados aleatoriamente por meio de um software construído especificamente para esta finalidade, denominado “Gerador de arquivos de dados”. A interface deste software é mostrada nas figuras 4.7 e 4.8.

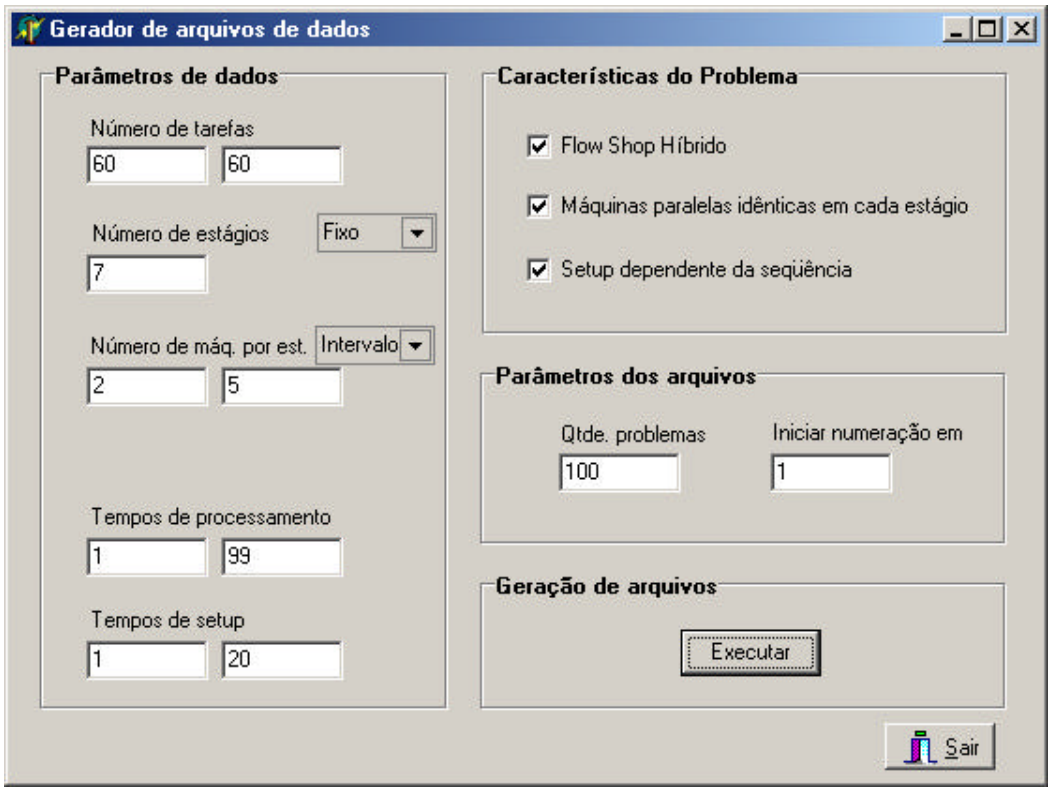


FIGURA 4.7 – Interface do "Gerador de arquivos de dados" com a opção de número de máquinas por estágio em *intervalo*

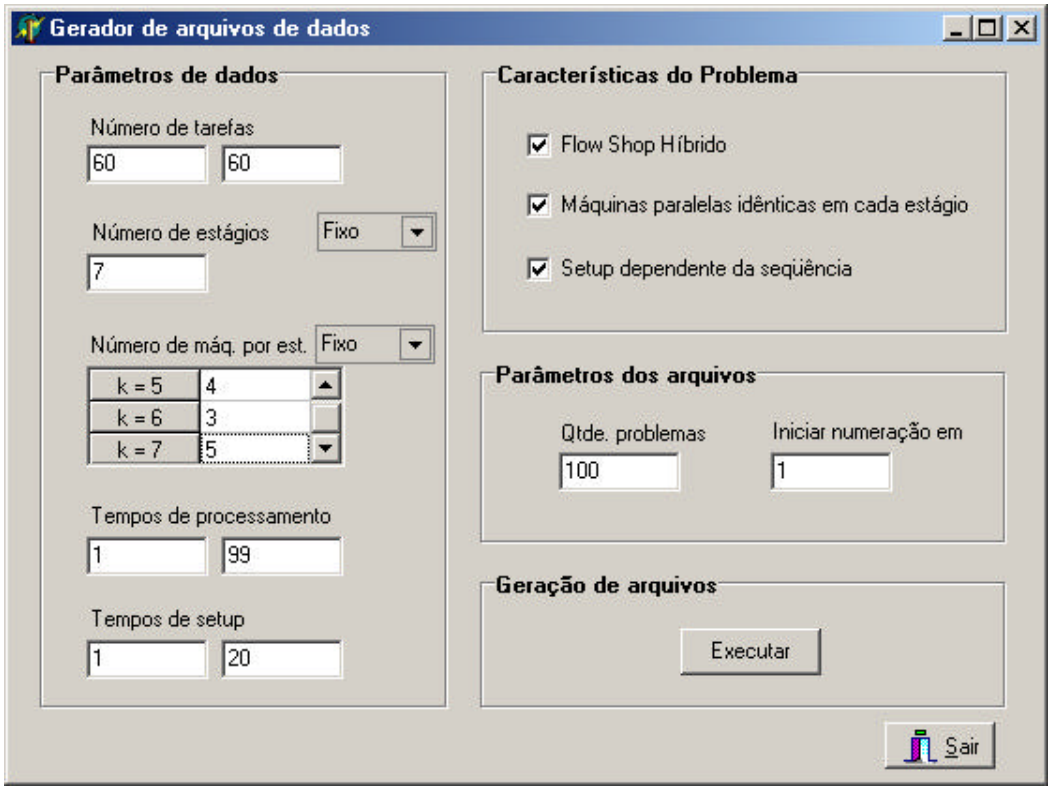


FIGURA 4.8 – Interface do "Gerador de arquivos de dados" com a opção de número de máquinas por estágio em *fixo*

Foi utilizada a linguagem de programação Delphi e o sistema operacional Windows. As configurações da máquina são as seguintes: processador Pentium 4 da Intel com 2 GHz de freqüência, 512 MB de memória RAM e disco rígido de 37,2 GB.

4.1.4 Processo de análise

Os resultados obtidos na experimentação computacional foram analisados por meio da porcentagem de sucesso, desvio relativo (%), desvio-padrão do desvio relativo e tempo médio de computação dos quatro métodos desenvolvidos para cada classe de problemas.

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

O **desvio relativo** mede a variação correspondente à melhor solução obtida pelos métodos. Quando o desvio relativo da solução de um problema é igual a zero para um determinado método, significa que a duração total da programação fornecida é a menor, ou seja, o algoritmo apresentou a melhor programação. Entretanto, mais de um método pode fornecer a melhor programação.

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

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

$$DR_h = \frac{D_h - D^*}{D^*} \tag{4.1}$$

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

O **desvio-padrão** de uma amostra mede o grau de dispersão dos elementos em torno da média. Neste trabalho, o desvio-padrão do desvio relativo é o valor da variação

dos desvios relativos de uma classe de problemas em torno do desvio relativo médio. Quanto menor for o valor do desvio-padrão, melhor é o método de solução quando comparado com um outro, no caso em que ambos apresentarem desvios relativos médios com diferença não significativa.

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

$$S_h = \sqrt{\frac{\sum_{i=1}^{L} (DR_{h_i} - DRM)^2}{L-1}} \tag{4.2}$$

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

O **tempo médio de computação** de um método é calculado pela soma dos tempos de computação de cada problema dividida pelo número total de problemas resolvidos (média aritmética dos tempos de computação). Na experimentação computacional, o tempo médio de computação foi medido em milissegundos (ms).

Como não foram encontrados na literatura métodos heurísticos construtivos para programação do ambiente tratado neste trabalho, os algoritmos desenvolvidos foram comparados entre si.

4.2 Resultados obtidos

Para resolver os problemas gerados foi construído o software "*Flow Shop* Híbrido", cuja interface é apresentada na figura 4.9. Este software resolve os problemas de uma classe utilizando cada um dos quatro métodos separadamente – pelos botões "LPT", "TOTAL", "SCT" e "SCT/LPST", e gera um arquivo de saída em formato *.txt* com o número do problema, o *makespan* e a programação das máquinas.

Também é possível gerar arquivos comparativos do *makespan* e do tempo de computação dos quatro métodos – botões "Makespan" e "Tempo", respectivamente. Foi desta forma que os resultados dos problemas foram comparados e analisados.

Para simplificação da notação, o Procedimento 1 com a regra de prioridade LPT foi denotado como "11" e com a regra de prioridade TOTAL como "12". O Procedimento 2 com a regra de alocação SCT foi denominado "21" e com a regra SCT/LPST foi indicado como "22".

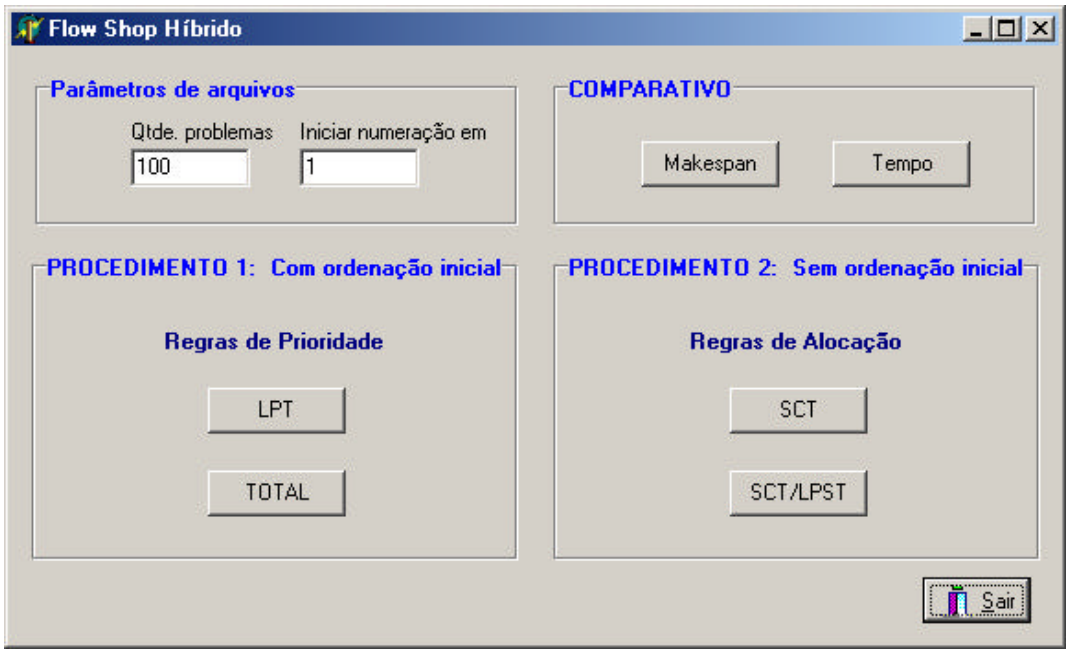


FIGURA 4.9 – Interface do software "*Flow Shop* Híbrido"

As tabelas 4.2 a 4.7 referem-se aos resultados obtidos pelos métodos para as porcentagens de sucesso das seis relações de ordens dos tempos de processamento e de *setup*. As porcentagens são apresentadas em função do número de tarefas e separadamente para 4 e 7 estágios.

TABELA 4.2 – Porcentagem de sucesso da relação I para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	24	21	52	3	28	17	49	6
20	10	21	69	1	27	21	50	2
30	15	29	55	2	18	31	49	2
40	14	32	54	1	24	28	47	1
50	13	43	43	1	14	35	49	2
60	12	42	45	1	17	33	49	1
70	7	51	41	1	16	44	40	0
80	9	54	34	3	14	47	34	5
90	6	65	29	0	14	54	32	0
100	7	57	35	1	10	51	38	1
110	5	60	34	1	8	63	27	2
120	5	65	30	0	9	69	22	0
% média	**10,6**	**45,0**	**43,4**	**1,3**	**16,6**	**41,1**	**40,5**	**1,8**

TABELA 4.3 – Porcentagem de sucesso da relação II para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	40	12	45	5	39	17	37	9
20	22	21	56	2	31	14	54	1
30	24	26	50	0	27	14	56	4
40	13	33	54	0	21	25	52	3
50	19	23	59	1	19	27	54	0
60	13	23	64	0	20	28	49	4
70	16	22	62	0	20	29	53	0
80	9	38	53	1	13	37	51	0
90	7	43	49	1	12	37	50	3
100	7	38	54	2	18	35	46	1
110	4	40	56	0	17	41	40	2
120	2	45	53	1	19	32	49	0
% média	**14,7**	**30,3**	**54,6**	**1,1**	**21,3**	**28**	**49,3**	**2,3**

TABELA 4.4 – Porcentagem de sucesso da relação III para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	40	10	52	4	39	9	48	7
20	37	16	47	1	28	16	55	2
30	28	13	58	1	31	7	61	1
40	21	9	70	0	23	12	65	0
50	27	9	64	0	33	12	55	1
60	29	17	56	0	28	17	53	2
70	21	7	72	0	26	14	58	2
80	24	12	64	0	27	4	68	1
90	21	10	70	0	24	9	66	3
100	21	11	68	0	29	12	59	0
110	20	11	69	0	22	12	65	2
120	23	10	70	1	28	13	58	1
% média	**26**	**11,3**	**63,3**	**0,6**	**28,2**	**11,4**	**59,3**	**1,8**

TABELA 4.5 – Porcentagem de sucesso da relação IV para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	27	15	55	3	28	15	53	5
20	12	25	63	1	26	19	53	4
30	9	27	62	3	20	28	51	3
40	10	32	52	6	13	31	54	2
50	14	29	58	1	17	35	44	4
60	11	48	39	2	16	47	32	6
70	11	48	40	3	18	41	39	3
80	10	45	43	2	17	46	36	2
90	9	54	37	0	12	53	33	3
100	3	59	37	1	8	61	27	4
110	4	56	40	1	8	53	39	1
120	8	50	42	0	15	49	34	2
% média	**10,7**	**40,7**	**47,3**	**1,9**	**16,5**	**39,8**	**41,3**	**3,3**

TABELA 4.6 – Porcentagem de sucesso da relação V para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	24	21	57	0	27	23	44	6
20	22	21	55	2	21	29	49	1
30	17	32	47	5	18	37	44	2
40	13	41	46	0	24	38	39	0
50	10	58	32	0	14	48	37	1
60	9	51	40	0	13	45	43	0
70	9	54	36	1	8	49	41	3
80	4	65	29	2	11	58	30	1
90	5	71	24	0	10	61	27	2
100	7	60	32	1	9	69	19	4
110	4	70	25	1	11	65	23	1
120	8	68	23	1	8	65	25	2
% média	**11**	**51**	**37,2**	**1,1**	**14,5**	**48,9**	**35,1**	**1,9**

TABELA 4.7 – Porcentagem de sucesso da relação VI para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	49	12	36	3	35	12	52	1
20	29	16	55	0	33	10	55	3
30	26	19	55	0	27	15	58	1
40	17	12	70	1	22	10	66	2
50	19	8	72	1	26	16	58	1
60	18	10	72	0	26	14	60	0
70	21	18	62	0	26	12	60	2
80	12	21	65	2	27	9	64	0
90	11	20	67	2	23	13	62	2
100	11	12	77	0	24	21	54	2
110	9	21	70	0	21	21	56	2
120	10	19	72	0	19	20	58	3
% média	**19,3**	**15,7**	**64,4**	**0,8**	**25,8**	**14,4**	**58,6**	**1,6**

TABELA 4.8 – Porcentagem de sucesso para 4 e 7 estágios agregando as relações $O(p_i)/O(s_{ij})$

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	34,0	15,2	49,5	3,0	32,7	15,5	47,2	5,7
20	22,0	20,0	57,5	1,2	27,7	18,2	52,7	2,2
30	19,8	24,3	54,5	1,8	23,5	22,0	53,2	2,2
40	14,7	26,5	57,7	1,3	21,2	24,0	53,8	1,3
50	17,0	28,3	54,7	0,7	20,5	28,8	49,5	1,5
60	15,3	31,8	52,7	0,5	20,0	30,7	47,7	2,2
70	14,2	33,3	52,2	0,8	19,0	31,5	48,5	1,7
80	11,3	39,2	48,0	1,7	18,2	33,5	47,2	1,5
90	9,8	43,8	46,0	0,5	15,8	37,8	45,0	2,2
100	9,3	39,5	50,5	0,8	16,3	41,5	40,5	2,0
110	7,7	43,0	49,0	0,5	14,5	42,5	41,7	1,7
120	9,3	42,8	48,3	0,5	16,3	41,3	41,0	1,3
% média	**15,4**	**32,3**	**51,7**	**1,1**	**20,5**	**30,6**	**47,3**	**2,1**

A tabela 4.8 apresenta o resultado da porcentagem de sucesso para 4 e 7 estágios com relações $O(p_i)/O(s_{ij})$ agregadas. A tabela 4.9 contém o total geral das porcentagens de sucesso, agregando as relações $O(p_i)/O(s_{ij})$ e o número de tarefas.

TABELA 4.9 – Total geral das porcentagens de sucesso

	11	**12**	**21**	**22**
total de problemas	2581	4531	7130	232
% total média	17,9	31,5	49,5	1,6

As tabelas 4.10 a 4.15 referem-se aos resultados obtidos para o desvio relativo médio das seis relações de ordens dos tempos de processamento e de *setup*. O valor do desvio relativo médio em porcentagem é apresentado em função do número de tarefas e separadamente para 4 e 7 estágios.

TABELA 4.10 – Desvio relativo médio (%) da relação I para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	8,9	11,1	3,9	24,8	5,7	8,6	4,6	18,9
20	11,1	11,1	2,0	33,0	6,3	7,3	3,3	19,7
30	9,8	7,4	2,9	30,9	6,0	5,2	2,8	21,3
40	9,3	6,7	2,1	33,6	6,1	4,9	2,6	24,3
50	10,7	6,5	3,2	38,4	7,4	5,6	2,1	25,1
60	9,4	6,4	2,6	39,3	6,1	4,2	2,3	28,9
70	10,0	6,1	3,6	37,9	5,9	4,5	2,3	25,7
80	11,8	7,8	3,2	40,1	7,1	3,7	3,1	25,1
90	10,4	4,8	4,6	44,2	7,5	4,2	3,2	29,2
100	9,9	5,0	4,2	42,4	5,6	2,8	2,3	26,9
110	13,3	6,4	4,0	46,2	5,2	2,3	3,0	25,8
120	12,6	5,5	4,0	44,7	5,8	1,8	3,7	27,0
média	**10,6**	**7,0**	**3,4**	**38,0**	**6,2**	**4,6**	**2,9**	**24,8**

TABELA 4.11 – Desvio relativo médio (%) da relação II para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	4,0	7,8	3,3	10,5	3,7	5,8	3,4	9,2
20	3,2	4,4	1,2	8,4	2,6	4,6	1,4	7,8
30	2,7	3,5	1,4	7,6	2,8	3,7	1,1	6,7
40	2,5	4,6	1,0	7,0	2,5	2,2	0,9	6,2
50	1,9	3,4	0,6	5,7	2,1	2,1	0,8	5,7
60	2,2	4,4	0,5	6,8	2,0	1,8	0,9	5,4
70	2,2	2,4	0,7	6,3	1,6	2,7	0,8	5,9
80	2,1	3,0	0,8	5,7	1,8	1,8	0,7	4,7
90	1,9	3,0	0,7	5,9	1,6	2,4	0,7	5,2
100	1,8	3,5	0,6	6,3	1,5	1,4	0,7	5,1
110	1,9	2,7	0,6	6,0	1,4	1,3	0,7	4,5
120	1,9	2,3	0,6	5,9	1,3	1,7	0,7	4,8
média	**2,4**	**3,8**	**1,0**	**6,8**	**2,1**	**2,6**	**1,1**	**5,9**

TABELA 4.12 – Desvio relativo médio (%) da relação III para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	4,7	10,3	3,5	14,4	4,6	8,3	3,1	12,2
20	4,7	8,1	2,6	14,3	5,1	7,0	1,8	11,5
30	4,0	7,9	2,2	12,5	4,0	6,6	1,8	12,4
40	4,5	6,9	1,0	13,6	3,8	5,6	1,2	11,2
50	3,6	5,7	1,0	12,5	3,4	6,2	1,5	11,8
60	3,0	5,0	1,0	13,1	3,1	4,3	1,3	10,4
70	3,1	6,5	0,5	12,1	3,1	5,1	0,8	10,3
80	2,4	5,5	0,5	11,7	2,6	4,8	0,8	11,2
90	2,4	5,4	0,5	12,0	2,4	4,3	0,7	10,0
100	2,0	7,3	0,4	12,1	2,5	4,2	0,7	11,2
110	2,2	4,7	0,4	12,4	2,4	3,2	0,7	9,7
120	1,8	6,0	0,4	11,1	2,1	4,6	0,7	8,6
média	**3,2**	**6,6**	**1,2**	**12,6**	**3,3**	**5,4**	**1,2**	**10,9**

TABELA 4.13 – Desvio relativo médio (%) da relação IV para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	4,4	5,3	2,1	11,3	3,8	4,6	1,5	7,1
20	4,4	5,7	1,3	12,6	3,4	3,9	1,3	8,8
30	4,4	5,2	1,3	13,1	2,9	3,6	1,6	8,5
40	4,1	3,4	1,3	11,5	3,5	3,1	1,0	10,8
50	4,9	5,1	0,9	16,1	2,7	2,5	1,3	9,6
60	4,3	3,6	1,6	13,4	2,7	1,8	1,8	7,0
70	3,6	3,1	1,7	11,0	2,7	2,1	1,3	9,8
80	4,4	4,2	1,2	14,1	2,9	1,6	1,3	10,1
90	5,1	4,8	1,8	14,9	2,9	2,4	1,5	10,1
100	5,2	3,8	1,7	15,5	2,8	1,3	1,7	9,8
110	5,4	5,2	1,6	16,9	3,1	1,7	1,5	11,9
120	6,0	5,3	1,7	17,8	2,9	1,8	1,3	10,6
média	**4,7**	**4,5**	**1,5**	**14,0**	**3,0**	**2,5**	**1,4**	**9,5**

TABELA 4.14 – Desvio relativo médio (%) da relação V para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	8,6	10,4	3,9	28,8	6,6	7,7	3,8	17,6
20	12,9	12,3	4,0	39,0	6,5	5,7	3,9	20,7
30	9,3	5,9	3,6	36,9	7,8	5,4	2,9	24,5
40	11,0	7,6	3,5	42,1	6,4	4,4	3,5	24,2
50	11,3	5,6	4,0	41,7	6,8	3,5	3,5	27,3
60	10,8	4,9	3,5	44,1	6,5	3,1	2,8	29,0
70	10,8	7,1	4,0	43,2	7,7	4,6	2,9	33,8
80	12,8	4,6	4,7	46,4	6,7	2,9	4,2	28,9
90	13,6	4,9	5,4	51,7	6,7	2,4	3,6	28,0
100	13,0	5,5	5,4	47,4	6,7	2,6	4,1	28,9
110	12,5	4,9	5,3	57,0	7,2	2,5	3,3	30,4
120	15,8	6,2	5,5	54,5	7,3	2,8	3,7	32,0
média	**11,9**	**6,7**	**4,4**	**44,4**	**6,9**	**4,0**	**3,5**	**27,1**

TABELA 4.15 – Desvio relativo médio (%) da relação VI para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	5,3	11,7	5,7	16,4	5,4	9,6	3,2	14,8
20	5,6	8,8	2,4	18,1	5,7	9,0	2,6	16,6
30	5,2	6,9	1,6	18,4	4,9	6,9	1,5	13,3
40	5,8	7,5	1,2	17,5	5,2	6,5	1,2	14,3
50	4,7	7,8	0,7	18,5	3,9	5,2	1,4	12,5
60	3,9	6,1	0,6	17,6	4,1	5,4	1,2	13,5
70	4,2	6,0	1,1	16,0	3,4	5,2	1,2	12,8
80	3,8	4,3	0,8	15,0	3,3	5,6	1,1	15,4
90	3,8	5,7	0,7	15,1	3,2	4,0	1,1	12,1
100	4,1	5,4	0,4	16,1	2,7	4,3	1,1	12,9
110	4,0	7,0	0,5	17,5	3,0	3,9	1,2	12,6
120	3,8	5,3	0,4	17,8	2,9	3,2	0,8	12,9
média	**4,5**	**6,9**	**1,3**	**17,0**	**4,0**	**5,7**	**1,5**	**13,6**

As tabelas 4.16 a 4.21 referem-se aos resultados obtidos para o desvio-padrão do desvio relativo das seis relações de ordens dos tempos de processamento e de *setup*. O valor do desvio-padrão é apresentado em função do número de tarefas e separadamente para 4 e 7 estágios.

TABELA 4.16 – Desvio-padrão do DR da relação I para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	0,08	0,10	0,06	0,15	0,06	0,08	0,07	0,11
20	0,08	0,14	0,05	0,20	0,06	0,07	0,06	0,13
30	0,08	0,10	0,05	0,23	0,05	0,06	0,04	0,16
40	0,08	0,11	0,04	0,23	0,06	0,07	0,04	0,19
50	0,11	0,11	0,04	0,31	0,07	0,08	0,03	0,22
60	0,09	0,11	0,03	0,28	0,06	0,06	0,04	0,23
70	0,10	0,13	0,04	0,31	0,07	0,10	0,03	0,23
80	0,14	0,14	0,04	0,38	0,08	0,08	0,03	0,26
90	0,11	0,11	0,05	0,34	0,09	0,09	0,03	0,27
100	0,11	0,12	0,05	0,33	0,07	0,08	0,03	0,24
110	0,14	0,11	0,04	0,38	0,04	0,07	0,03	0,21
120	0,15	0,11	0,04	0,40	0,06	0,06	0,03	0,23
média	**0,11**	**0,12**	**0,04**	**0,30**	**0,06**	**0,08**	**0,04**	**0,21**

TABELA 4.17 – Desvio-padrão do DR da relação II para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	0,05	0,06	0,05	0,06	0,05	0,05	0,04	0,06
20	0,03	0,05	0,02	0,05	0,03	0,04	0,02	0,04
30	0,03	0,05	0,02	0,04	0,03	0,05	0,02	0,04
40	0,02	0,10	0,02	0,04	0,02	0,02	0,01	0,04
50	0,02	0,08	0,01	0,04	0,02	0,02	0,01	0,04
60	0,02	0,09	0,01	0,04	0,02	0,02	0,01	0,04
70	0,02	0,04	0,01	0,04	0,01	0,06	0,01	0,04
80	0,01	0,08	0,01	0,04	0,02	0,04	0,01	0,04
90	0,01	0,07	0,01	0,04	0,01	0,06	0,01	0,04
100	0,01	0,09	0,01	0,04	0,01	0,02	0,01	0,04
110	0,01	0,06	0,01	0,04	0,01	0,03	0,01	0,04
120	0,01	0,06	0,01	0,04	0,01	0,04	0,01	0,04
média	**0,02**	**0,07**	**0,02**	**0,04**	**0,02**	**0,04**	**0,02**	**0,04**

TABELA 4.18 – Desvio-padrão do DR da relação III para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	0,06	0,09	0,07	0,10	0,06	0,06	0,04	0,08
20	0,06	0,07	0,04	0,07	0,05	0,07	0,03	0,06
30	0,04	0,08	0,04	0,06	0,05	0,06	0,03	0,06
40	0,04	0,06	0,02	0,07	0,03	0,05	0,02	0,06
50	0,03	0,06	0,02	0,07	0,04	0,06	0,03	0,07
60	0,03	0,06	0,02	0,07	0,03	0,04	0,02	0,07
70	0,03	0,10	0,01	0,07	0,03	0,06	0,01	0,07
80	0,02	0,10	0,01	0,07	0,03	0,04	0,02	0,07
90	0,02	0,08	0,01	0,08	0,02	0,06	0,01	0,07
100	0,02	0,13	0,01	0,08	0,03	0,05	0,01	0,07
110	0,02	0,08	0,01	0,09	0,02	0,03	0,01	0,08
120	0,02	0,13	0,01	0,08	0,02	0,07	0,01	0,07
média	**0,03**	**0,09**	**0,02**	**0,07**	**0,03**	**0,05**	**0,02**	**0,07**

TABELA 4.19 – Desvio-padrão do DR da relação IV para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	0,04	0,04	0,03	0,06	0,04	0,04	0,03	0,05
20	0,03	0,08	0,02	0,08	0,03	0,03	0,02	0,05
30	0,04	0,06	0,02	0,10	0,03	0,06	0,02	0,07
40	0,04	0,05	0,02	0,10	0,03	0,04	0,02	0,09
50	0,05	0,08	0,02	0,12	0,03	0,04	0,02	0,09
60	0,05	0,07	0,02	0,12	0,03	0,04	0,02	0,08
70	0,03	0,07	0,02	0,11	0,03	0,04	0,02	0,09
80	0,05	0,08	0,02	0,13	0,03	0,03	0,01	0,10
90	0,06	0,11	0,02	0,15	0,03	0,06	0,02	0,10
100	0,05	0,08	0,02	0,14	0,02	0,04	0,02	0,09
110	0,06	0,09	0,02	0,16	0,04	0,05	0,02	0,12
120	0,07	0,09	0,02	0,16	0,03	0,05	0,02	0,11
média	**0,05**	**0,08**	**0,02**	**0,12**	**0,03**	**0,04**	**0,02**	**0,09**

TABELA 4.20 – Desvio-padrão do DR da relação V para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	0,08	0,10	0,07	0,17	0,07	0,07	0,05	0,11
20	0,11	0,14	0,07	0,27	0,06	0,07	0,06	0,13
30	0,09	0,08	0,05	0,27	0,07	0,08	0,04	0,19
40	0,10	0,13	0,04	0,32	0,07	0,07	0,05	0,21
50	0,11	0,12	0,05	0,34	0,07	0,06	0,04	0,23
60	0,11	0,11	0,04	0,34	0,06	0,05	0,04	0,22
70	0,13	0,13	0,05	0,37	0,09	0,10	0,03	0,30
80	0,14	0,10	0,05	0,38	0,08	0,07	0,05	0,25
90	0,14	0,11	0,05	0,41	0,07	0,06	0,04	0,26
100	0,14	0,12	0,05	0,43	0,07	0,09	0,04	0,28
110	0,14	0,12	0,05	0,38	0,09	0,08	0,03	0,30
120	0,18	0,13	0,05	0,49	0,10	0,09	0,04	0,33
média	**0,12**	**0,12**	**0,05**	**0,35**	**0,07**	**0,07**	**0,04**	**0,23**

TABELA 4.21 – Desvio-padrão do DR da relação VI para 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	0,07	0,09	0,08	0,09	0,06	0,07	0,05	0,09
20	0,06	0,08	0,04	0,09	0,06	0,07	0,04	0,08
30	0,05	0,06	0,03	0,09	0,05	0,07	0,03	0,07
40	0,04	0,06	0,02	0,10	0,05	0,05	0,02	0,08
50	0,04	0,09	0,02	0,10	0,04	0,06	0,02	0,07
60	0,03	0,06	0,01	0,09	0,04	0,06	0,02	0,09
70	0,04	0,10	0,02	0,09	0,03	0,05	0,02	0,09
80	0,03	0,06	0,02	0,10	0,03	0,04	0,02	0,09
90	0,03	0,09	0,01	0,11	0,03	0,04	0,02	0,09
100	0,03	0,08	0,01	0,10	0,03	0,06	0,02	0,09
110	0,03	0,13	0,01	0,11	0,03	0,06	0,02	0,10
120	0,03	0,08	0,01	0,11	0,02	0,03	0,01	0,10
média	**0,04**	**0,08**	**0,02**	**0,10**	**0,04**	**0,06**	**0,02**	**0,09**

As tabelas 4.22 a 4.27 referem-se aos resultados obtidos para os tempos médios de computação, medidos em milissegundos (ms), das seis relações de ordens dos tempos de processamento e de *setup*. Os tempos médios são apresentados em função das tarefas e separadamente para 4 e 7 estágios.

TABELA 4.22 – Tempo médio de computação (ms) dos problemas da relação I com 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	1,7	1,6	1,7	1,7	2,2	2,2	2,1	2,1
20	1,9	2,0	1,9	1,8	1,8	2,0	1,7	1,7
30	1,7	2,8	1,7	1,7	2,1	2,7	1,9	2,0
40	1,6	3,6	1,9	1,8	1,8	3,8	1,9	2,1
50	2,0	5,2	1,9	1,8	1,9	5,4	2,4	2,0
60	1,8	7,9	2,1	1,9	5,4	10,8	5,8	6,1
70	1,9	10,7	2,1	2,0	5,8	14,8	5,8	5,7
80	2,6	16,9	2,5	2,4	2,2	16,1	2,6	2,6
90	1,9	19,6	2,4	2,4	2,4	24,3	7,1	6,4
100	2,0	26,4	2,5	2,4	3,1	26,5	3,2	3,7
110	2,2	35,8	2,9	2,6	6,0	40,0	7,0	7,3
120	2,2	44,6	3,6	2,9	5,9	46,2	7,4	7,7
média	**2,0**	**14,8**	**2,3**	**2,1**	**3,4**	**16,2**	**4,1**	**4,1**

TABELA 4.23 – Tempo médio de computação (ms) dos problemas da relação II com 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	4,9	5,0	4,8	5,0	1,7	1,8	1,7	1,8
20	1,9	2,0	1,9	1,9	1,9	2,0	1,8	1,8
30	4,9	5,5	5,2	4,9	1,7	2,6	1,9	2,1
40	7,3	8,9	8,2	7,4	4,8	6,1	4,9	4,9
50	1,7	5,2	2,0	1,9	6,4	8,7	6,1	6,5
60	1,8	7,6	2,5	2,1	6,0	11,3	6,1	6,1
70	1,9	10,3	2,2	2,1	5,8	14,1	5,9	6,1
80	2,0	16,0	2,4	2,1	5,9	17,9	6,6	6,6
90	2,0	19,9	2,5	2,4	6,4	22,7	6,7	6,3
100	2,0	26,0	2,7	2,4	2,2	26,5	3,2	2,9
110	2,3	37,1	3,1	2,7	2,2	35,1	3,4	3,4
120	2,2	43,5	3,2	2,8	6,0	46,2	7,8	7,5
média	**2,9**	**15,6**	**3,4**	**3,1**	**4,2**	**16,2**	**4,7**	**4,7**

TABELA 4.24 – Tempo médio de computação (ms) dos problemas da relação III com 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	4,5	5,0	5,0	5,0	4,4	5,3	4,9	4,6
20	4,7	5,0	5,0	5,1	4,7	6,2	5,3	5,3
30	8,6	8,5	9,0	8,9	5,0	6,5	5,0	5,0
40	4,9	9,5	5,1	5,1	5,0	9,8	5,4	5,2
50	4,9	10,1	5,0	5,0	6,3	7,1	5,7	6,1
60	5,1	10,2	5,0	5,0	6,0	12,1	7,0	6,9
70	5,1	14,8	5,4	5,1	6,4	14,5	6,6	6,4
80	6,2	18,3	7,4	7,1	6,1	17,9	6,2	6,1
90	5,6	21,2	5,7	5,4	5,8	22,7	6,9	6,9
100	5,8	27,9	6,3	6,1	6,1	29,2	7,1	6,9
110	6,1	37,0	7,0	6,8	6,4	37,0	7,3	6,8
120	5,8	45,0	7,0	6,8	5,7	45,5	7,3	6,5
média	**5,6**	**17,7**	**6,1**	**6,0**	**5,7**	**17,8**	**6,2**	**6,1**

TABELA 4.25 – Tempo médio de computação (ms) dos problemas da relação IV com 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	4,1	5,0	5,0	5,0	4,6	4,6	4,9	4,9
20	4,8	5,0	5,1	5,0	6,3	9,5	7,5	6,8
30	5,4	5,4	5,9	6,0	5,1	8,5	5,8	5,4
40	5,1	9,7	5,5	5,5	6,6	7,3	6,4	6,8
50	5,0	9,8	5,0	5,1	5,1	9,1	5,3	5,2
60	5,5	11,2	5,6	5,5	5,3	10,9	5,5	5,5
70	5,7	13,9	6,1	6,1	5,6	15,0	6,3	7,0
80	5,2	18,5	5,7	5,6	5,3	17,7	6,1	6,6
90	5,4	22,0	5,8	5,5	5,0	23,0	7,4	6,8
100	5,8	28,9	6,2	6,1	10,7	28,5	7,2	6,5
110	5,7	37,4	7,0	6,7	6,0	37,4	7,4	8,3
120	5,9	53,6	7,3	6,3	6,2	49,4	9,1	7,8
média	**5,3**	**18,4**	**5,9**	**5,7**	**6,0**	**18,4**	**6,6**	**6,5**

TABELA 4.26 – Tempo médio de computação (ms) dos problemas da relação V com 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	5,0	5,1	5,0	5,1	4,7	4,9	5,0	5,0
20	4,8	5,0	5,0	5,1	6,9	9,6	8,5	7,8
30	5,5	5,5	5,7	5,7	6,1	9,2	7,0	6,3
40	5,6	9,9	5,8	5,7	6,1	7,7	6,9	7,1
50	5,2	9,3	5,3	5,5	5,5	9,7	5,8	5,3
60	5,6	11,5	6,2	5,5	5,3	16,6	5,3	5,6
70	6,1	13,7	6,2	6,5	5,8	14,3	6,0	6,5
80	5,2	18,2	6,0	5,5	6,2	17,6	6,6	6,2
90	5,7	22,7	6,3	6,0	6,1	25,0	3,7	6,0
100	5,2	29,4	5,6	6,7	5,5	30,5	7,6	6,9
110	5,8	38,7	7,3	6,7	5,7	40,5	8,8	6,3
120	6,0	50,3	6,9	6,6	6,5	48,3	7,5	9,4
média	**5,5**	**18,3**	**5,9**	**5,9**	**5,9**	**19,5**	**6,6**	**6,5**

TABELA 4.27 – Tempo médio de computação (ms) dos problemas da relação VI com 4 e 7 estágios

	K = 4				K = 7			
n	**11**	**12**	**21**	**22**	**11**	**12**	**21**	**22**
10	1,8	1,8	1,8	1,9	2,0	2,0	2,0	2,0
20	1,4	2,1	2,0	1,7	5,3	8,6	6,5	6,0
30	1,7	2,5	1,7	1,7	5,0	7,3	5,2	5,3
40	5,1	9,7	5,1	5,2	1,8	3,5	1,9	1,8
50	4,9	9,4	5,0	5,1	5,2	9,4	5,1	5,4
60	5,4	11,2	6,0	5,8	5,5	11,4	5,9	5,7
70	5,1	10,8	5,2	4,9	4,0	12,7	5,1	4,1
80	5,5	17,5	6,2	5,2	6,3	17,7	7,1	6,6
90	2,3	16,4	2,4	2,3	6,0	23,5	6,0	6,6
100	2,1	25,5	2,8	2,5	8,1	36,9	7,4	5,5
110	2,3	36,5	3,6	2,8	6,2	37,6	7,3	6,8
120	2,7	51,3	4,3	3,3	6,2	45,9	8,1	7,0
média	**3,4**	**16,2**	**3,8**	**3,5**	**5,1**	**18,0**	**5,6**	**5,2**

O apêndice B apresenta os parâmetros de cada classe de problemas das seis relações $O(p_i)/O(s_{ij})$ e tabelas gerais com os resultados dos quatro métodos para cada uma das quatro medidas de comparação.

4.3 Análise dos resultados

Primeiramente, será discutida a análise dos resultados com base na porcentagem de sucesso dos métodos de solução. O apêndice C apresenta os gráficos gerais para 4 e 7 estágios e cada uma das quatro medidas de comparação.

Como o foco da pesquisa é analisar a influência da relação entre as ordens de grandeza dos tempos de processamento e de *setup* em cada método de solução, foram apresentados gráficos para as relações separadamente.

Os gráficos das figuras 4.10 a 4.15 mostram a comparação da porcentagem de sucesso entre os métodos para as seis relações.

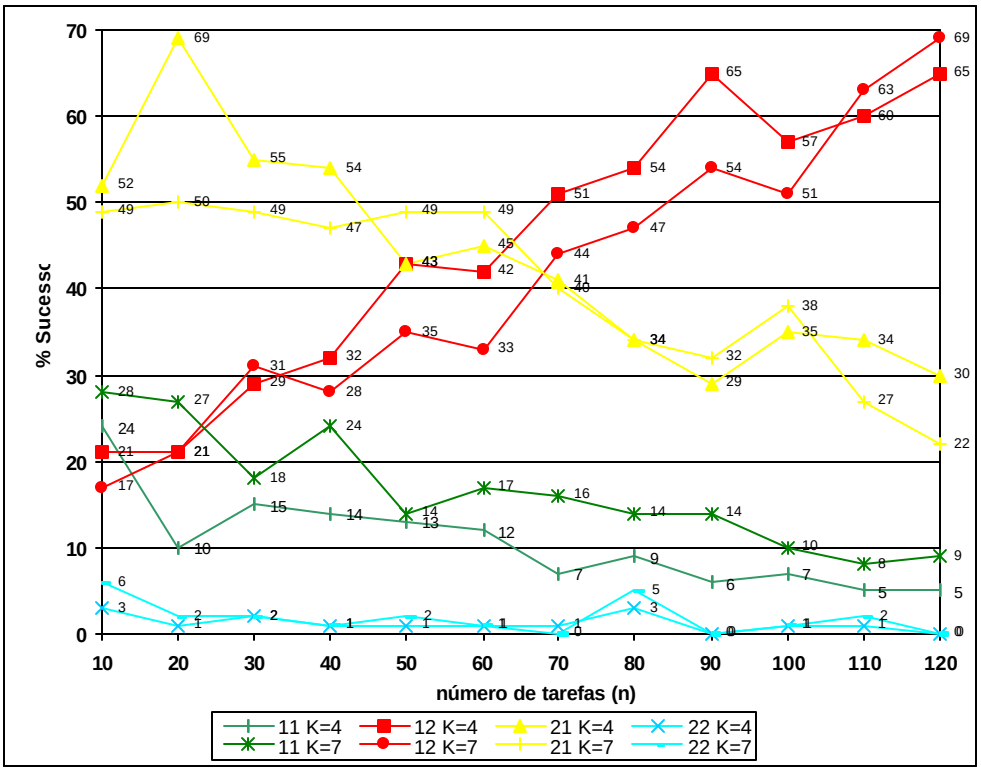


FIGURA 4.10 – Comparação da porcentagem de sucesso entre os métodos – relação I

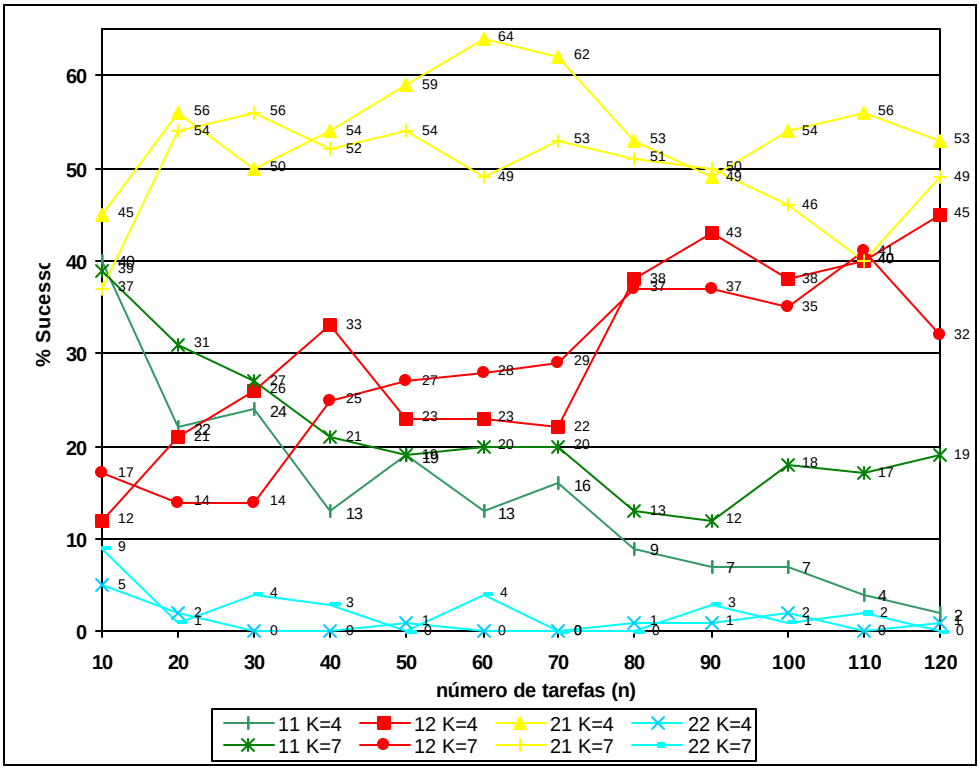


FIGURA 4.11 – Comparação da porcentagem de sucesso entre os métodos – relação II

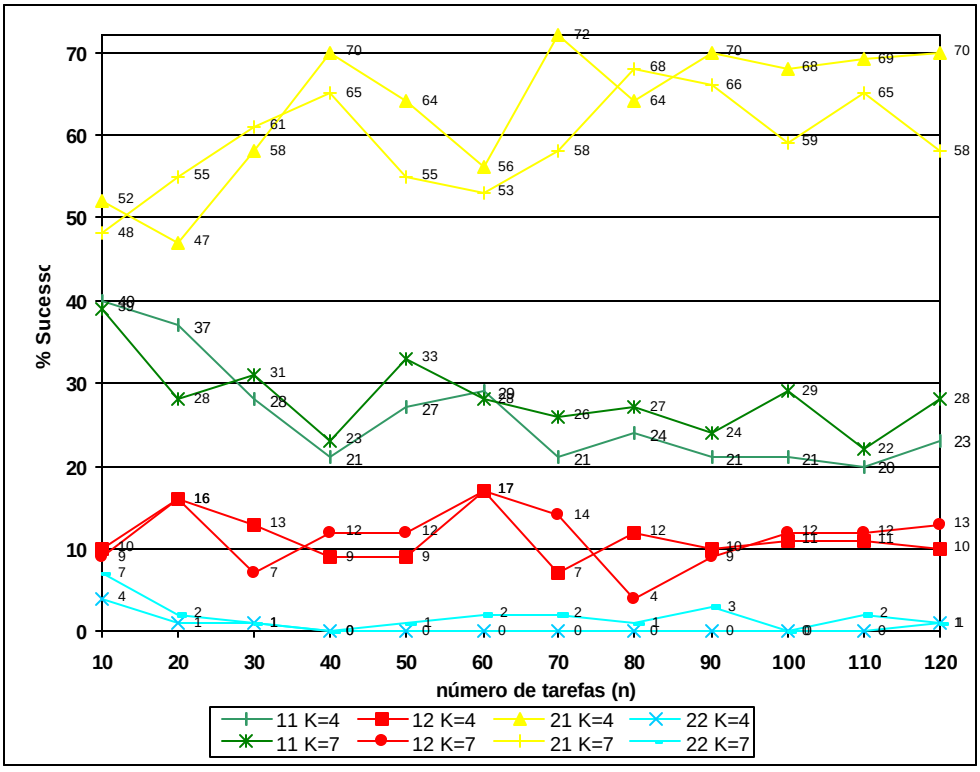


FIGURA 4.12 – Comparação da porcentagem de sucesso entre os métodos – relação III

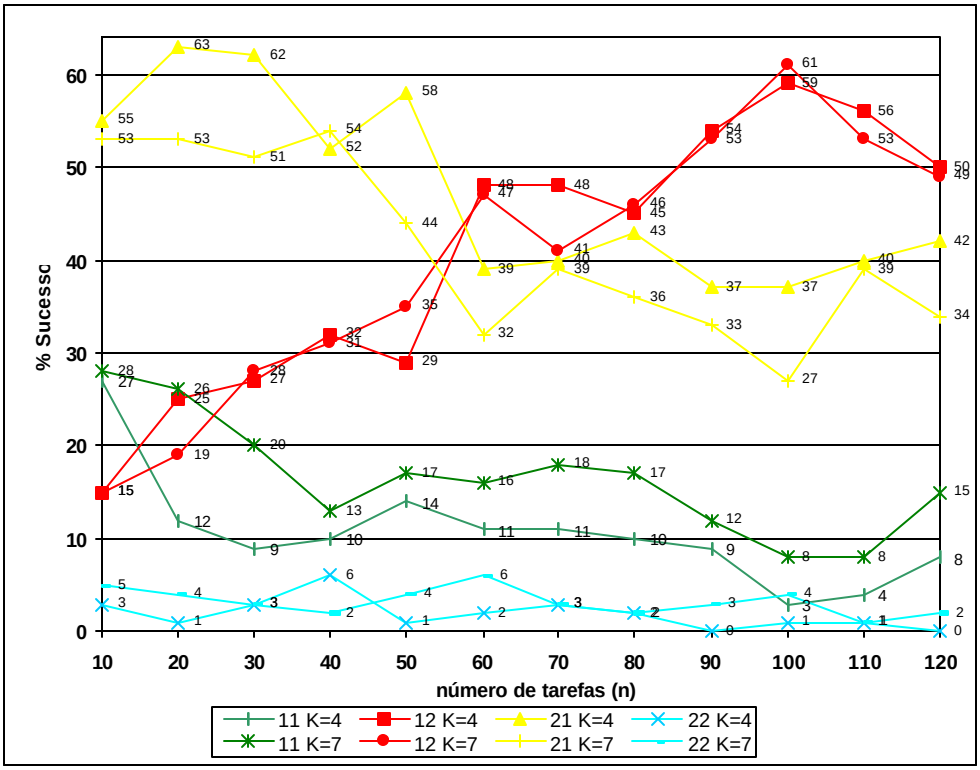


FIGURA 4.13 – Comparação da porcentagem de sucesso entre os métodos – relação IV

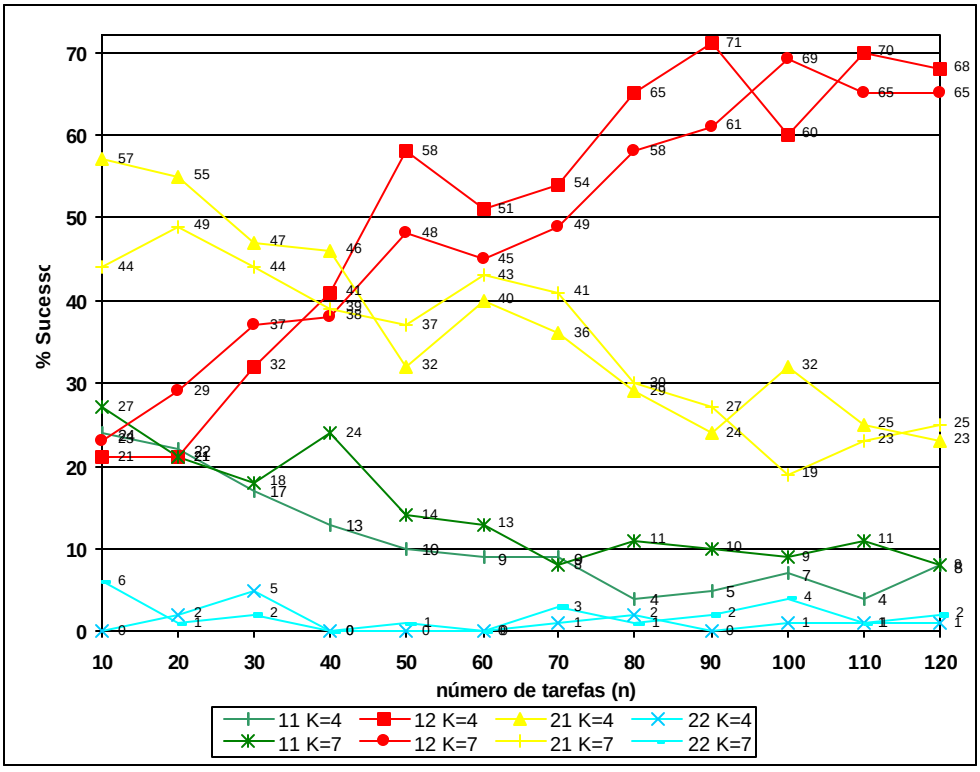


FIGURA 4.14 – Comparação da porcentagem de sucesso entre os métodos – relação V

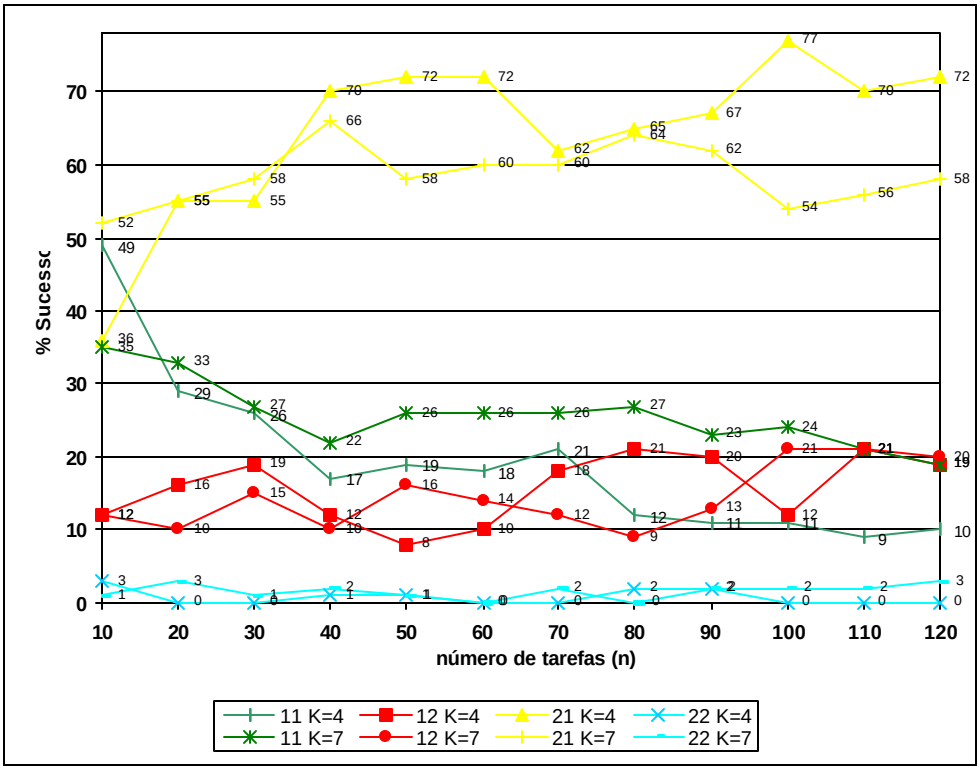


FIGURA 4.15 – Comparação da porcentagem de sucesso entre os métodos – relação VI

Os gráficos das figuras 4.10 a 4.15 mostram que em geral, com o aumento do número de tarefas, o método 12 melhora o seu desempenho e o método 11 piora levemente o desempenho. O método 21 melhora o desempenho nas relações III e VI, piora nas relações I, IV e V, e apresenta certa variação na amplitude do desempenho na relação II (de 1 a 25%). Para todas as relações $O(p_i)/O(s_{ij})$ e portes de problema, o método 22 obteve desempenho inferior aos outros métodos, com um considerável número de casos em que o desempenho foi 0% de vitórias. Algumas destas informações também podem ser vistas no gráfico da figura 4.16, que apresenta a comparação entre os métodos agregando as relações $O(p_i)/O(s_{ij})$.

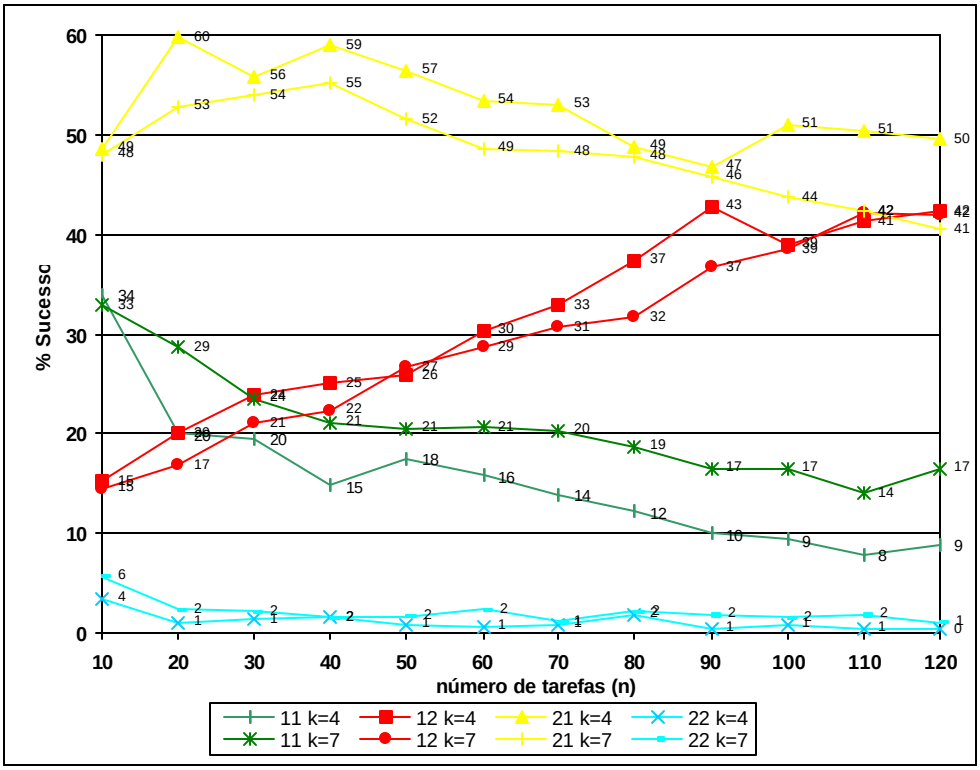


FIGURA 4.16 – Comparação da porcentagem de sucesso entre os métodos agregando as relações $O(p_i)/O(s_{ij})$

Para as relações I, IV e V, o método 21 possui desempenho superior em problemas de pequeno porte (10 a 40 tarefas) e o método 12 é melhor para problemas de médio e grande porte (60 a 120 tarefas). O método 21 mostrou-se superior em quase todos os portes de problemas para as relações II (com exceção dos problemas com 10 e 110 tarefas e 7 estágios) e VI (exceção de problemas com 10 tarefas e 4 estágios). Para a relação III, método 21 também obteve desempenho superior para todas as classes de problemas.

Para a relação III, a amplitude da variação do desempenho dos métodos, em função do número de tarefas, não é grande (em média, em torno de 16%) e a ordem de superioridade no desempenho se mantém constante: primeiro o método 21, em seguida o 11, depois o 12 e por fim, o 22.

Em geral, as curvas de desempenho dos problemas com 4 e 7 estágios mantêm o mesmo comportamento, podendo indicar que o número de estágios não afeta o desempenho de um método.

A figura 4.16, com as relações $O(p_i)/O(s_{ij})$ agregadas, mostra que predomina a superioridade no desempenho dos métodos 12 e 21 para 4 estágios e dos métodos 11 e 22 para 7 estágios. Entretanto, os gráficos sem agregação das figuras 4.10 a 4.15 mostram que a superioridade no desempenho dos métodos se alterna para as duas opções de número de estágios.

Para um mesmo método, a comparação entre as relações $O(p_i)/O(s_{ij})$ é apresentada nos gráficos das figuras 4.17 a 4.20, agregando-se o número de estágios. Nos métodos 12 e 21, a amplitude das diferenças de desempenho das relações aumenta consideravelmente com o aumento do número de tarefas. No método 22, essa amplitude diminui e no método 11, mantém-se praticamente constante em torno de 16 a 19%.

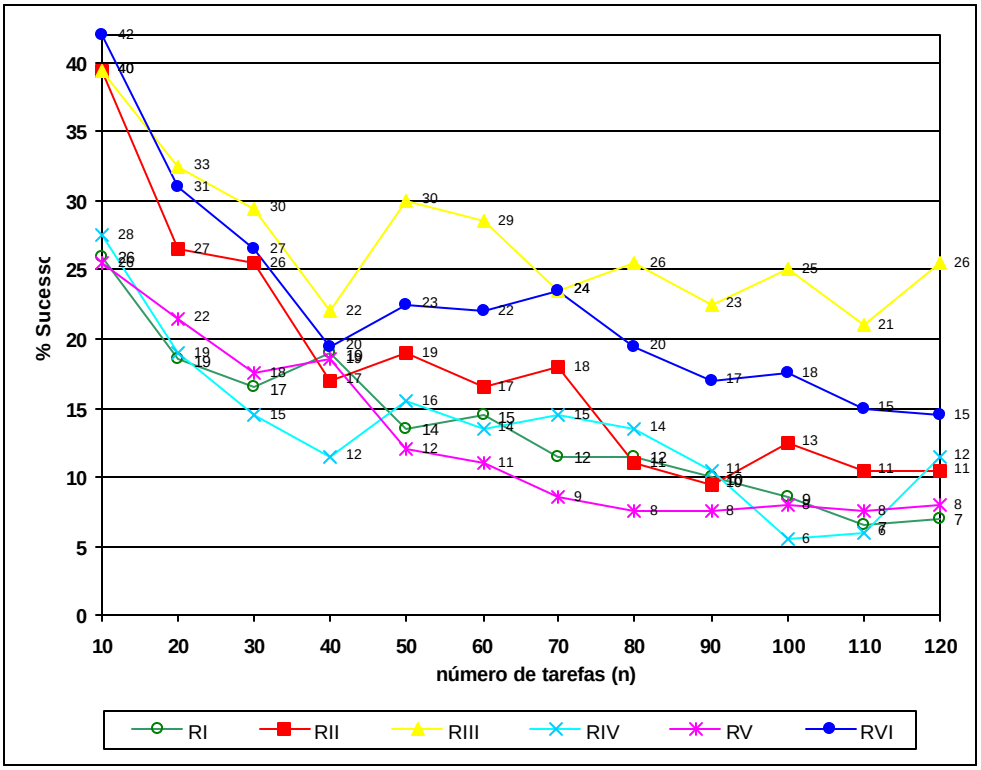


FIGURA 4.17 – Comparação da porcentagem de sucesso entre as relações $O(p_i)/O(s_{ij})$ agregando o número de estágios – método 11

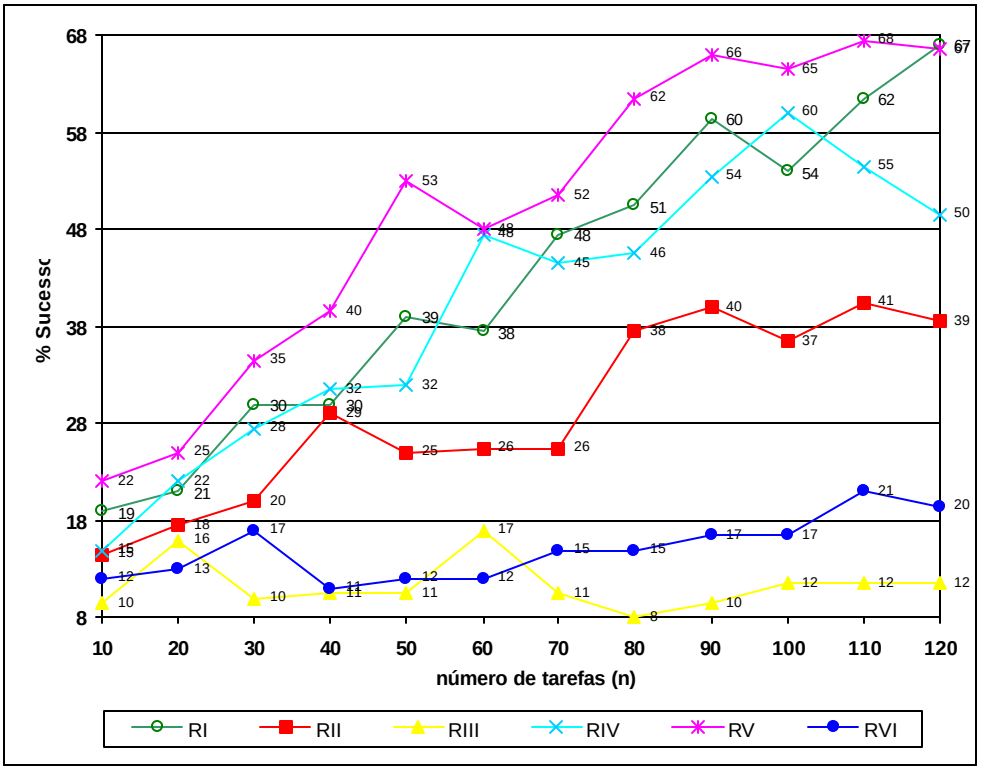


FIGURA 4.18 – Comparação da porcentagem de sucesso entre as relações $O(p_i)/O(s_{ij})$ agregando o número de estágios – método 12

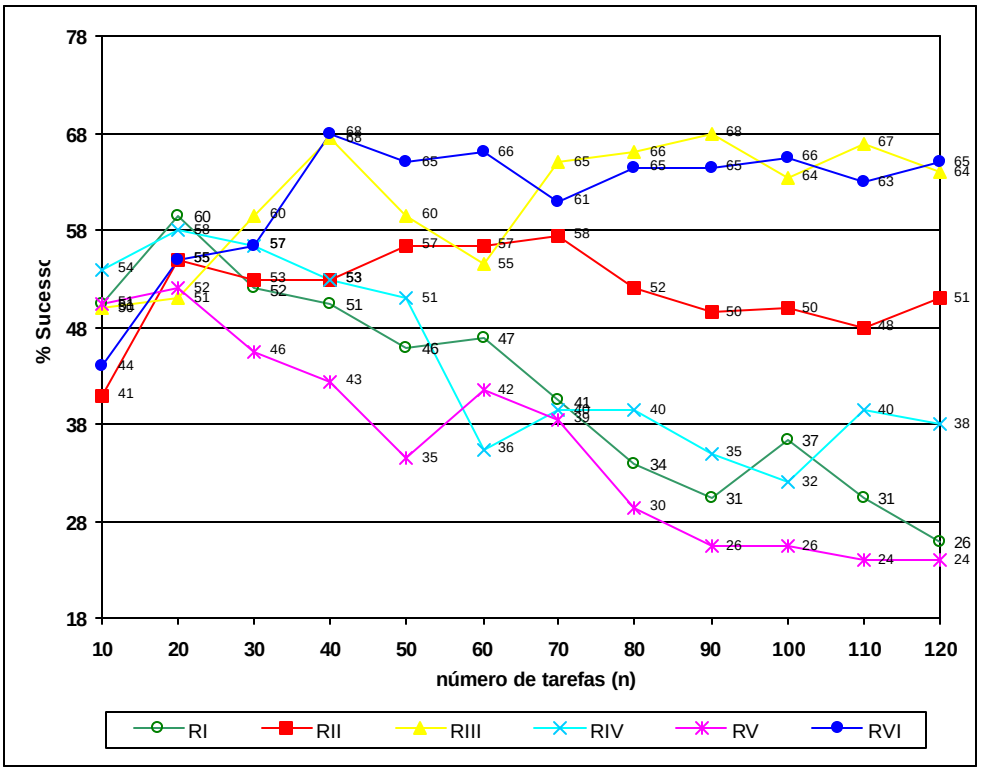


FIGURA 4.19 – Comparação da porcentagem de sucesso entre as relações $O(p_i)/O(s_{ij})$ agregando o número de estágios – método 21

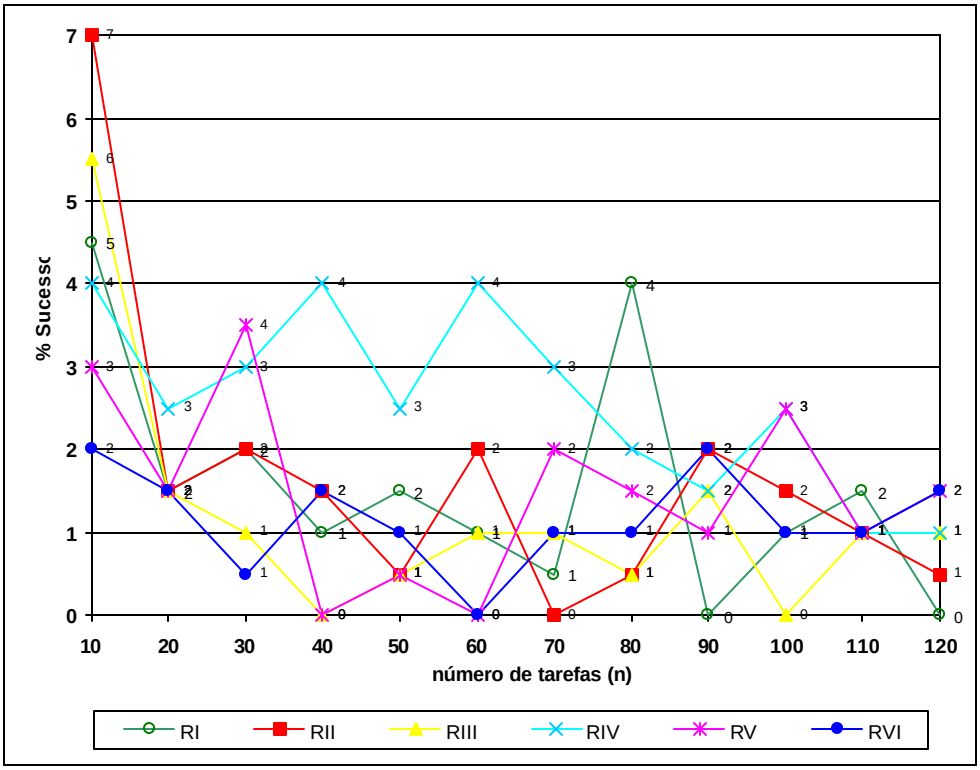


FIGURA 4.20 – Comparação da porcentagem de sucesso entre as relações $O(p_i)/O(s_{ij})$ agregando o número de estágios – método 22

Como o método 22 teve muita freqüência de 0% de vitórias, o gráfico comparativo entre relações apresenta muitos picos e vales, embora sejam de pequena amplitude (no máximo 5%).

Nos gráficos das figuras C.1 e C.2, do apêndice C, pode-se notar que, tanto para 4 como para 7 estágios, com 20, 30 e 40 tarefas, o desempenho do método 21 é superior em todas as relações. Para o número de tarefas acima de 40, a superioridade é disputada com o método 12, dependendo da relação $O(p_i)/O(s_{ij})$.

Na totalidade dos problemas resolvidos, o método 21 apresentou a melhor solução 7.130 vezes num total de 14.400 problemas, correspondendo a 49,5% de sucesso. Em segundo lugar, o método 12 obteve a melhor solução em 4.531 problemas, equivalente a 31,5% de sucesso. Em seguida, o método 11 forneceu a melhor solução 2581 vezes, ou seja, atingiu 17,9% de sucesso. E, por último, 232 problemas tiveram a melhor solução pelo método 22, com 1,6% de sucesso.

Em seguida, serão apresentadas as comparações e análises do desvio relativo e do desvio-padrão do desvio relativo. O apêndice C apresenta os gráficos gerais para 4 e 7 estágios.

Os gráficos das figuras 4.21 a 4.26 mostram a comparação dos desvios relativos médios em porcentagem entre os quatro métodos de solução para as seis relações de tempos de processamento e de *setup*.

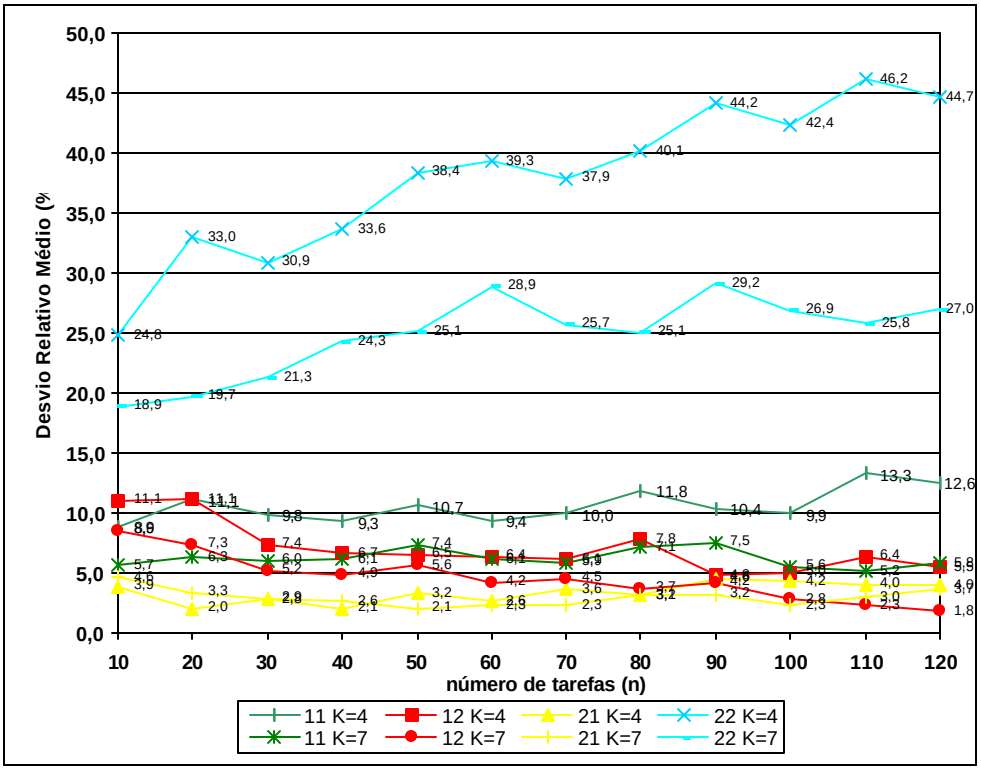


FIGURA 4.21 – Comparação do desvio relativo médio (%) entre os métodos – relação I

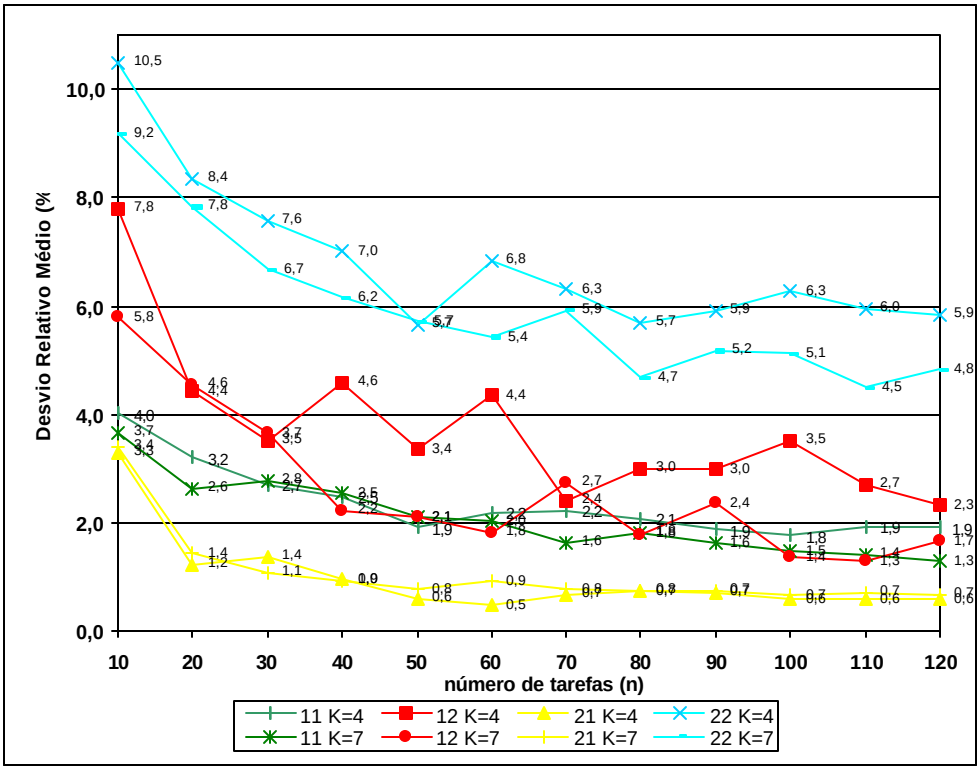


FIGURA 4.22 – Comparação do desvio relativo médio (%) entre os métodos – relação II

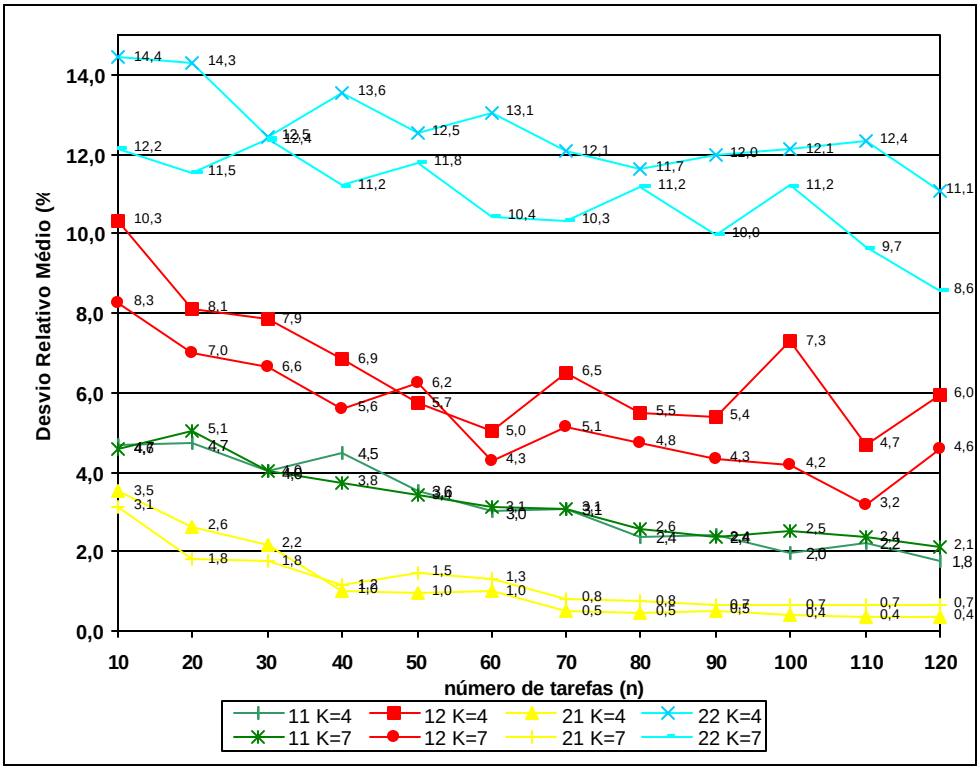


FIGURA 4.23 – Comparação do desvio relativo médio (%) entre os métodos – relação III

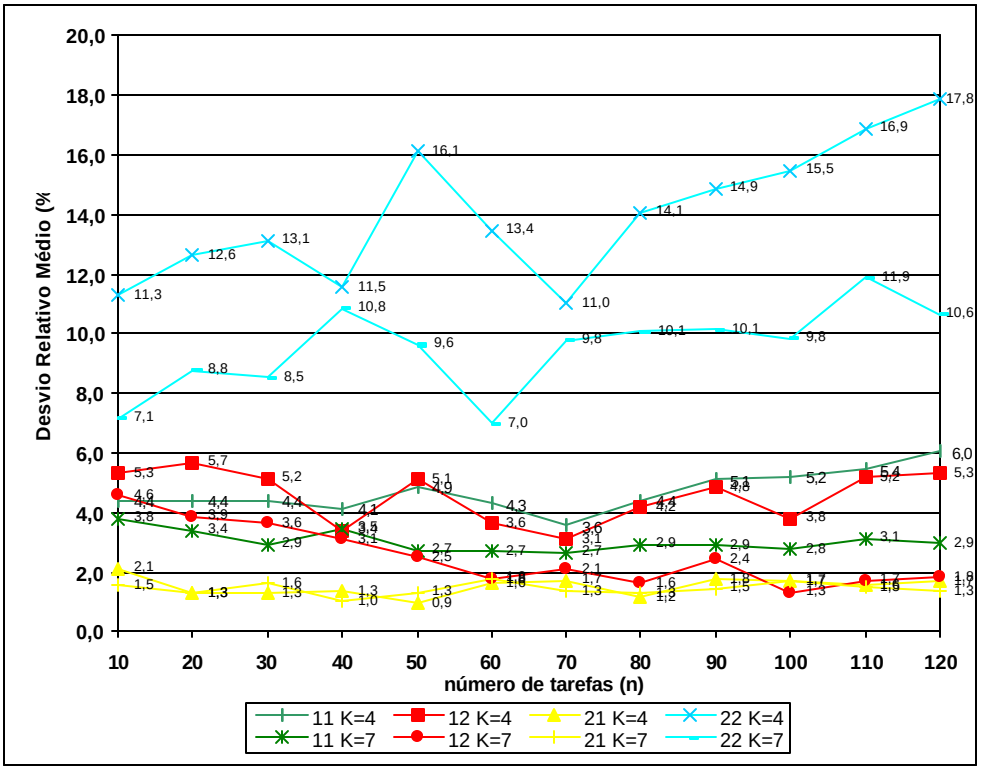


FIGURA 4.24 – Comparação do desvio relativo médio (%) entre os métodos – relação IV

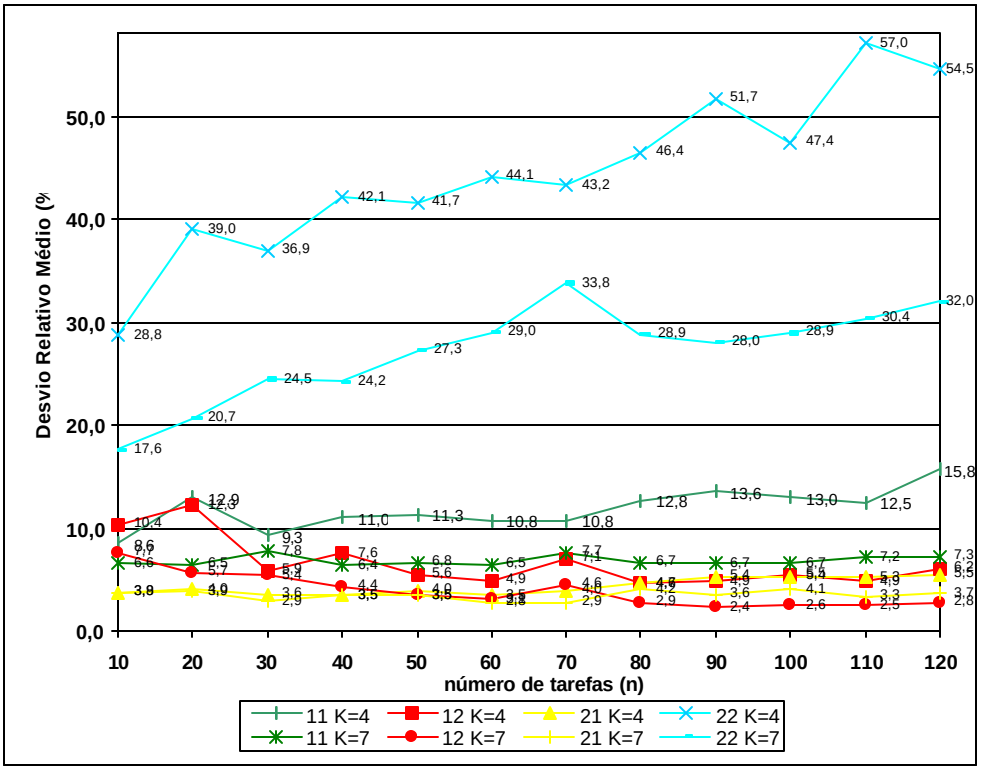


FIGURA 4.25 – Comparação do desvio relativo médio (%) entre os métodos – relação V

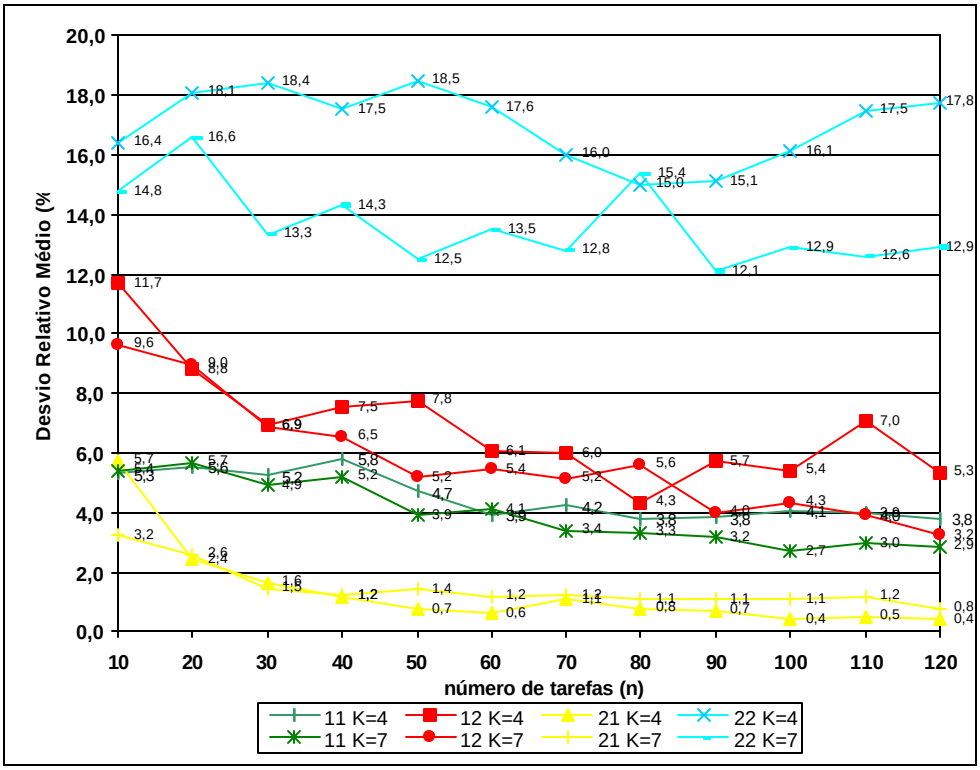


FIGURA 4.26 – Comparação do desvio relativo médio (%) entre os métodos – relação VI

Os valores dos desvios relativos confirmam a análise feita para a porcentagem de sucesso dos métodos. O método 21, com bom desempenho na porcentagem de sucesso, teve os menores valores de desvio relativo.

Os valores dos desvios relativos do método 22, que teve o pior desempenho na porcentagem de sucesso, são bastante elevados, principalmente nas relações I e V. Isto indica que além de fornecer a pior solução, ela possui também um valor muito acima (pior) da melhor solução obtida.

Os gráficos mostram que a tendência nas relações I, IV e V, sem considerar o método 22, é de relativa estabilidade dos valores dos desvios relativos com o aumento do número de tarefas.

Com as relações II, III e VI, a tendência é de diminuição dos desvios relativos com o aumento do número de tarefas. Isto significa que, à medida que o porte do problema aumenta, as diferenças nos desempenhos dos métodos tendem a diminuir.

Da mesma forma que a figura 4.12, que mostra a comparação da porcentagem de sucesso entre os métodos na relação III, o gráfico da figura 4.23 apresenta certa estabilidade no desempenho dos métodos também na relação III.

A tabela 4.28 apresenta o número de vitórias de um método em termos de desvio relativo, ou seja, o número de vezes que o seu valor foi inferior em relação aos outros métodos. A tabela 4.29 totaliza o número de vitórias e a porcentagem.

TABELA 4.28 – Número de vitórias do desvio relativo médio para problemas com 4 e 7 estágios

K = 4				K = 7			
11	**12**	**21**	**22**	**11**	**12**	**21**	**22**
1	3	68	0	0	9	63	0

TABELA 4.29 – Total geral do número de vitórias do desvio relativo médio

	11	**12**	**21**	**22**
total de problemas	1	12	131	0
% total média	0,7	8,3	91,0	0,0

Os gráficos das figuras 4.27 a 4.32 apresentam a comparação do desvio-padrão do desvio relativo entre os métodos de solução para as seis relações de tempos de processamento e de *setup*.

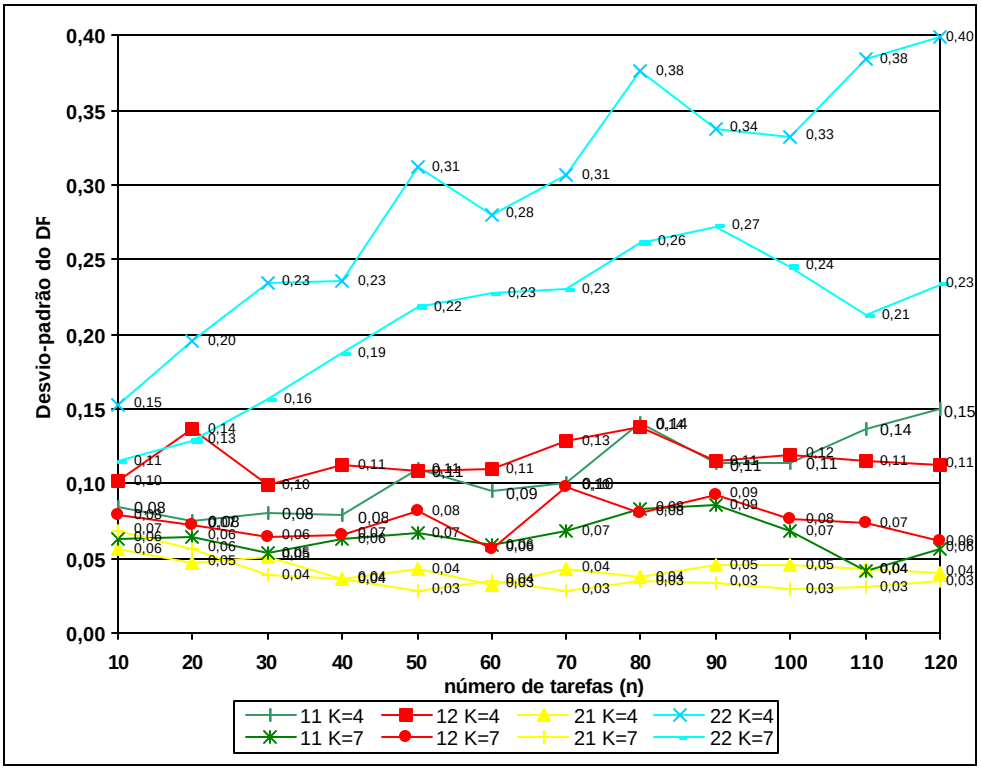


FIGURA 4.27 – Comparação do desvio-padrão do DR entre os métodos – relação I

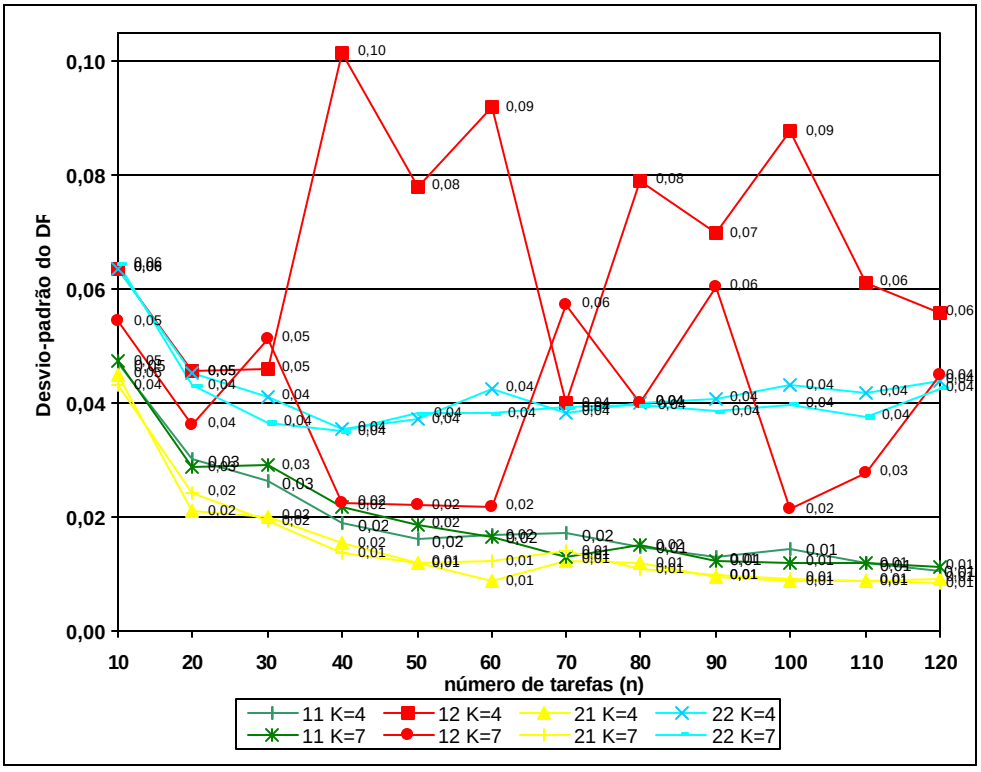


FIGURA 4.28 – Comparação do desvio-padrão do DR entre os métodos – relação II

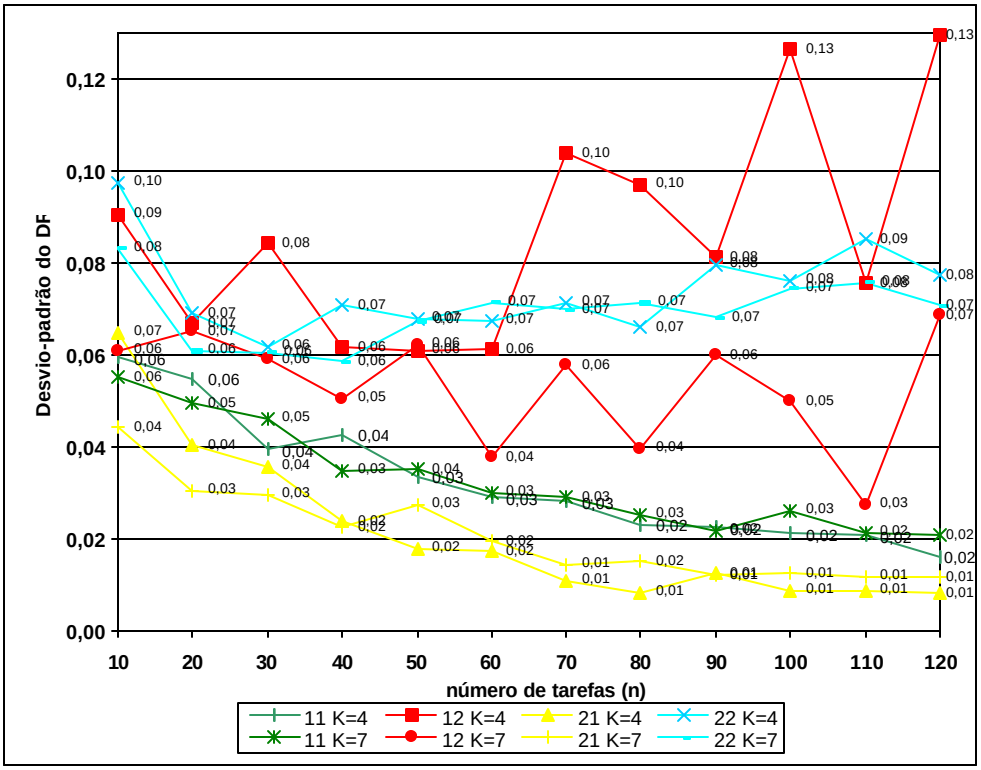


FIGURA 4.29 – Comparação do desvio-padrão do DR entre os métodos – relação III

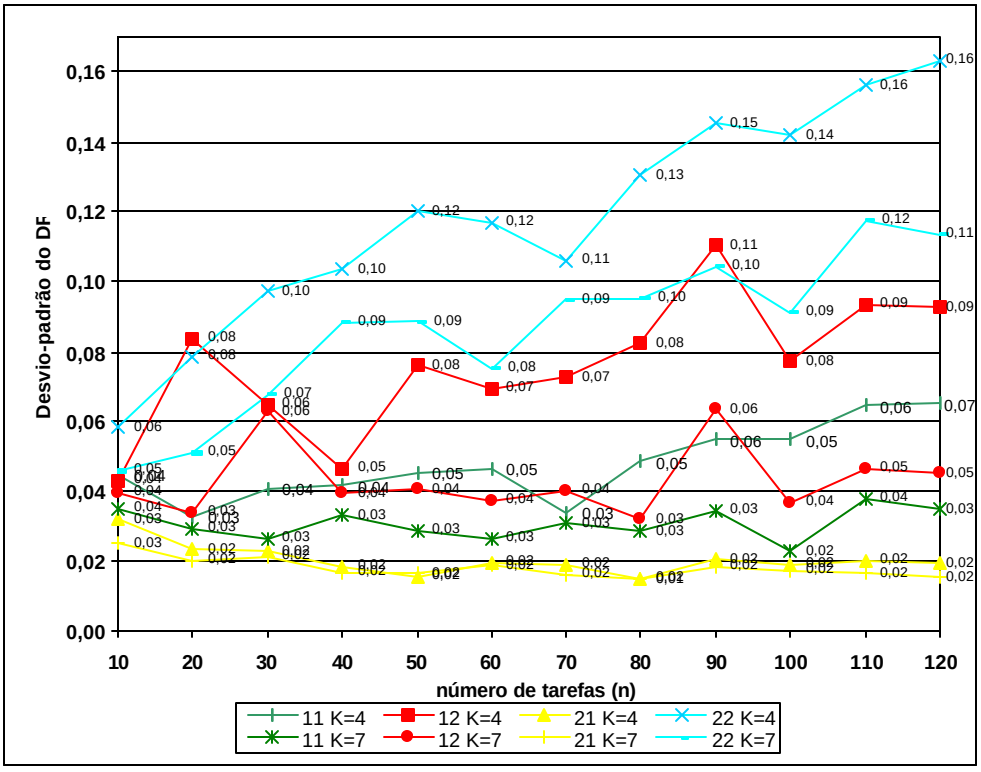


FIGURA 4.30 – Comparação do desvio-padrão do DR entre os métodos – relação IV

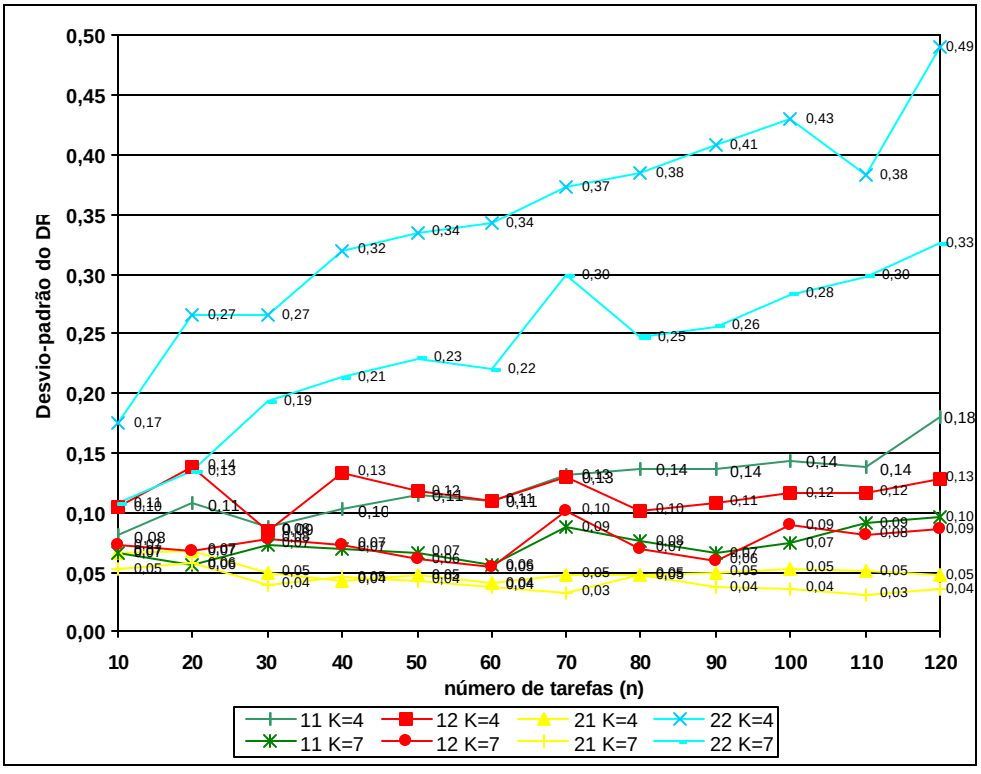


FIGURA 4.31 – Comparação do desvio-padrão do DR entre os métodos – relação V

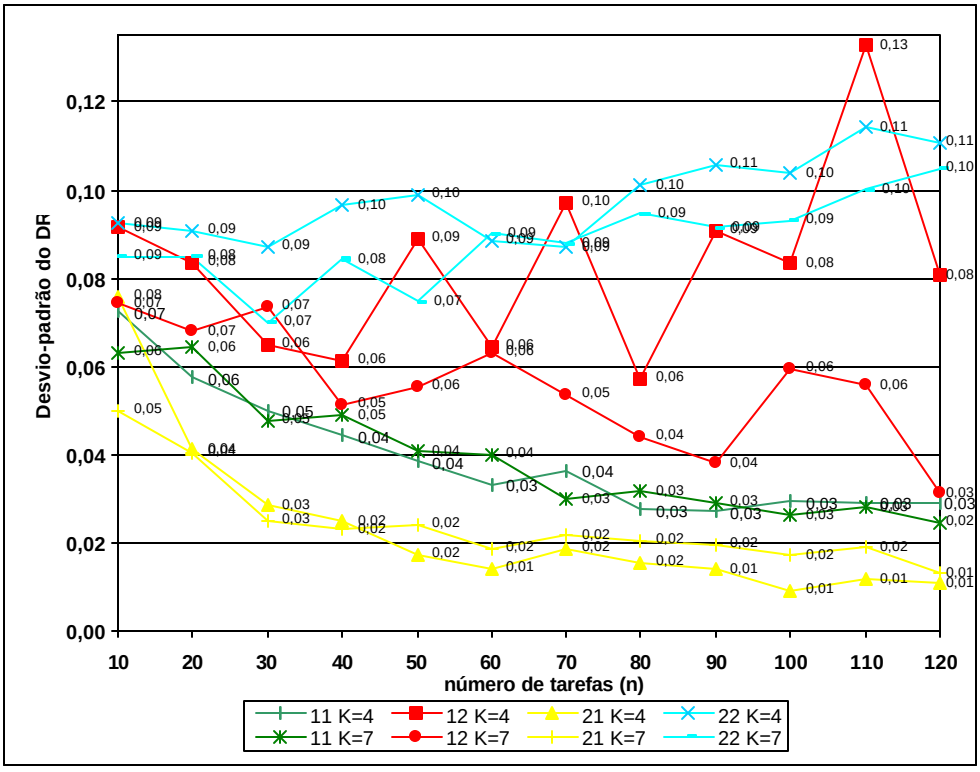


FIGURA 4.32 – Comparação do desvio-padrão do DR entre os métodos – relação VI

O desvio-padrão do DR é uma medida de comparação complementar que identifica qual método obteve desempenho superior quando os valores dos desvios relativos são próximos. Nesta análise, o desvio-padrão indica o quanto um método é estável em relação ao desvio relativo.

Os gráficos das figuras 4.27 a 4.32 mostram uma certa estabilidade na curva do método 21. Existe uma instabilidade considerável do método 12, principalmente nas relações II e III, e para problemas de maior porte na relação VI.

A tabela 4.30 apresenta o número de vitórias de um método em termos do desvio-padrão do DR, ou seja, o número de vezes que o desvio-padrão foi inferior ao dos outros métodos. A tabela 4.31 totaliza o número de vitórias e a porcentagem.

TABELA 4.30 – Número de vitórias do desvio-padrão do DR para problemas com 4 e 7 estágios

K = 4				K = 7			
11	**12**	**21**	**22**	**11**	**12**	**21**	**22**
2	0	70	0	3	0	69	0

TABELA 4.31 – Total geral do número de vitórias do desvio-padrão do DR

	11	12	21	22
total de problemas	5	0	139	0
% total média	3,5	0,0	96,5	0,0

O gráfico da figura 4.33 apresenta o tempo de computação dos quatro métodos de solução para problemas com 4 e 7 estágios. Como as curvas de cada método possuem comportamento similar, foi apresentado o gráfico com as relações dos tempos de processamento e de *setup* agregadas.

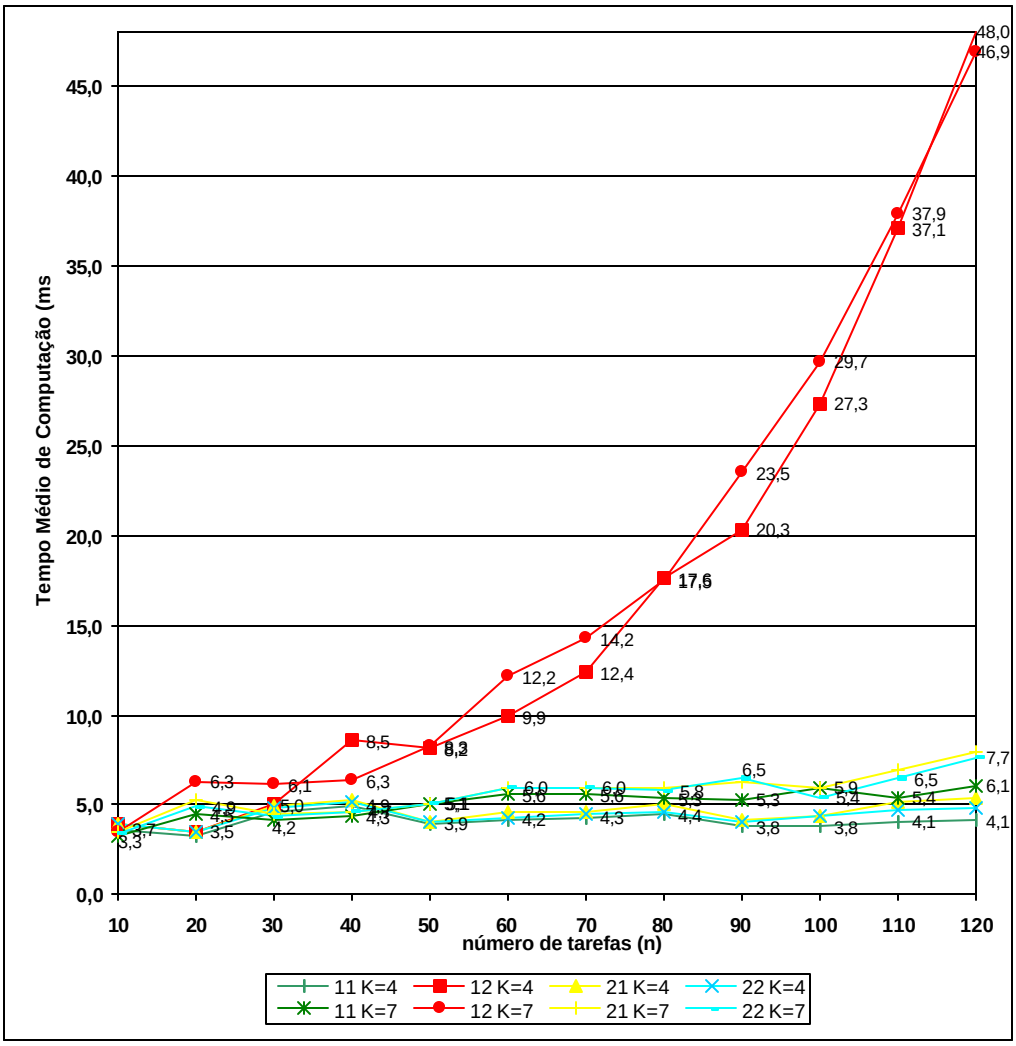


FIGURA 4.33 – Comparação do tempo médio de computação (ms) para 4 e 7 estágios agregando as relações $O(p_i)/O(s_{ij})$

Os gráficos das figuras 4.34 a 4.37 mostram a comparação dos tempos médios de computação das seis relações de tempo de processamento e de *setup*, agregando o número de estágios, para cada um dos métodos de solução. No apêndice C, são apresentados os gráficos gerais dos tempos médios de computação para problemas com 4 e 7 estágios e todas as relações de tempo de processamento e de *setup* (figuras C.7 e C.8).

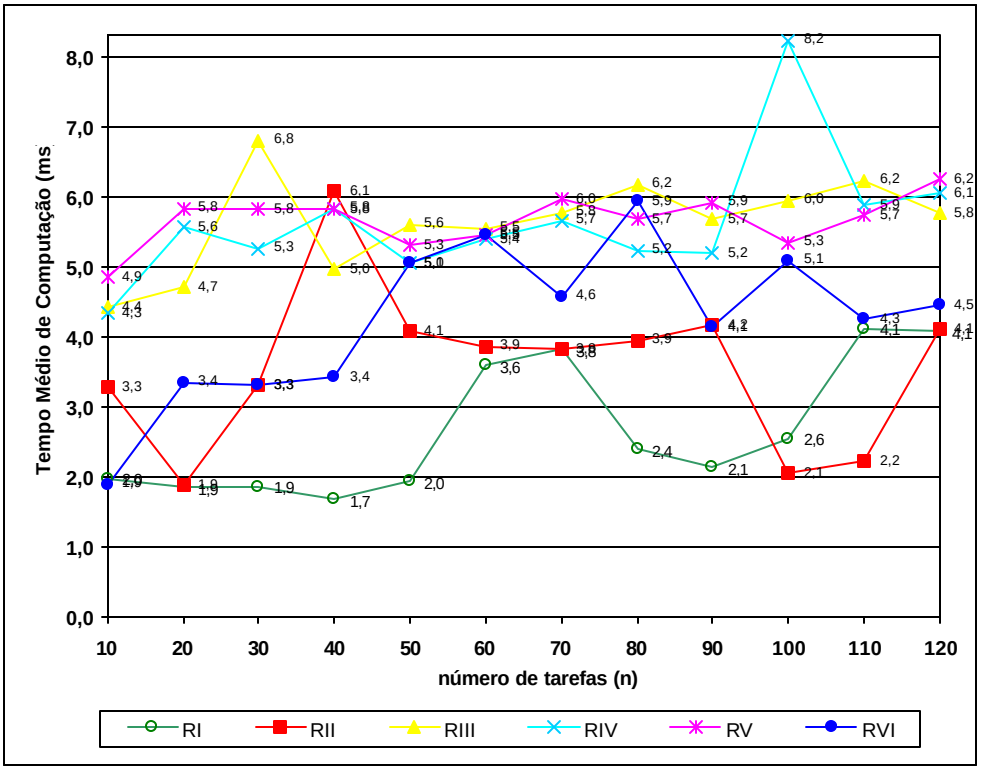


FIGURA 4.34 – Comparação do tempo médio de computação (ms) entre as relações $O(p_i)/O(s_{ij})$ agregando o número de estágios – método 11

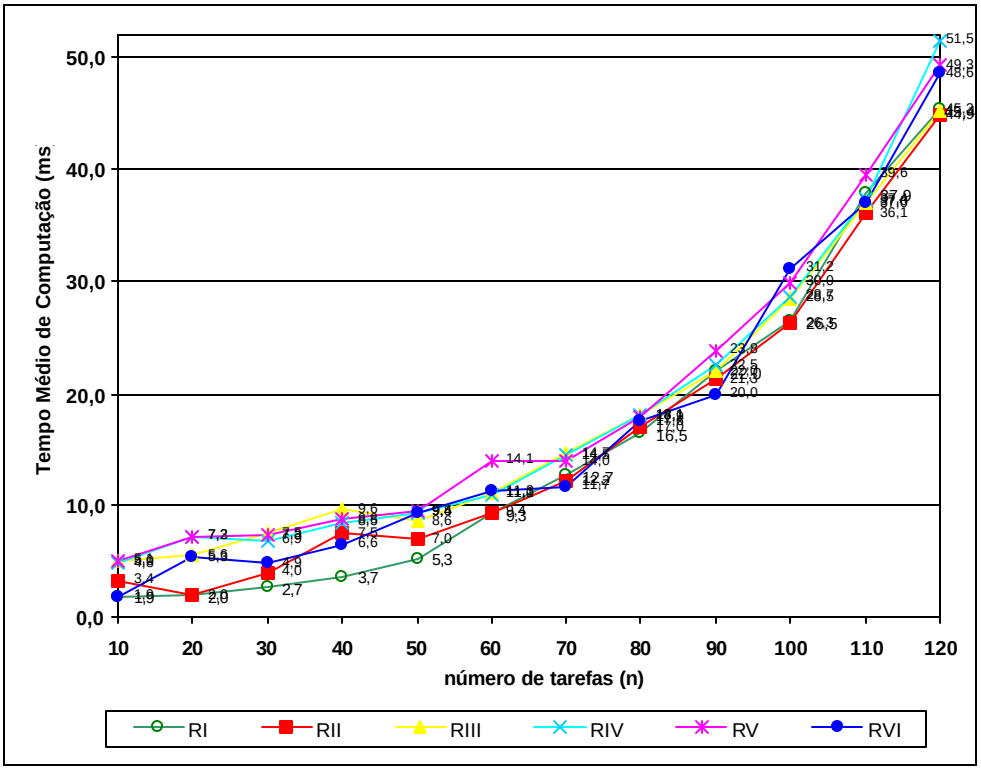


FIGURA 4.35 – Comparação do tempo médio de computação (ms) entre as relações $O(p_i)/O(s_{ij})$ agregando o número de estágios – método 12

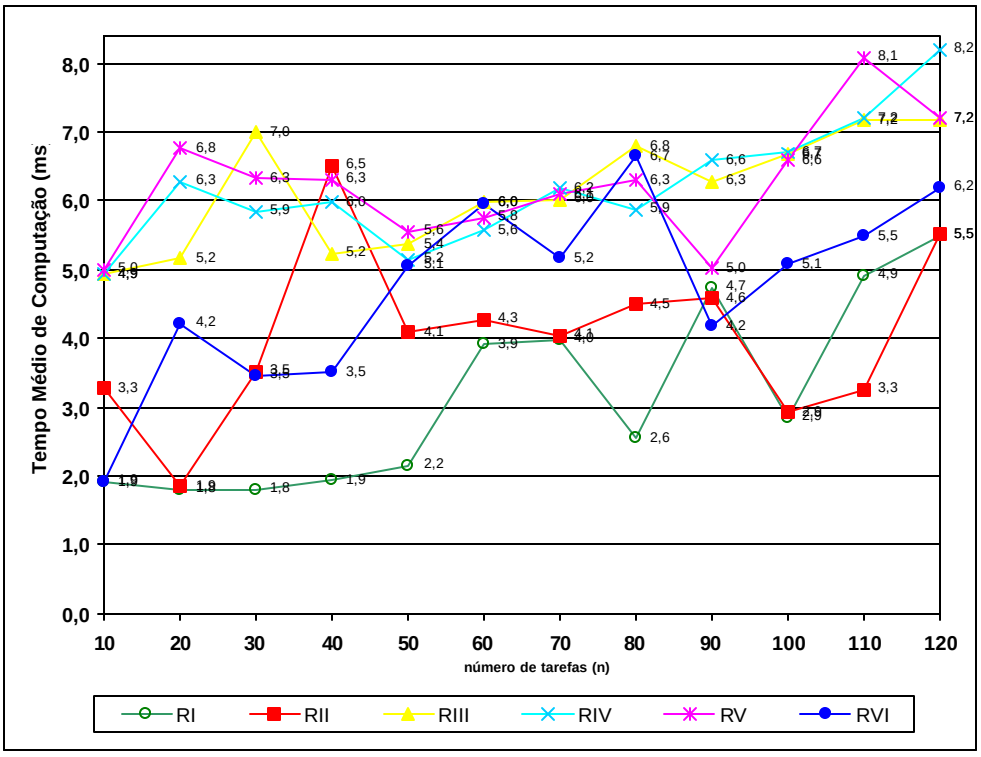


FIGURA 4.36 – Comparação do tempo médio de computação (ms) entre as relações $O(p_i)/O(s_{ij})$ agregando o número de estágios – método 21

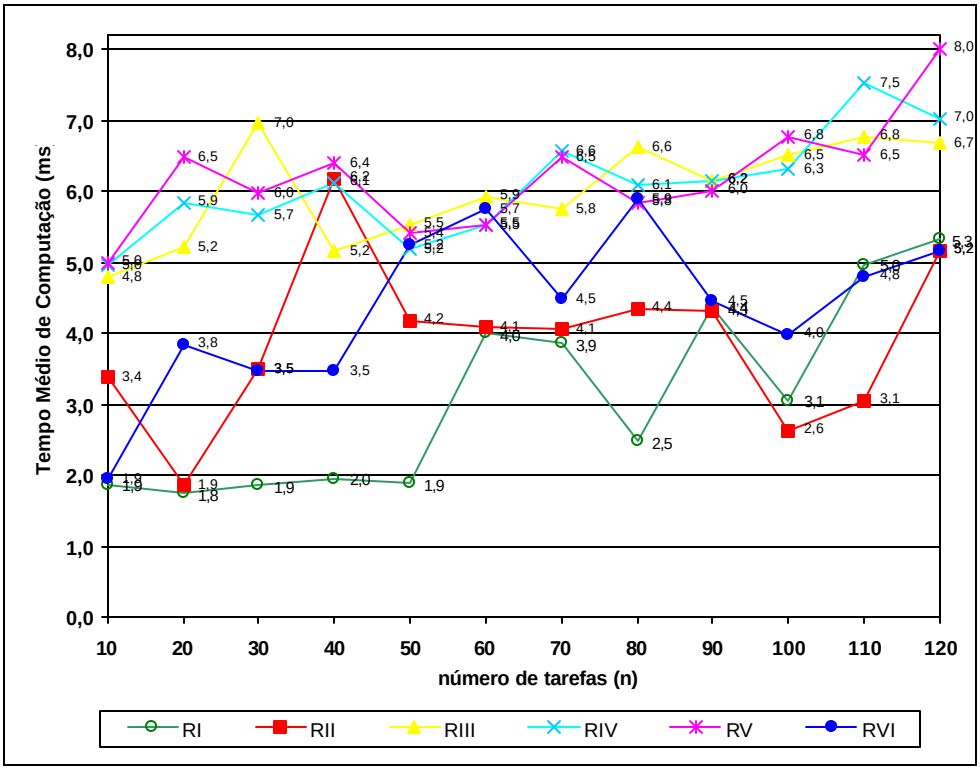


FIGURA 4.37 – Comparação do tempo médio de computação (ms) entre as relações $O(p_i)/O(s_{ij})$ agregando o número de estágios – método 22

Em todos os casos, como poderia se esperar, o tempo de computação torna-se maior com o aumento do número de tarefas. O método 12 tem o crescimento mais acentuado, o que pode ser explicado pela manipulação de matrizes, utilizada no Método de Simons Jr. (1992) e implementada no código computacional. O gráfico da figura 4.35 mostra o crescimento exponencial do tempo de computação do método 12 para todas as relações de tempos de processamento e de *setup*.

A implementação computacional dos outros métodos, que utiliza comandos semelhantes e códigos de tamanhos parecidos, produz crescimento similar no tempo de processamento do programa. A variação que pode ser observada nos gráficos das figuras 4.34, 4.36 e 4.37 deve-se à precisão da unidade de medida do tempo de computação (milissegundos), pois a amplitude da variação de uma relação não ultrapassa 5 ms. Além disso, em cada problema o número de tarefas a serem analisadas na designação a uma determinada máquina varia aleatoriamente.

A tabela 4.32 apresenta uma comparação geral do desempenho dos métodos em todas as relações $O(p_i)/O(s_{ij})$ e para 4 e 7 estágios com, base na porcentagem de sucesso.

Os valores dos desvios relativos e do desvio-padrão dos métodos reiteram os resultados da porcentagem de sucesso. Esta tabela recomenda um determinado método em função do número de tarefas, número de estágios e relação $O(p_i)/O(s_{ij})$.

TABELA 4.32 – Resumo do desempenho dos métodos em termos de porcentagem de sucesso

	K = 4						K = 7					
n	RI	RII	RIII	RIV	RV	RVI	RI	RII	RIII	RIV	RV	RVI
10	21	21	21	21	21	**11**	21	**11**	21	21	21	21
20	21	21	21	21	21	21	21	21	21	21	21	21
30	21	21	21	21	21	21	21	21	21	21	21	21
40	21	21	21	21	21	21	21	21	21	21	21	21
50	**12** 21	21	21	21	**12**	21	21	21	21	21	**12**	21
60	21	21	21	**12**	**12**	21	21	21	21	**12**	**12**	21
70	**12**	21	21	**12**	**12**	21	**12**	21	21	**12**	**12**	21
80	**12**	21	21	**12**	**12**	21	**12**	21	21	**12**	**12**	21
90	**12**	21	21	**12**	**12**	21	**12**	21	21	**12**	**12**	21
100	**12**	21	21	**12**	**12**	21	**12**	21	21	**12**	**12**	21
110	**12**	21	21	**12**	**12**	21	**12**	**12**	21	**12**	**12**	21
120	**12**	21	21	**12**	**12**	21	**12**	21	21	**12**	**12**	21

Observações sobre o método 12:

- é instável em termos de desvio relativo, principalmente nas relações II e III, e para problemas de maior porte na relação VI;
- obteve apenas 8,3% de vitórias sobre a quantidade total de classes de problemas em termos do desvio relativo, contra 91% de vitórias do método 21.

Observações sobre o método 11:

- obteve o melhor desempenho em somente duas situações: apenas para problemas com 10 tarefas (pequeno porte) nas relações VI (com 4 estágios) e II (com 7 estágios).

Observações sobre o método 22:

- apesar do método 22 ser uma modificação do método 21, os resultados de ambos foram bem diferentes. O pior desempenho do método 22 pode ser explicada pela característica da sua regra de alocação, que acaba priorizando a alocação de tarefas com maior tempo de *setup*.

5 CONCLUSÕES

Com base na análise dos resultados obtidos na experimentação computacional, verifica-se que em geral o método 21, em que o procedimento não utiliza ordenação inicial e observa a regra de alocação SCT, forneceu os melhores resultados em termos de porcentagem de sucesso, principalmente em problemas de 10 a 40 tarefas, para todas as relações $O(p_i)/O(s_{ij})$.

Em problemas com número de tarefas acima de 40, a superioridade do desempenho é disputada com o método 12, que utiliza ordenação inicial com a regra de prioridade TOTAL, dependendo da relação entre os tempos de processamento e de *setup*.

Tanto para 4 como 7 estágios, os resultados do método 21 foram superiores para as relações II, III e VI, cujos intervalos de tempos de processamento possuem amplitudes diferentes dos intervalos dos tempos de *setup* (1-99 e 100-120, 10-99 e 1-9, 1-99 e 1-20, respectivamente).

Para as relações I, IV e V, em que as amplitudes dos intervalos dos tempos de processamento e de *setup* são próximas (1-99 e 1-99, 50-99 e 1-49, 1-99 e 1-120, respectivamente), o método 21 apresentou os melhores resultados em problemas de pequeno porte (10 a 40 tarefas). Para problemas de médio e grande porte (60 a 120 tarefas), o desempenho do método 12 foi superior. Entretanto, em termos de desvio relativo, o método 12 é instável, principalmente nas relações II e III, e obteve apenas 8,3% de vitórias sobre a quantidade total de classes de problemas, contra 91% de vitórias do método 21.

O método 11, cujo procedimento utiliza ordenação inicial com regra de prioridade LPT, piora levemente o seu desempenho com o aumento do número de tarefas.

Para a relação III, em que os tempos de processamento são de 10 a 99 e os tempos de *setup* são de 1 a 9, a amplitude da variação do desempenho dos métodos, em função do número de tarefas, fica em torno de 16%, mostrando certa estabilidade, e a ordem de superioridade se mantém constante: primeiro o método 21, em seguida o 11, depois o 12 e por fim, o 22, conforme discorrido na seção 4.3 (figura 4.12).

Para todas as relações $O(p_i)/O(s_{ij})$ e portes de problema, o método 22, que utiliza a regra de alocação SCT/LPST sem ordenação inicial de tarefas, obteve desempenho inferior aos outros métodos, com um considerável número de casos em que o desempenho foi 0% de vitórias.

Em geral, as curvas de desempenho dos problemas com 4 e 7 estágios mantêm o mesmo comportamento, podendo indicar que o número de estágios não afeta o desempenho de um método e que problemas com outras opções do número de estágios teriam resultados similares.

As análises dos desvios relativos e do desvio-padrão do desvio relativo confirmam as conclusões dos resultados da porcentagem de sucesso, acrescentando a informação da instabilidade do método 12. Sem considerar o método 22, também pode ser observada uma relativa estabilidade dos valores dos desvios relativos com o aumento do número de tarefas nas relações I, IV e V. Com as relações II, III e VI, à medida que o porte do problema aumenta, há a tendência de diminuição dos desvios relativos, ou seja, as diferenças nos desempenhos dos métodos tendem a diminuir, conforme pode ser observado nos gráficos das figuras 4.21 a 4.26.

Em todos os casos, como poderia se esperar, o tempo de computação torna-se maior com o aumento do número de tarefas. O método 12 tem o crescimento mais acentuado (exponencial), refletindo a maior complexidade do seu código computacional. O tempo de processamento dos outros métodos, onde a implementação computacional utiliza comandos e códigos de tamanhos parecidos, possui crescimento semelhante.

Para o desenvolvimento de futuros trabalhos, sugere-se o estudo do ambiente de produção *flow shop* híbrido com tempos de preparação das máquinas separados dos tempos de processamento das tarefas (dependentes e/ou independentes da seqüência de execução) e a proposição de novos métodos heurísticos de solução, eventualmente com melhores desempenhos do que os propostos neste trabalho. Poderá também ser feita uma análise da estrutura dos problemas de programação para determinar propriedades e procedimentos que melhorem o desempenho dos algoritmos.

REFERÊNCIAS

ALDOWAISAN, T.; ALLAHVERDI, A. (2003). New heuristics for no-wait flowshops to minimize makespan. **Computers & Operations Research**, Oxford, v.30, n.8, p.1219-1231, jul.

ALLAHVERDI, A.; GUPTA, J.N.D.; ALDOWAISAN, T. (1999). A review of scheduling research involving setup considerations. **Omega – The International Journal of Management Science**, Oxford, v.27, p.219-239.

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

BALAKRISHNAN, N.; KANET, J.J.; SRIDHARAN, S.V. (1999). Early/tardy scheduling with sequence dependent setups on uniform parallel machines. **Computers & Operations Research**, Oxford, v.26, p.127-141.

BANK, J.; WERNER, F. (2001). Heuristic algorithms for unrelated parallel machine scheduling with a common due date, release dates, and linear earliness and tardiness penalties. **Mathematical and Computer Modeling**, Oxford, v.33, n.4-5, p.363-383, feb.-mar.

BARROS, A.D.; MOCCELLIN, J.V. (2003). Programação flow shop permutacional com tempos de *setup* assimétricos e dependentes da seqüência por meio de análise da flutuação do gargalo. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, XXXV., 2003, Natal. **Anais**... Rio de Janeiro: SOBRAPO. 1 CD-ROM.

BOTTA-GENOULAZ, V. (2000). Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. **International Journal of Production Economics**, Amsterdam, v.64, n.1-3, p.101-111, mar.

BRAUN, O. *et al*. (2002). Stability of Johnson's schedule with respect to limited machine availability. **International Journal of Production Research**, Oxon, v.40, n.17, p.4381-4400, nov.

BURBIDGE, J.L. (1975). **The introduction of group technology**. London: Heinemann.

BUZZO, W.R.; MOCCELLIN, J.V. (2000). Programação da produção em sistemas flow shop utilizando um método heurístico híbrido Algoritmo Genético-Simulated Anneling. **Gestão & Produção**, São Carlos, v.7, n.3, p.264-377, dez.

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, p.B630-637.

CHENG, T.C.E.; SIN, C.C.S. (1990). A state-of-the-art review of parallel-machine scheduling research. **European Journal of Operational Research**, Amsterdam, v.47, n.3, p.271-292, aug.

CHIU, N.; FANG, S.; LEE, Y. (1999). Sequencing parallel machining operations by genetic algorithms. **Computer & Industrial Engineering**, Exeter, v.36, p.259-280.

DANNENBRING, D.G. (1977). An evaluation of flow shop sequencing heuristics. **Management Science**, Linthicum, v.23, n.11, p.1174-1182.

DAS, S.R.; GUPTA, J.N.D.; KHUMAWALA, B.M. (1995). A saving index heuristic algorithm for flowshop scheduling with sequence dependent set-up times. **Journal of Operational Research Society**, Birmingham, v.46, p.1365-1373.

ERDMANN, R.H. (2000). **Administração da produção: planejamento, programação e controle**. Florianópolis: Papa Livro.

FLOOD, M.M. (1956). The Traveling-Salesman Problem. **Operations Research**, Linthicum, v.4, n.1, p.61-75, feb.

FRAMINAN, J.M.; LEISTEN, R.; RAJENDRAN, C. (2003). Different initial sequences for the heuristic of Nawaz, Enscore and Ham to minimize makespan, idletime or flowtime in the static permutation flowshop sequencing problem. **International Journal of Production Research**, Oxon, v.41, n.1, p.121-148.

GANGADHARAN, R.; RAJENDRAN, C. (1994). A Simulated Annealing heuristic for scheduling in a flowshop with bicriteria. **Computer & Industrial Engineering**, Exeter, v.27, n.1-4, p.473-476.

GAVETT, J.W. (1965). Three heuristic rules for sequencing jobs to a single production facility. **Management Science**, Linthicum, v.11, n.8, p.B166-B176.

GELDERS, L.F.; SAMBANDAM, N. (1978). Four simple heuristics for scheduling a flow-shop. **International Journal of Production Research**, London, v.16, n.3, p.221-231.

GUINET, A.G.P.; SOLOMON, M.M. (1996). Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time. **International Journal of Production Research**, London, v.34, n.6, p.1643-1654.

GUPTA, J.N.D. (1971). Functional heuristic algorithm for the flowshop scheduling problem. **Operational Research Quarterly**, Oxford, v.22, n.1, p.39-47.

GUPTA, J.N.D.; HO, J.C. (2001). Minimizing makespan subject to minimum flowtime on two identical parallel machines. **Computers & Operations Research**, Oxford, v.28, n.7, p.705-717, jun.

GUPTA, J.N.D.; RUIZ-TORRES, A.J. (2000). Minimizing makespan subject to minimum total flow-time on identical parallel machines. **European Journal of Operational Research**, Amsterdam, v.125, n.2., p.370-380, sep.

GUPTA, J.N.D.; TUNC, E.A. (1998). Minimizing tardy jobs in a two-stage hybrid flowshop. **International Journal of Production Research**, London, v.36, n.9, p.2397-2417, sep.

HAOUARI, M.; M'HALLAH, R. (1997). Heuristic algorithms for the two-stage hybrid flowshop problem. **Operations Research Letters** , Amsterdam, v.21, n.1, p.43-53, aug.

HARDING, H.A. (1981). **Administração da produção**. São Paulo: Atlas.

HUNDAL, T.S.; RAJGOPAL, J. (1988). An extension of Palmer Heuristic for the flow-shop scheduling problem. **International Journal of Production Research**, London, v.26, n.6, p.1119-1124, jun.

HO, H.C.; GUPTA, J.N.D. (1995). Flowshop scheduling with dominant machines. **Computers & Operations Research**, Oxford, v.22, n.2, p.237-246.

HUANG, W.; LI, S. (1998). A two-stage hybrid flowshop with uniform machines and setup times. **Mathematical and Computer Modeling** , Oxford, v.27, n.2, p.27-45, jan.

HUNSUCKER, J.L.; SHAH, J.R. (1994). Comparative performance analysis of priority rules in a constrained flow shop with multiple processors environment. **European Journal of Operational Research**, Amsterdam, v.72, p.102-114.

HURINK, J.; KNUST, S. (2001). List scheduling in a parallel machine environment with precedence constraints and setup times. **Operations Research Letters**, Amsterdam, v.29, n.5, p.231-239, dec.

IGNALL, E.; SCHRAGE, L. (1965). Application of the branch and bound technique to some flow-shop scheduling problems. **Operations Research**, Linthicum, v.13, n.3, p.400-412.

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 Logistic s Quarterly**, Hoboken, v.1, p.61-68.

KIM, D.W. *et al*. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. **Robotics and Computer Integrated Manufacturing**, Oxford, v.18, n.3-4, p.223-231, jun.-aug.

KIM, D.W.; NA, D.G.; CHEN, F.F. (2003). Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective. **Robotics and Computer Integrated Manufacturing**, Oxford, v.19, p.173-181.

KIM, J.S.; KANG, S.H.; LEE, S.M. (1997). Transfer batch scheduling for a two-stage flowshop with identical parallel machines at each stage. **Omega – The International Journal of Management Science** , Oxford, v.25, n.5, p.547-555, abr.

KIRKPATRICK, S.; GELATT JR., C.D.; VECCHI, M.P. (1983). Optimization by simulated annealing. **Science**, Washington, v.220, p.671-680, may.

KOULAMAS, C.; KYPARISIS, G.J. (2000). Scheduling uniform parallel machines to minimize maximum lateness. **Operations Research Letters**, Amsterdam, v.26, n.4, p.175-179, may.

KRAVCHENKO, S.A.; WERNER, F. (2001). A heuristic algorithm for minimizing mean flow time with unit setups. **Information Processing Letters**, Amsterdam, v.79, n.6, p.291-296, sep.

LANCIA, G. (2000). Scheduling jobs with release dates and tails on two unrelated parallel machines to minimize the makespan. **European Journal of Operational Research**, Amsterdam, v.120, n.2, p.277-288, jan.

LEE, Y.H.; PINEDO, M. (1997). Scheduling jobs on parallel machines with sequence-dependent setup times. **European Journal of Operational Research**, Amsterdam, v.100, p.464-474.

LEE, C.Y.; VAIRAKTARAKIS, G.L. (1994). Minimizing makespan in hybrid flowshops. **Operations Research Letters**, Amsterdam, v.16, n.3, p.149-158, oct.

LI, S. (1997). A hybrid two-stage flowshop with part family, batch production, major and minor set-ups. **European Journal of Operational Research**, Amsterdam, v.102, p.142-156.

LIAEE, M.M.; EMMONS, H. (1997). Scheduling families of jobs with setup times. **International Journal of Production Economics**, Amsterdam, v.51, n.3, p.165-176.

LIAO, C.J.; LIN, C.H. (2003). Makespan minimization for two uniform parallel machines. **International Journal of Production Economics**, Amsterdam, v.84, n.2, p.205-213, may.

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**, Amsterdam, v.86, n.2, p.133-143, nov.

LINN, R.; ZHANG, W. (1999). Hybrid flow shop scheduling: a survey. **Computers & Industrial Engineering**, Oxford, v.37, n.1-2, p.57-61, oct.

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

MANDEL, M.; MOSHEIOV, G. (2001). Minimizing maximum earliness on parallel identical machines. **Computers & Operations Research**, Oxford, v.28, p.317-327.

MOCCELLIN, J.V. (1995). A new heuristic method for the permutation flow-shop scheduling problem. **Journal of the Operational Research Society**, Oxford, v.46, n.7, p.883-886, jul.

MOCCELLIN, J.V.; NAGANO, M.S. (2003a). Flow shop híbrido com estágios gargalos. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, XXXV., 2003, Natal. **Anais**... Rio de Janeiro: SOBRAPO. 1 CD-ROM.

_____. (2003b). Flow shop com máquinas paralelas genéricas. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, XXXV., 2003, Natal. **Anais**... Rio de Janeiro: SOBRAPO. 1 CD-ROM.

MOCCELLIN, J.V.; SANTOS, M.O. (2000). An adaptive hybrid metaheuristic for permutation flowshop scheduling. **Journal of Control and Cybernetics**, Warsaw, v.29, n.3.

MORTON, T.E.; PENTICO, D.W. (1993). **Heuristic scheduling systems**. New York: John Wiley & Sons, Inc.

MOURSLI, O.; POCHET, Y. (2000). A branch-and-bound algorithm for the hybrid flowshop. **International Journal of Production Economics**, Amsterdam, v.64, n.1-3, p.113-125, mar.

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

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

NAWAZ, M.; ENSCORE JR., E.E.; HAM, I. (1983). A heuristic algorithm for de *m*-machine, *n*-job flow-shop sequencing problem. **Omega – The International Journal of Management Science**, Oxford, v.11, n.1, p.91-95.

OGUZ, C. *et al*. (2003). Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow-shop. **European Journal of Operational Research**, Amsterdam, v.149, n.2, p.390-403, sep.

OGUZ, C. *et al*. (2004). Hybrid flow-shop scheduling problems with multiprocessor task systems. **European Journal of Operational Research**, Amsterdam, v.152, n.1, p.115-131, jan.

OSMAN, I.H.; POTTS, C.N. (1989). Simulated annealing for permutation flow-shop scheduling. **Omega – The International Journal of Management Science**, Oxford,v.17, n.6, p.551-557.

PALMER, D.S. (1965). Sequencing jobs through a multi-stage process in the minimum total time – a quick method of obtaining a near optimum. **Operational Research Quarterly**, Oxford, v.16, n.1, p.101-107.

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

PIERSMA, N.; VANDIJK, W. (1996). A local search heuristic for unrelated parallel machine scheduling with efficient neighborhood search. **Mathematical and Computer Modelling**, Oxford, v.24, n.9, p.11-19, nov.

PINEDO, M. (1995). **Scheduling: theory, algorithms, and systems**. New Jersey: Prentice-Hall.

POTTS, C.N. (1980). An adaptive branching rule for the permutation flow-shop problem. **European Journal of Operational Research**, Amsterdam, v.5, n.1, p.19-25, jul.

RAJENDRAN, C.; ZIEGLER, H. (1997). A heuristic for scheduling to minimize the sum of weighted flowtime of jobs in a flowshop with sequence-dependent setup times of jobs. **Computers & Industrial Engineering**, Oxford, v.33, n.1-2, p.281-284, oct.

_____. (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, v.149, n.3, p.513-522, sep.

REDDY, V.; NARENDRAN, T.T. (2003). Heuristics for scheduling sequence-dependent set-up jobs in flow line cells. **International Journal of Production Research**, Oxon, v.41, n.1, p.193-206.

REEVES, C.R. (1995). A genetic algorithm for flowshop sequencing. **Computers & Operations Research**, Oxford, v.22, n.1, p.5-13, jan.

REINFELD, N.V.; VOGEL, W.R. (1958). **Mathematical Programming**. Englewood Cliffs: Prentice-Hall, Inc. Cap.4, p.59-70.

RIANE, F.; ARTIBA, A.; ELMAGHRABY, S.E. (1998). A hybrid three-stage flowshop problem: efficient heuristics to minimize makespan. **European Journal of Operational Research**, Amsterdam, v.109, p.321-329, sep.

_____. (2002). Sequencing a hybrid two-stage flowshop with dedicated machines. **International Journal of Production Research**, Oxon, v.40, n.17, p.4353-4380, nov.

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

_____. (1999). An enhanced TSP-based heuristic for makespan minimization in a flow shop with setup times. **Journal of Heuristics**, Boston, v.5, p.53-70.

ROBINSON, A. (1990). **Modern approaches to manufacturing improvement: The Shingo System**. Portland: Productivity Press.

RUIZ, R.; MAROTO, C. (2003). Hybrid flowshops with sequence dependent setup times: a general metaheuristic. Universidade Politécnica de Valência.

_____. (2004). A comprehensive review and evaluation of permutation flowshop heuristics. **European Journal of Operational Research**, *in press*.

RUIZ, R.; MAROTO, C.; ALCARAZ, J. (2004). Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. **European Journal of Operational Research**, *in press*.

SARIN, S.C.; HARIHARAN, R. (2000). A two machine bicriteria scheduling problem. **International Journal of Production Economics**, Amsterdam, v.65, p.125-139.

SCHMIDT, G. (2000). Scheduling with limited machine availability. **European Journal of Operational Research**, Amsterdam, v.121, n.1, p.1-15, feb.

SCHUTTEN, J.M.J. (1996). List scheduling revisited. **Operations Research Letters**, Amsterdam, v.18, p.167-170.

SELEN, W.J.; HOTT, D.D. (1986). A mixed-integer goal programming formulation of the standard flow-shop scheduling problem. **Journal of the Operational Research Society**, Oxford, v.37, n.12, p.1121-1128.

SIMONS JR., J.V. (1992). Heuristics in flow shop scheduling with sequence dependent setup times. **Omega – The International Journal of Management Science**, Oxford, v.20, n2, p.215-225.

SHINGO, S. (1996). **Sistemas de produção com estoque zero: o Sistema Shingo para melhorias contínuas** . trad. Lia Weber Mendes. Porto Alegre: Bookman.

SHORE, H.M. (1970). The transportation problem and the Vogel Approximation Method. **Decision Sciences**, v.1, p.441-457.

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

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

SRISKANDARAJAH, C.; SETHI, S.P. (1989). Scheduling algorithms for flexible flowshops: worst and average case performance. **European Journal of Operational Research**, Amsterdam, v.43, p.143-160.

STINSON, J.P.; SMITH, A.W. (1982). A heuristic programming procedure for sequencing the static flow shop. **International Journal of Production Research**, London, v.20, n.6, p.753-764.

SULIMAN, S.M.A. (2000). A two-phase heuristic approach to the permutation flow-shop scheduling problem. **International Journal of Production Economics**, Amsterdam, v.64, p.143-152.

SUNDARARAGHAVAN, P.S.; KUNNATHUR, A.S.; VISWANATHAN, I. (1997). Minimizing makespan in parallel flowshops. **Journal of the Operational Research Society**, Birmingham, v.48, p.834-842.

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

TOWNSEND, W. (1977). Sequencing *n* jobs on *m* machines to minimize maximum tardiness: a branch-and-bound solution. **Management Science**, Rhode Island, v.23, n.9, p.1016-1019, may.

VIGNIER, A.; BILLAUT, J.C.; PROUST, C. (1999). Les problèmes d'ordonnancement de type flow-shop hybride: état de l'art. **RAIRO – Recherche Opérationnelle**, Paris, v.33, n.2, p.117-183.

WENG, M.X.; LU, J.; REN, H. (2001). Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. **International Journal of Production Economics**, Amsterdam, v.70, n.3, p.215-226, apr.

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

WILSON, J.M. (1989). Alternative formulations of a flow shop scheduling problem. **Journal of the Operational Research Society**, Oxford, v.40, n.4, p.395-399, apr.

XIANG, S.; TANG, G.; CHENG, T.C.E. (2000). Solvable cases of permutation flowshop scheduling with dominating machines. **International Journal of Production Economics**, Amsterdam, v.66, n.1, p.53-57, jun.

YANG, D.L.; CHERN, M.S. (2000). Two-machine flowshop group scheduling problem. **Computers & Operations Research**, Oxford, v.27, p.975-985.

GLOSSÁRIO

Dead line – prazo final de entrega de uma tarefa que se for atingido anula o processamento já realizado.

Due date – data de entrega ou prazo de término de uma tarefa.

Earliness – antecipação relativa ao prazo de término de uma tarefa, expressa por $E_i = \max\{0, d_i - C_i\}$

Flow shop – vide seção 1.3.

Flow time – tempo de fluxo ou tempo de permanência de uma tarefa ($F_i = C_i - r_i$).

Job shop – vide seção 1.3.

Lateness – diferença entre a data de término de uma tarefa e o prazo de término; se for positivo indica um atraso na entrega e se for negativo indica uma antecipação na entrega ($L_i = C_i - d_i$).

Makespan – duração total da programação; tempo máximo de fluxo (F_{max}) ou data de término máxima (C_{max}) das tarefas, quando as datas de liberação das tarefas forem iguais.

No-wait flow shop – *flow shop* onde não há tempo de espera entre as operações sucessivas das tarefas.

Open shop – vide seção 1.3.

Release date – data de liberação de uma tarefa a partir da qual ela pode ser executada.

Setup time – tempo de preparação de máquina.

Tail time – tempo gasto por tarefa em uma máquina após o seu processamento.

Tardiness – atraso na execução da tarefa, expresso por $T_i = \max(0, L_i)$.

Trade-off – literalmente, troca; quando o valor de uma variável é reduzido em função do aumento de uma outra.

APÊNDICE A

ESTUDOS EM PROGRAMAÇÃO DE OPERAÇÕES EM MÁQUINAS – REFERÊNCIAS

Desde o início da década de 1990, os estudos referentes à área de Programação de Operações em Máquinas na Escola de Engenharia de São Carlos (EESC) da USP resultaram em uma tese de livre docência, uma de doutorado e cinco dissertações de mestrado:

MOCCELLIN, J.V. (1992). **Contribuição à programação de operações em sistemas de produção intermitente flow-shop**. 126p. Tese (Livre Docência) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos. 1992.

NAGANO, M.S. (1995). **Novos procedimentos de busca tabu para o problema de programação de operações flow-shop permutacional**. 118p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos. 1995.

MOTA, W.L. (1996). **Análise comparativa de algoritmos genéticos para o problema de programação de operações flow shop permutacional**. 128p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos. 1996.

NAGANO, M.S. (1999). **Um novo método heurístico construtivo de alto desempenho para a programação de operações flow-shop permutacional**. 71p. Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos. 1999.

BUZZO, W.R. (1999). **Proposição de um método metaheurístico híbrido Algoritmo Genético-Simulated Annealing para o problema de programação de operações flow shop permutacional**. 96p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos. 1999.

SOUZA, A.B.D. (2000). **Desenvolvimento de um método metaheurístico híbrido Algoritmo Genético-Busca Tabu para o problema de programação de operações flow-shop permutacional**. 82p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos. 2000.

BARROS, A.D. (2002). **Algoritmo metaheurístico para busca do gargalo flutuante em flow shop permutacional com tempos de setup assimétricos e dependentes da seqüência**. 112p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos. 2002.

APÊNDICE B

SOLUÇÕES DOS PROBLEMAS DO EXPERIMENTO

A seguir serão apresentadas as tabelas com os parâmetros de cada classe de problemas das seis relações $O(p_i)/O(s_{ij})$ e os resultados dos quatro métodos para cada uma das medidas de comparação. Para simplificação da notação, o Procedimento 1 com a regra de prioridade LPT foi denominado "11" e com a regra TOTAL foi representado por "12"; e o Procedimento 2 com a regra de alocação SCT foi indicado como "21" e com a regra SCT/LPST como "22".

TABELA B.1 – Parâmetros das classes de problemas das relações I e II

relação I: O(pi) / O(sij) = 1					
Classe	***n***	***K***	***Mk***	***pi***	***sij***
1	10	4	2-5	1-99	1-99
2	20	4	2-5	1-99	1-99
3	30	4	2-5	1-99	1-99
4	40	4	2-5	1-99	1-99
5	50	4	2-5	1-99	1-99
6	60	4	2-5	1-99	1-99
7	70	4	2-5	1-99	1-99
8	80	4	2-5	1-99	1-99
9	90	4	2-5	1-99	1-99
10	100	4	2-5	1-99	1-99
11	110	4	2-5	1-99	1-99
12	120	4	2-5	1-99	1-99
13	10	7	2-5	1-99	1-99
14	20	7	2-5	1-99	1-99
15	30	7	2-5	1-99	1-99
16	40	7	2-5	1-99	1-99
17	50	7	2-5	1-99	1-99
18	60	7	2-5	1-99	1-99
19	70	7	2-5	1-99	1-99
20	80	7	2-5	1-99	1-99
21	90	7	2-5	1-99	1-99
22	100	7	2-5	1-99	1-99
23	110	7	2-5	1-99	1-99
24	120	7	2-5	1-99	1-99

relação II: O(pi) / O(sij) < 1					
Classe	***n***	***K***	***Mk***	***pi***	***sij***
25	10	4	2-5	1-99	100-120
26	20	4	2-5	1-99	100-120
27	30	4	2-5	1-99	100-120
28	40	4	2-5	1-99	100-120
29	50	4	2-5	1-99	100-120
30	60	4	2-5	1-99	100-120
31	70	4	2-5	1-99	100-120
32	80	4	2-5	1-99	100-120
33	90	4	2-5	1-99	100-120
34	100	4	2-5	1-99	100-120
35	110	4	2-5	1-99	100-120
36	120	4	2-5	1-99	100-120
37	10	7	2-5	1-99	100-120
38	20	7	2-5	1-99	100-120
39	30	7	2-5	1-99	100-120
40	40	7	2-5	1-99	100-120
41	50	7	2-5	1-99	100-120
42	60	7	2-5	1-99	100-120
43	70	7	2-5	1-99	100-120
44	80	7	2-5	1-99	100-120
45	90	7	2-5	1-99	100-120
46	100	7	2-5	1-99	100-120
47	110	7	2-5	1-99	100-120
48	120	7	2-5	1-99	100-120

TABELA B.2 – Parâmetros das classes de problemas das relações III e IV

relação III: O(pi) / O(sij) > 1 (i)					
Classe	***n***	***K***	***Mk***	***pi***	***sij***
49	10	4	2-5	10-99	1-9
50	20	4	2-5	10-99	1-9
51	30	4	2-5	10-99	1-9
52	40	4	2-5	10-99	1-9
53	50	4	2-5	10-99	1-9
54	60	4	2-5	10-99	1-9
55	70	4	2-5	10-99	1-9
56	80	4	2-5	10-99	1-9
57	90	4	2-5	10-99	1-9
58	100	4	2-5	10-99	1-9
59	110	4	2-5	10-99	1-9
60	120	4	2-5	10-99	1-9
61	10	7	2-5	10-99	1-9
62	20	7	2-5	10-99	1-9
63	30	7	2-5	10-99	1-9
64	40	7	2-5	10-99	1-9
65	50	7	2-5	10-99	1-9
66	60	7	2-5	10-99	1-9
67	70	7	2-5	10-99	1-9
68	80	7	2-5	10-99	1-9
69	90	7	2-5	10-99	1-9
70	100	7	2-5	10-99	1-9
71	110	7	2-5	10-99	1-9
72	120	7	2-5	10-99	1-9

relação IV: O(pi) / O(sij) > 1 (ii)					
Classe	***n***	***K***	***Mk***	***pi***	***sij***
73	10	4	2-5	50-99	1-49
74	20	4	2-5	50-99	1-49
75	30	4	2-5	50-99	1-49
76	40	4	2-5	50-99	1-49
77	50	4	2-5	50-99	1-49
78	60	4	2-5	50-99	1-49
79	70	4	2-5	50-99	1-49
80	80	4	2-5	50-99	1-49
81	90	4	2-5	50-99	1-49
82	100	4	2-5	50-99	1-49
83	110	4	2-5	50-99	1-49
84	120	4	2-5	50-99	1-49
85	10	7	2-5	50-99	1-49
86	20	7	2-5	50-99	1-49
87	30	7	2-5	50-99	1-49
88	40	7	2-5	50-99	1-49
89	50	7	2-5	50-99	1-49
90	60	7	2-5	50-99	1-49
91	70	7	2-5	50-99	1-49
92	80	7	2-5	50-99	1-49
93	90	7	2-5	50-99	1-49
94	100	7	2-5	50-99	1-49
95	110	7	2-5	50-99	1-49
96	120	7	2-5	50-99	1-49

TABELA B.3 – Parâmetros das classes de problemas das relações V e VI

relação V: O(pi) / O(sij) <= 1					
Classe	***n***	***K***	***Mk***	***pi***	***sij***
97	10	4	2-5	1-99	1-120
98	20	4	2-5	1-99	1-120
99	30	4	2-5	1-99	1-120
100	40	4	2-5	1-99	1-120
101	50	4	2-5	1-99	1-120
102	60	4	2-5	1-99	1-120
103	70	4	2-5	1-99	1-120
104	80	4	2-5	1-99	1-120
105	90	4	2-5	1-99	1-120
106	100	4	2-5	1-99	1-120
107	110	4	2-5	1-99	1-120
108	120	4	2-5	1-99	1-120
109	10	7	2-5	1-99	1-120
110	20	7	2-5	1-99	1-120
111	30	7	2-5	1-99	1-120
112	40	7	2-5	1-99	1-120
113	50	7	2-5	1-99	1-120
114	60	7	2-5	1-99	1-120
115	70	7	2-5	1-99	1-120
116	80	7	2-5	1-99	1-120
117	90	7	2-5	1-99	1-120
118	100	7	2-5	1-99	1-120
119	110	7	2-5	1-99	1-120
120	120	7	2-5	1-99	1-120

relação VI: O(pi) / O(sij) >= 1					
Classe	***n***	***K***	***Mk***	***pi***	***sij***
121	10	4	2-5	1-99	1-20
122	20	4	2-5	1-99	1-20
123	30	4	2-5	1-99	1-20
124	40	4	2-5	1-99	1-20
125	50	4	2-5	1-99	1-20
126	60	4	2-5	1-99	1-20
127	70	4	2-5	1-99	1-20
128	80	4	2-5	1-99	1-20
129	90	4	2-5	1-99	1-20
130	100	4	2-5	1-99	1-20
131	110	4	2-5	1-99	1-20
132	120	4	2-5	1-99	1-20
133	10	7	2-5	1-99	1-20
134	20	7	2-5	1-99	1-20
135	30	7	2-5	1-99	1-20
136	40	7	2-5	1-99	1-20
137	50	7	2-5	1-99	1-20
138	60	7	2-5	1-99	1-20
139	70	7	2-5	1-99	1-20
140	80	7	2-5	1-99	1-20
141	90	7	2-5	1-99	1-20
142	100	7	2-5	1-99	1-20
143	110	7	2-5	1-99	1-20
144	120	7	2-5	1-99	1-20

TABELA B.4 – Soluções dos problemas das classes 1 a 48

Classe	Porcent. de Sucesso				Desvio Relativo Médio (%)				Desvio-Padrão do DR				Tempo Médio (ms)			
	11	12	21	22	11	12	21	22	11	12	21	22	11	12	21	22
1	24	21	52	3	8,9	11,1	3,9	24,8	0,08	0,10	0,06	0,15	1,7	1,6	1,7	1,7
2	10	21	69	1	11,1	11,1	2,0	33,0	0,08	0,14	0,05	0,20	1,9	2,0	1,9	1,8
3	15	29	55	2	9,8	7,4	2,9	30,9	0,08	0,10	0,05	0,23	1,7	2,8	1,7	1,7
4	14	32	54	1	9,3	6,7	2,1	33,6	0,08	0,11	0,04	0,23	1,6	3,6	1,9	1,8
5	13	43	43	1	10,7	6,5	3,2	38,4	0,11	0,11	0,04	0,31	2,0	5,2	1,9	1,8
6	12	42	45	1	9,4	6,4	2,6	39,3	0,09	0,11	0,03	0,28	1,8	7,9	2,1	1,9
7	7	51	41	1	10,0	6,1	3,6	37,9	0,10	0,13	0,04	0,31	1,9	10,7	2,1	2,0
8	9	54	34	3	11,8	7,8	3,2	40,1	0,14	0,14	0,04	0,38	2,6	16,9	2,5	2,4
9	6	65	29	0	10,4	4,8	4,6	44,2	0,11	0,11	0,05	0,34	1,9	19,6	2,4	2,4
10	7	57	35	1	9,9	5,0	4,2	42,4	0,11	0,12	0,05	0,33	2,0	26,4	2,5	2,4
11	5	60	34	1	13,3	6,4	4,0	46,2	0,14	0,11	0,04	0,38	2,2	35,8	2,9	2,6
12	5	65	30	0	12,6	5,5	4,0	44,7	0,15	0,11	0,04	0,40	2,2	44,6	3,6	2,9
13	28	17	49	6	5,7	8,6	4,6	18,9	0,06	0,08	0,07	0,11	2,2	2,2	2,1	2,1
14	27	21	50	2	6,3	7,3	3,3	19,7	0,06	0,07	0,06	0,13	1,8	2,0	1,7	1,7
15	18	31	49	2	6,0	5,2	2,8	21,3	0,05	0,06	0,04	0,16	2,1	2,7	1,9	2,0
16	24	28	47	1	6,1	4,9	2,6	24,3	0,06	0,07	0,04	0,19	1,8	3,8	1,9	2,1
17	14	35	49	2	7,4	5,6	2,1	25,1	0,07	0,08	0,03	0,22	1,9	5,4	2,4	2,0
18	17	33	49	1	6,1	4,2	2,3	28,9	0,06	0,06	0,04	0,23	5,4	10,8	5,8	6,1
19	16	44	40	0	5,9	4,5	2,3	25,7	0,07	0,10	0,03	0,23	5,8	14,8	5,8	5,7
20	14	47	34	5	7,1	3,7	3,1	25,1	0,08	0,08	0,03	0,26	2,2	16,1	2,6	2,6
21	14	54	32	0	7,5	4,2	3,2	29,2	0,09	0,09	0,03	0,27	2,4	24,3	7,1	6,4
22	10	51	38	1	5,6	2,8	2,3	26,9	0,07	0,08	0,03	0,24	3,1	26,5	3,2	3,7
23	8	63	27	2	5,2	2,3	3,0	25,8	0,04	0,07	0,03	0,21	6,0	40,0	7,0	7,3
24	9	69	22	0	5,8	1,8	3,7	27,0	0,06	0,06	0,03	0,23	5,9	46,2	7,4	7,7
25	40	12	45	5	4,0	7,8	3,3	10,5	0,05	0,06	0,05	0,06	4,9	5,0	4,8	5,0
26	22	21	56	2	3,2	4,4	1,2	8,4	0,03	0,05	0,02	0,05	1,9	2,0	1,9	1,9
27	24	26	50	0	2,7	3,5	1,4	7,6	0,03	0,05	0,02	0,04	4,9	5,5	5,2	4,9
28	13	33	54	0	2,5	4,6	1,0	7,0	0,02	0,10	0,02	0,04	7,3	8,9	8,2	7,4
29	19	23	59	1	1,9	3,4	0,6	5,7	0,02	0,08	0,01	0,04	1,7	5,2	2,0	1,9
30	13	23	64	0	2,2	4,4	0,5	6,8	0,02	0,09	0,01	0,04	1,8	7,6	2,5	2,1
31	16	22	62	0	2,2	2,4	0,7	6,3	0,02	0,04	0,01	0,04	1,9	10,3	2,2	2,1
32	9	38	53	1	2,1	3,0	0,8	5,7	0,01	0,08	0,01	0,04	2,0	16,0	2,4	2,1
33	7	43	49	1	1,9	3,0	0,7	5,9	0,01	0,07	0,01	0,04	2,0	19,9	2,5	2,4
34	7	38	54	2	1,8	3,5	0,6	6,3	0,01	0,09	0,01	0,04	2,0	26,0	2,7	2,4
35	4	40	56	0	1,9	2,7	0,6	6,0	0,01	0,06	0,01	0,04	2,3	37,1	3,1	2,7
36	2	45	53	1	1,9	2,3	0,6	5,9	0,01	0,06	0,01	0,04	2,2	43,5	3,2	2,8
37	39	17	37	9	3,7	5,8	3,4	9,2	0,05	0,05	0,04	0,06	1,7	1,8	1,7	1,8
38	31	14	54	1	2,6	4,6	1,4	7,8	0,03	0,04	0,02	0,04	1,9	2,0	1,8	1,8
39	27	14	56	4	2,8	3,7	1,1	6,7	0,03	0,05	0,02	0,04	1,7	2,6	1,9	2,1
40	21	25	52	3	2,5	2,2	0,9	6,2	0,02	0,02	0,01	0,04	4,8	6,1	4,9	4,9
41	19	27	54	0	2,1	2,1	0,8	5,7	0,02	0,02	0,01	0,04	6,4	8,7	6,1	6,5
42	20	28	49	4	2,0	1,8	0,9	5,4	0,02	0,02	0,01	0,04	6,0	11,3	6,1	6,1
43	20	29	53	0	1,6	2,7	0,8	5,9	0,01	0,06	0,01	0,04	5,8	14,1	5,9	6,1
44	13	37	51	0	1,8	1,8	0,7	4,7	0,02	0,04	0,01	0,04	5,9	17,9	6,6	6,6
45	12	37	50	3	1,6	2,4	0,7	5,2	0,01	0,06	0,01	0,04	6,4	22,7	6,7	6,3
46	18	35	46	1	1,5	1,4	0,7	5,1	0,01	0,02	0,01	0,04	2,2	26,5	3,2	2,9
47	17	41	40	2	1,4	1,3	0,7	4,5	0,01	0,03	0,01	0,04	2,2	35,1	3,4	3,4
48	19	32	49	0	1,3	1,7	0,7	4,8	0,01	0,04	0,01	0,04	6,0	46,2	7,8	7,5

TABELA B.5 – Soluções dos problemas das classes 49 a 96

Classe	Porcent. de Sucesso				Desvio Relativo Médio (%)				Desvio-Padrão do DR				Tempo Médio (ms)			
	11	12	21	22	11	12	21	22	11	12	21	22	11	12	21	22
49	40	10	52	4	4,7	10,3	3,5	14,4	0,06	0,09	0,07	0,10	4,5	5,0	5,0	5,0
50	37	16	47	1	4,7	8,1	2,6	14,3	0,06	0,07	0,04	0,07	4,7	5,0	5,0	5,1
51	28	13	58	1	4,0	7,9	2,2	12,5	0,04	0,08	0,04	0,06	8,6	8,5	9,0	8,9
52	21	9	70	0	4,5	6,9	1,0	13,6	0,04	0,06	0,02	0,07	4,9	9,5	5,1	5,1
53	27	9	64	0	3,6	5,7	1,0	12,5	0,03	0,06	0,02	0,07	4,9	10,1	5,0	5,0
54	29	17	56	0	3,0	5,0	1,0	13,1	0,03	0,06	0,02	0,07	5,1	10,2	5,0	5,0
55	21	7	72	0	3,1	6,5	0,5	12,1	0,03	0,10	0,01	0,07	5,1	14,8	5,4	5,1
56	24	12	64	0	2,4	5,5	0,5	11,7	0,02	0,10	0,01	0,07	6,2	18,3	7,4	7,1
57	21	10	70	0	2,4	5,4	0,5	12,0	0,02	0,08	0,01	0,08	5,6	21,2	5,7	5,4
58	21	11	68	0	2,0	7,3	0,4	12,1	0,02	0,13	0,01	0,08	5,8	27,9	6,3	6,1
59	20	11	69	0	2,2	4,7	0,4	12,4	0,02	0,08	0,01	0,09	6,1	37,0	7,0	6,8
60	23	10	70	1	1,8	6,0	0,4	11,1	0,02	0,13	0,01	0,08	5,8	45,0	7,0	6,8
61	39	9	48	7	4,6	8,3	3,1	12,2	0,06	0,06	0,04	0,08	4,4	5,3	4,9	4,6
62	28	16	55	2	5,1	7,0	1,8	11,5	0,05	0,07	0,03	0,06	4,7	6,2	5,3	5,3
63	31	7	61	1	4,0	6,6	1,8	12,4	0,05	0,06	0,03	0,06	5,0	6,5	5,0	5,0
64	23	12	65	0	3,8	5,6	1,2	11,2	0,03	0,05	0,02	0,06	5,0	9,8	5,4	5,2
65	33	12	55	1	3,4	6,2	1,5	11,8	0,04	0,06	0,03	0,07	6,3	7,1	5,7	6,1
66	28	17	53	2	3,1	4,3	1,3	10,4	0,03	0,04	0,02	0,07	6,0	12,1	7,0	6,9
67	26	14	58	2	3,1	5,1	0,8	10,3	0,03	0,06	0,01	0,07	6,4	14,5	6,6	6,4
68	27	4	68	1	2,6	4,8	0,8	11,2	0,03	0,04	0,02	0,07	6,1	17,9	6,2	6,1
69	24	9	66	3	2,4	4,3	0,7	10,0	0,02	0,06	0,01	0,07	5,8	22,7	6,9	6,9
70	29	12	59	0	2,5	4,2	0,7	11,2	0,03	0,05	0,01	0,07	6,1	29,2	7,1	6,9
71	22	12	65	2	2,4	3,2	0,7	9,7	0,02	0,03	0,01	0,08	6,4	37,0	7,3	6,8
72	28	13	58	1	2,1	4,6	0,7	8,6	0,02	0,07	0,01	0,07	5,7	45,5	7,3	6,5
73	27	15	55	3	4,4	5,3	2,1	11,3	0,04	0,04	0,03	0,06	4,1	5,0	5,0	5,0
74	12	25	63	1	4,4	5,7	1,3	12,6	0,03	0,08	0,02	0,08	4,8	5,0	5,1	5,0
75	9	27	62	3	4,4	5,2	1,3	13,1	0,04	0,06	0,02	0,10	5,4	5,4	5,9	6,0
76	10	32	52	6	4,1	3,4	1,3	11,5	0,04	0,05	0,02	0,10	5,1	9,7	5,5	5,5
77	14	29	58	1	4,9	5,1	0,9	16,1	0,05	0,08	0,02	0,12	5,0	9,8	5,0	5,1
78	11	48	39	2	4,3	3,6	1,6	13,4	0,05	0,07	0,02	0,12	5,5	11,2	5,6	5,5
79	11	48	40	3	3,6	3,1	1,7	11,0	0,03	0,07	0,02	0,11	5,7	13,9	6,1	6,1
80	10	45	43	2	4,4	4,2	1,2	14,1	0,05	0,08	0,02	0,13	5,2	18,5	5,7	5,6
81	9	54	37	0	5,1	4,8	1,8	14,9	0,06	0,11	0,02	0,15	5,4	22,0	5,8	5,5
82	3	59	37	1	5,2	3,8	1,7	15,5	0,05	0,08	0,02	0,14	5,8	28,9	6,2	6,1
83	4	56	40	1	5,4	5,2	1,6	16,9	0,06	0,09	0,02	0,16	5,7	37,4	7,0	6,7
84	8	50	42	0	6,0	5,3	1,7	17,8	0,07	0,09	0,02	0,16	5,9	53,6	7,3	6,3
85	28	15	53	5	3,8	4,6	1,5	7,1	0,04	0,04	0,03	0,05	4,6	4,6	4,9	4,9
86	26	19	53	4	3,4	3,9	1,3	8,8	0,03	0,03	0,02	0,05	6,3	9,5	7,5	6,8
87	20	28	51	3	2,9	3,6	1,6	8,5	0,03	0,06	0,02	0,07	5,1	8,5	5,8	5,4
88	13	31	54	2	3,5	3,1	1,0	10,8	0,03	0,04	0,02	0,09	6,6	7,3	6,4	6,8
89	17	35	44	4	2,7	2,5	1,3	9,6	0,03	0,04	0,02	0,09	5,1	9,1	5,3	5,2
90	16	47	32	6	2,7	1,8	1,8	7,0	0,03	0,04	0,02	0,08	5,3	10,9	5,5	5,5
91	18	41	39	3	2,7	2,1	1,3	9,8	0,03	0,04	0,02	0,09	5,6	15,0	6,3	7,0
92	17	46	36	2	2,9	1,6	1,3	10,1	0,03	0,03	0,01	0,10	5,3	17,7	6,1	6,6
93	12	53	33	3	2,9	2,4	1,5	10,1	0,03	0,06	0,02	0,10	5,0	23,0	7,4	6,8
94	8	61	27	4	2,8	1,3	1,7	9,8	0,02	0,04	0,02	0,09	10,7	28,5	7,2	6,5
95	8	53	39	1	3,1	1,7	1,5	11,9	0,04	0,05	0,02	0,12	6,0	37,4	7,4	8,3
96	15	49	34	2	2,9	1,8	1,3	10,6	0,03	0,05	0,02	0,11	6,2	49,4	9,1	7,8

TABELA B.6 – Soluções dos problemas das classes 97 a 144

Classe	Porcent. de Sucesso				Desvio Relativo Médio (%)				Desvio-Padrão do DR				Tempo Médio (ms)			
	11	12	21	22	11	12	21	22	11	12	21	22	11	12	21	22
97	24	21	57	0	8,6	10,4	3,9	28,8	0,08	0,10	0,07	0,17	5,0	5,1	5,0	5,1
98	22	21	55	2	12,9	12,3	4,0	39,0	0,11	0,14	0,07	0,27	4,8	5,0	5,0	5,1
99	17	32	47	5	9,3	5,9	3,6	36,9	0,09	0,08	0,05	0,27	5,5	5,5	5,7	5,7
100	13	41	46	0	11,0	7,6	3,5	42,1	0,10	0,13	0,04	0,32	5,6	9,9	5,8	5,7
101	10	58	32	0	11,3	5,6	4,0	41,7	0,11	0,12	0,05	0,34	5,2	9,3	5,3	5,5
102	9	51	40	0	10,8	4,9	3,5	44,1	0,11	0,11	0,04	0,34	5,6	11,5	6,2	5,5
103	9	54	36	1	10,8	7,1	4,0	43,2	0,13	0,13	0,05	0,37	6,1	13,7	6,2	6,5
104	4	65	29	2	12,8	4,6	4,7	46,4	0,14	0,10	0,05	0,38	5,2	18,2	6,0	5,5
105	5	71	24	0	13,6	4,9	5,4	51,7	0,14	0,11	0,05	0,41	5,7	22,7	6,3	6,0
106	7	60	32	1	13,0	5,5	5,4	47,4	0,14	0,12	0,05	0,43	5,2	29,4	5,6	6,7
107	4	70	25	1	12,5	4,9	5,3	57,0	0,14	0,12	0,05	0,38	5,8	38,7	7,3	6,7
108	8	68	23	1	15,8	6,2	5,5	54,5	0,18	0,13	0,05	0,49	6,0	50,3	6,9	6,6
109	27	23	44	6	6,6	7,7	3,8	17,6	0,07	0,07	0,05	0,11	4,7	4,9	5,0	5,0
110	21	29	49	1	6,5	5,7	3,9	20,7	0,06	0,07	0,06	0,13	6,9	9,6	8,5	7,8
111	18	37	44	2	7,8	5,4	2,9	24,5	0,07	0,08	0,04	0,19	6,1	9,2	7,0	6,3
112	24	38	39	0	6,4	4,4	3,5	24,2	0,07	0,07	0,05	0,21	6,1	7,7	6,9	7,1
113	14	48	37	1	6,8	3,5	3,5	27,3	0,07	0,06	0,04	0,23	5,5	9,7	5,8	5,3
114	13	45	43	0	6,5	3,1	2,8	29,0	0,06	0,05	0,04	0,22	5,3	16,6	5,3	5,6
115	8	49	41	3	7,7	4,6	2,9	33,8	0,09	0,10	0,03	0,30	5,8	14,3	6,0	6,5
116	11	58	30	1	6,7	2,9	4,2	28,9	0,08	0,07	0,05	0,25	6,2	17,6	6,6	6,2
117	10	61	27	2	6,7	2,4	3,6	28,0	0,07	0,06	0,04	0,26	6,1	25,0	3,7	6,0
118	9	69	19	4	6,7	2,6	4,1	28,9	0,07	0,09	0,04	0,28	5,5	30,5	7,6	6,9
119	11	65	23	1	7,2	2,5	3,3	30,4	0,09	0,08	0,03	0,30	5,7	40,5	8,8	6,3
120	8	65	25	2	7,3	2,8	3,7	32,0	0,10	0,09	0,04	0,33	6,5	48,3	7,5	9,4
121	49	12	36	3	5,3	11,7	5,7	16,4	0,07	0,09	0,08	0,09	1,8	1,8	1,8	1,9
122	29	16	55	0	5,6	8,8	2,4	18,1	0,06	0,08	0,04	0,09	1,4	2,1	2,0	1,7
123	26	19	55	0	5,2	6,9	1,6	18,4	0,05	0,06	0,03	0,09	1,7	2,5	1,7	1,7
124	17	12	70	1	5,8	7,5	1,2	17,5	0,04	0,06	0,02	0,10	5,1	9,7	5,1	5,2
125	19	8	72	1	4,7	7,8	0,7	18,5	0,04	0,09	0,02	0,10	4,9	9,4	5,0	5,1
126	18	10	72	0	3,9	6,1	0,6	17,6	0,03	0,06	0,01	0,09	5,4	11,2	6,0	5,8
127	21	18	62	0	4,2	6,0	1,1	16,0	0,04	0,10	0,02	0,09	5,1	10,8	5,2	4,9
128	12	21	65	2	3,8	4,3	0,8	15,0	0,03	0,06	0,02	0,10	5,5	17,5	6,2	5,2
129	11	20	67	2	3,8	5,7	0,7	15,1	0,03	0,09	0,01	0,11	2,3	16,4	2,4	2,3
130	11	12	77	0	4,1	5,4	0,4	16,1	0,03	0,08	0,01	0,10	2,1	25,5	2,8	2,5
131	9	21	70	0	4,0	7,0	0,5	17,5	0,03	0,13	0,01	0,11	2,3	36,5	3,6	2,8
132	10	19	72	0	3,8	5,3	0,4	17,8	0,03	0,08	0,01	0,11	2,7	51,3	4,3	3,3
133	35	12	52	1	5,4	9,6	3,2	14,8	0,06	0,07	0,05	0,09	2,0	2,0	2,0	2,0
134	33	10	55	3	5,7	9,0	2,6	16,6	0,06	0,07	0,04	0,08	5,3	8,6	6,5	6,0
135	27	15	58	1	4,9	6,9	1,5	13,3	0,05	0,07	0,03	0,07	5,0	7,3	5,2	5,3
136	22	10	66	2	5,2	6,5	1,2	14,3	0,05	0,05	0,02	0,08	1,8	3,5	1,9	1,8
137	26	16	58	1	3,9	5,2	1,4	12,5	0,04	0,06	0,02	0,07	5,2	9,4	5,1	5,4
138	26	14	60	0	4,1	5,4	1,2	13,5	0,04	0,06	0,02	0,09	5,5	11,4	5,9	5,7
139	26	12	60	2	3,4	5,2	1,2	12,8	0,03	0,05	0,02	0,09	4,0	12,7	5,1	4,1
140	27	9	64	0	3,3	5,6	1,1	15,4	0,03	0,04	0,02	0,09	6,3	17,7	7,1	6,6
141	23	13	62	2	3,2	4,0	1,1	12,1	0,03	0,04	0,02	0,09	6,0	23,5	6,0	6,6
142	24	21	54	2	2,7	4,3	1,1	12,9	0,03	0,06	0,02	0,09	8,1	36,9	7,4	5,5
143	21	21	56	2	3,0	3,9	1,2	12,6	0,03	0,06	0,02	0,10	6,2	37,6	7,3	6,8
144	19	20	58	3	2,9	3,2	0,8	12,9	0,02	0,03	0,01	0,10	6,2	45,9	8,1	7,0

APÊNDICE C

GRÁFICOS GERAIS DA EXPERIMENTAÇÃO COMPUTACIONAL

As figuras C.1 a C.8 apresentam os gráficos gerais dos resultados da experimentação computacional discorrida no capítulo 4, para cada medida de comparação e opção de número de estágios.

Os gráficos das figuras C.1 e C2 refere-se aos resultados da porcentagem de sucesso dos problemas com 4 e 7 estágios, respectivamente. Nas figuras C.3 e C.4, os gráficos mostram o desvio relativo (em porcentagem) dos resultados. Os gráficos das figuras C.5 e C.6 demonstram o desvio-padrão do desvio relativo. E o tempo médio de computação (em milissegundos) é mostrado nos gráficos das figuras C.7 e C.8.

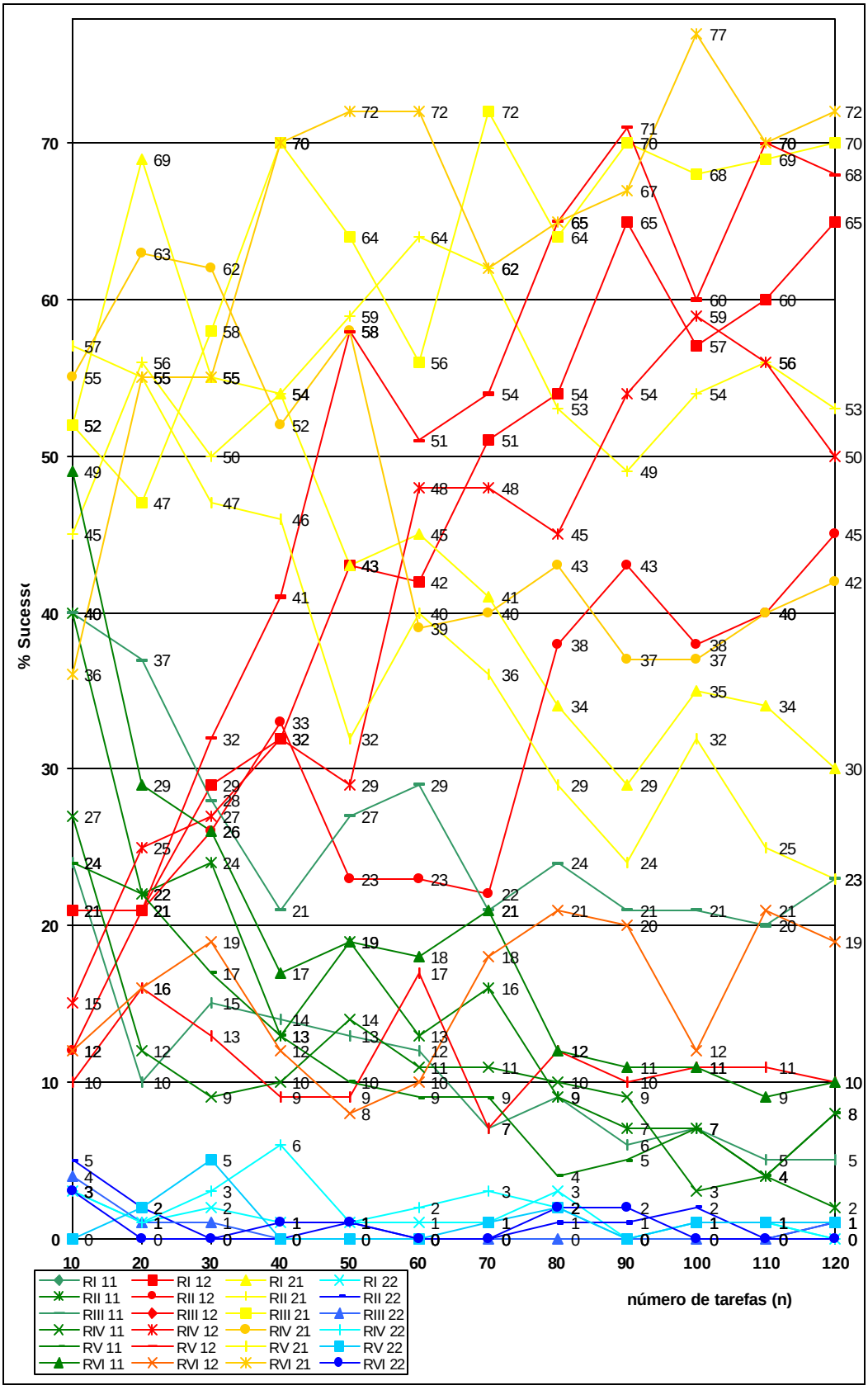


FIGURA C.1 – Comparação da porcentagem de sucesso para 4 estágios

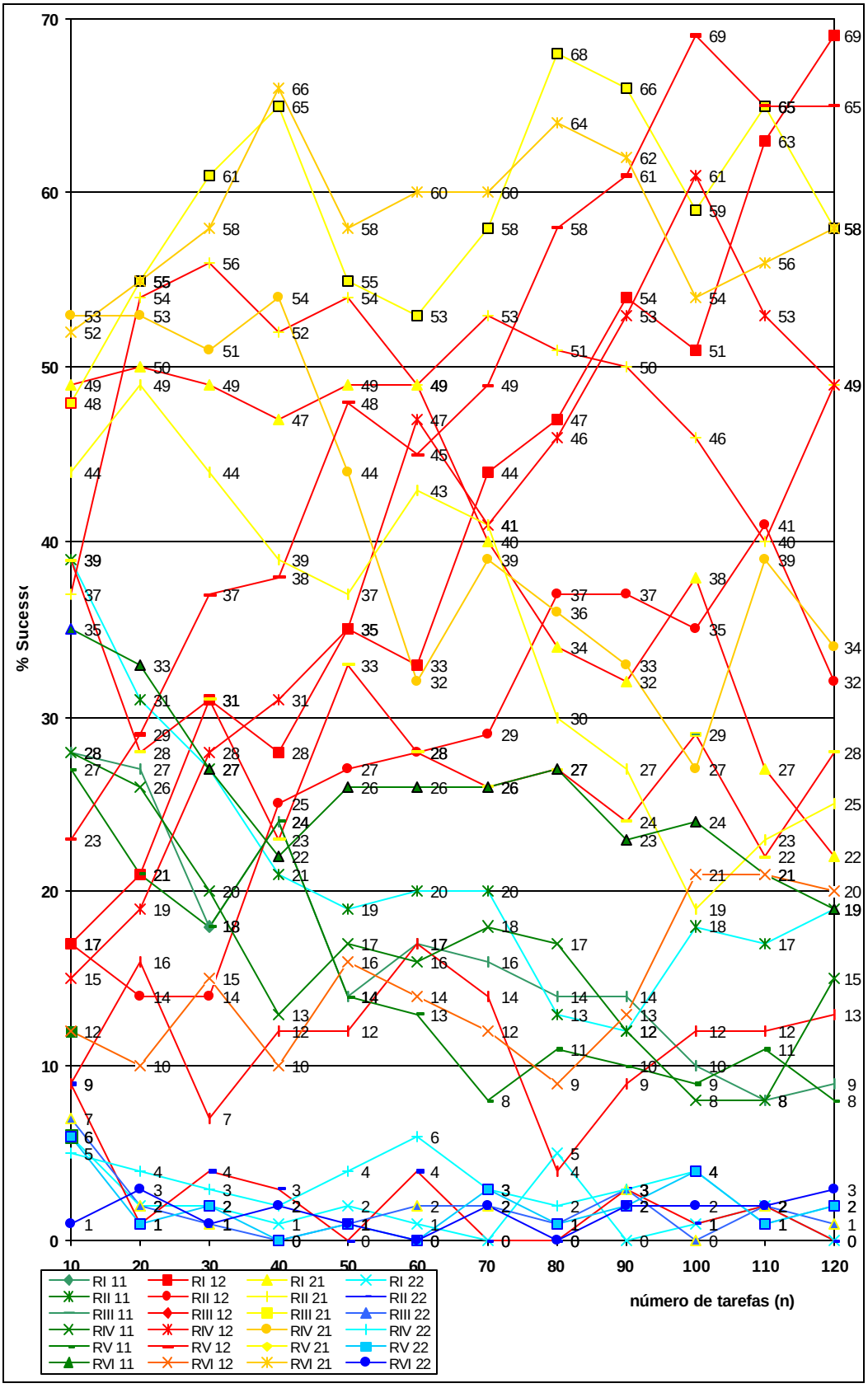


FIGURA C.2 – Comparação da porcentagem de sucesso para 7 estágios

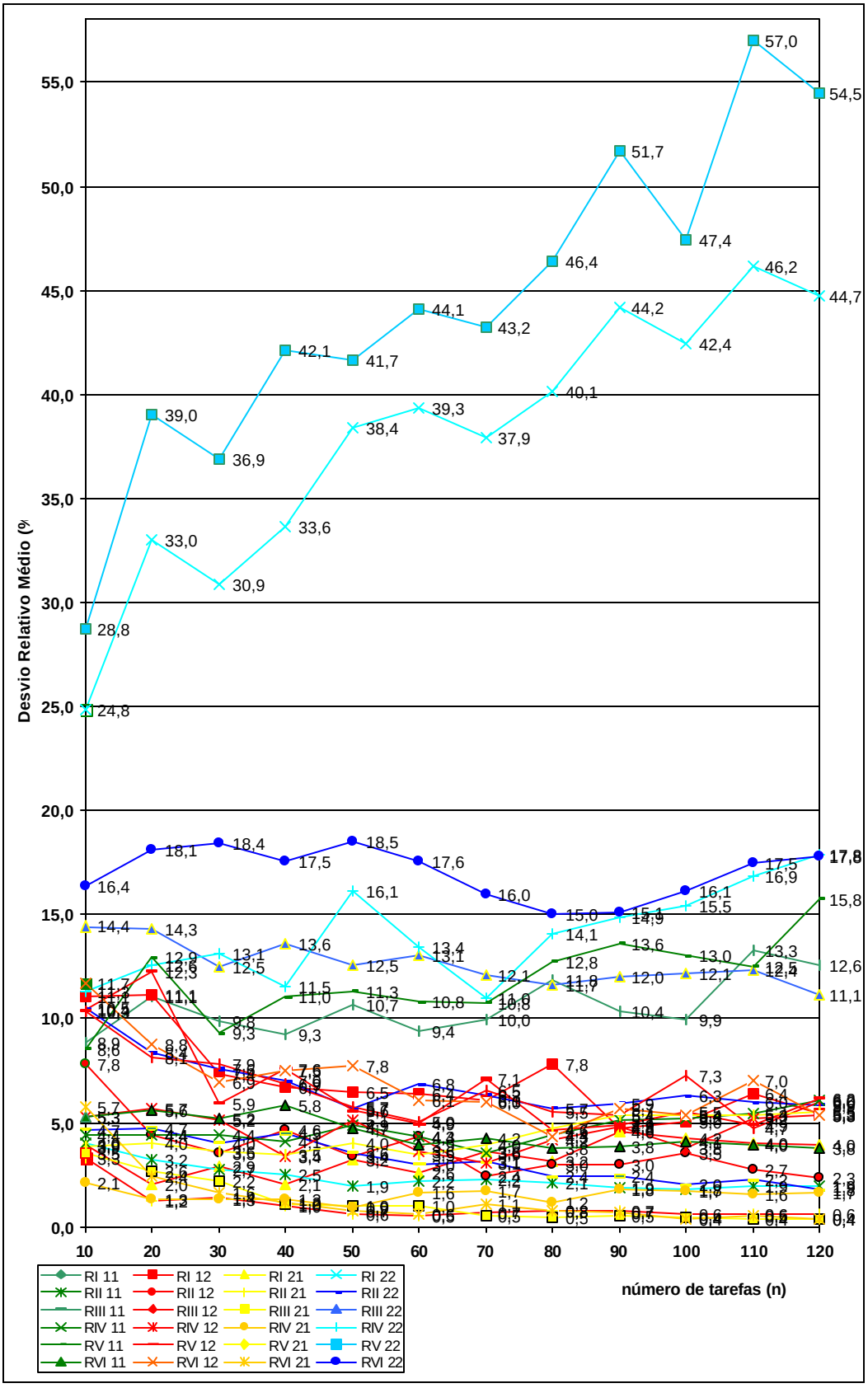


FIGURA C.3 – Comparação do desvio relativo (%) para 4 estágios

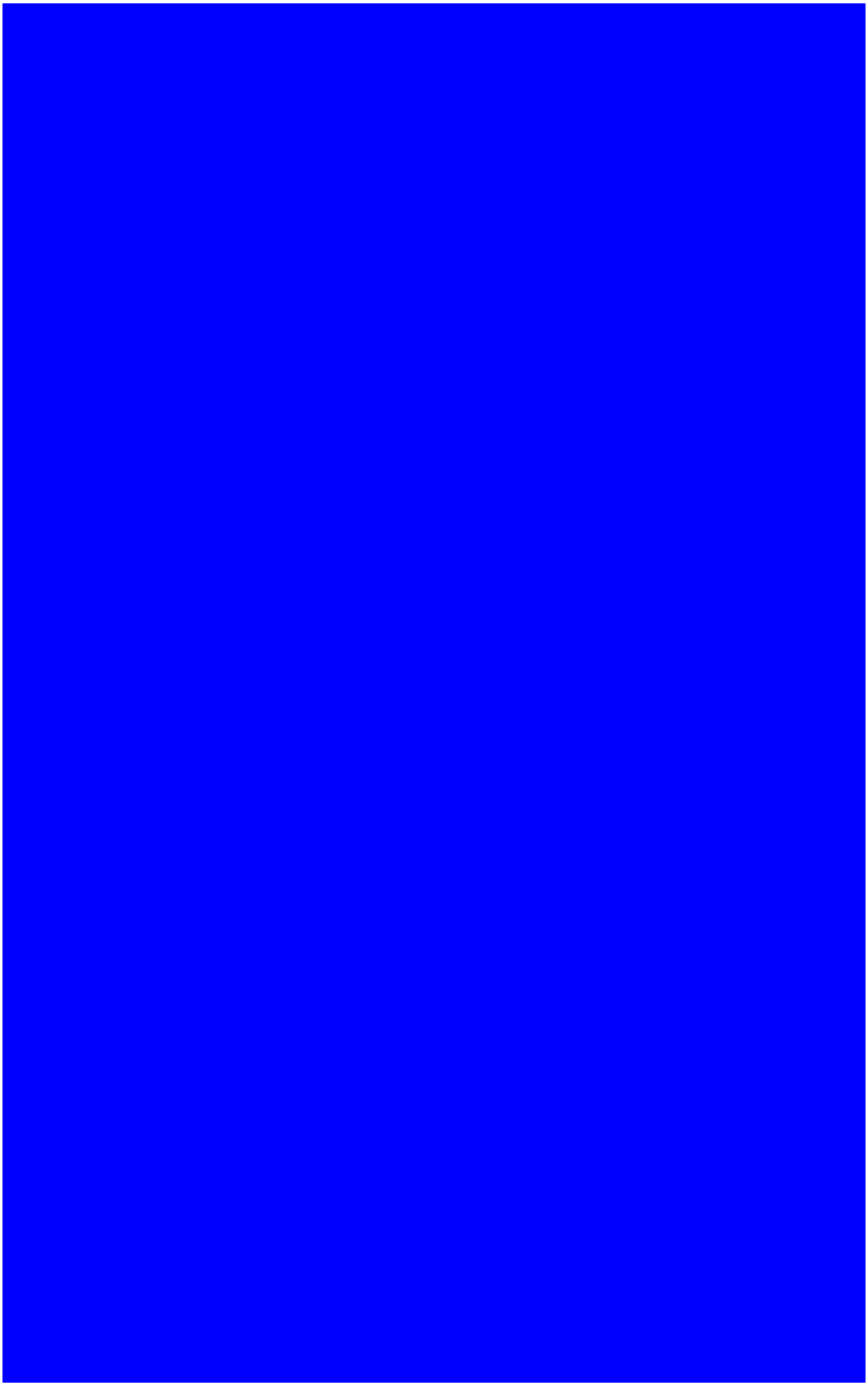


FIGURA C.4 – Comparação do desvio relativo (%) para 7 estágios

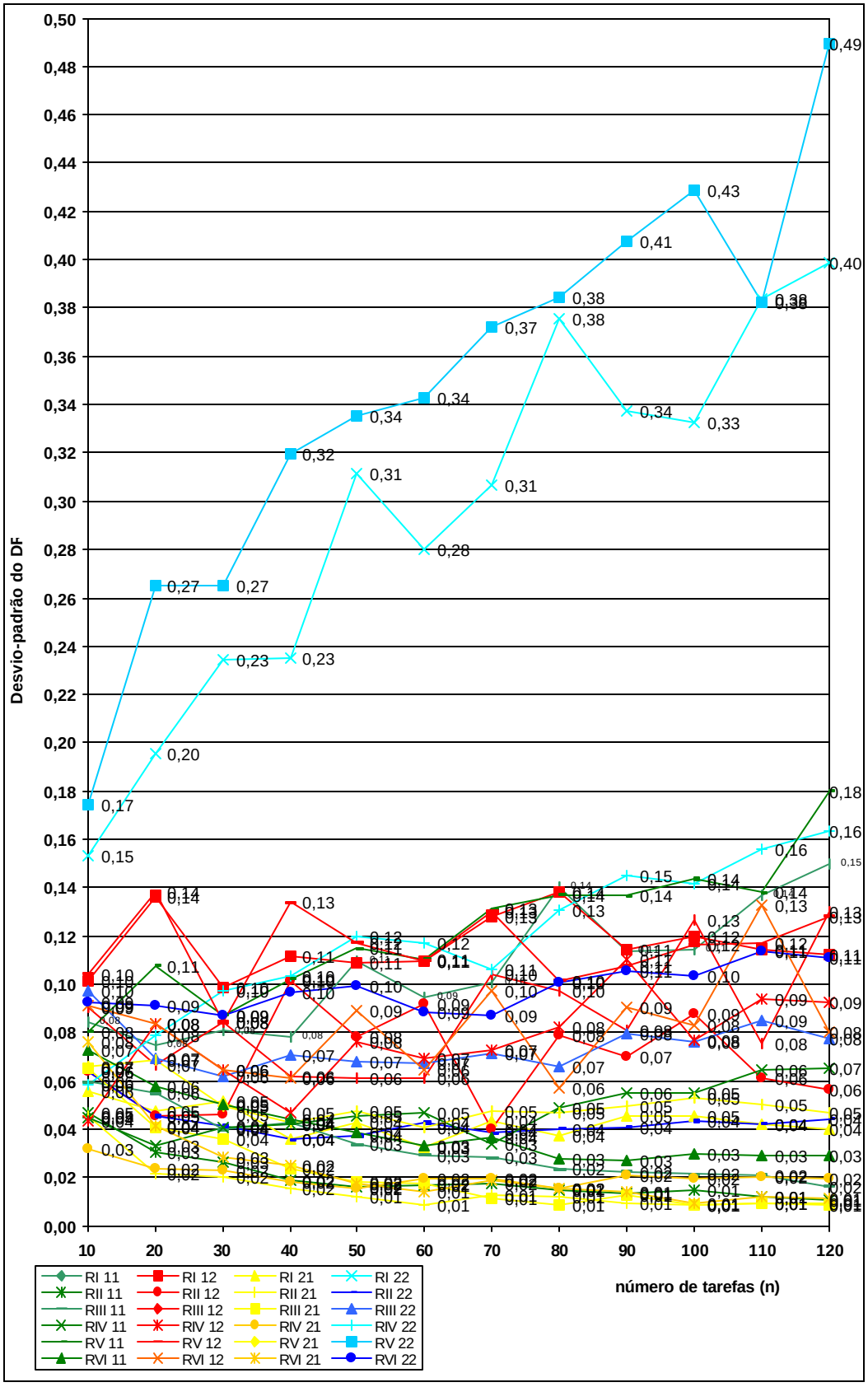


FIGURA C.5 – Comparação do desvio-padrão do desvio relativo para 4 estágios

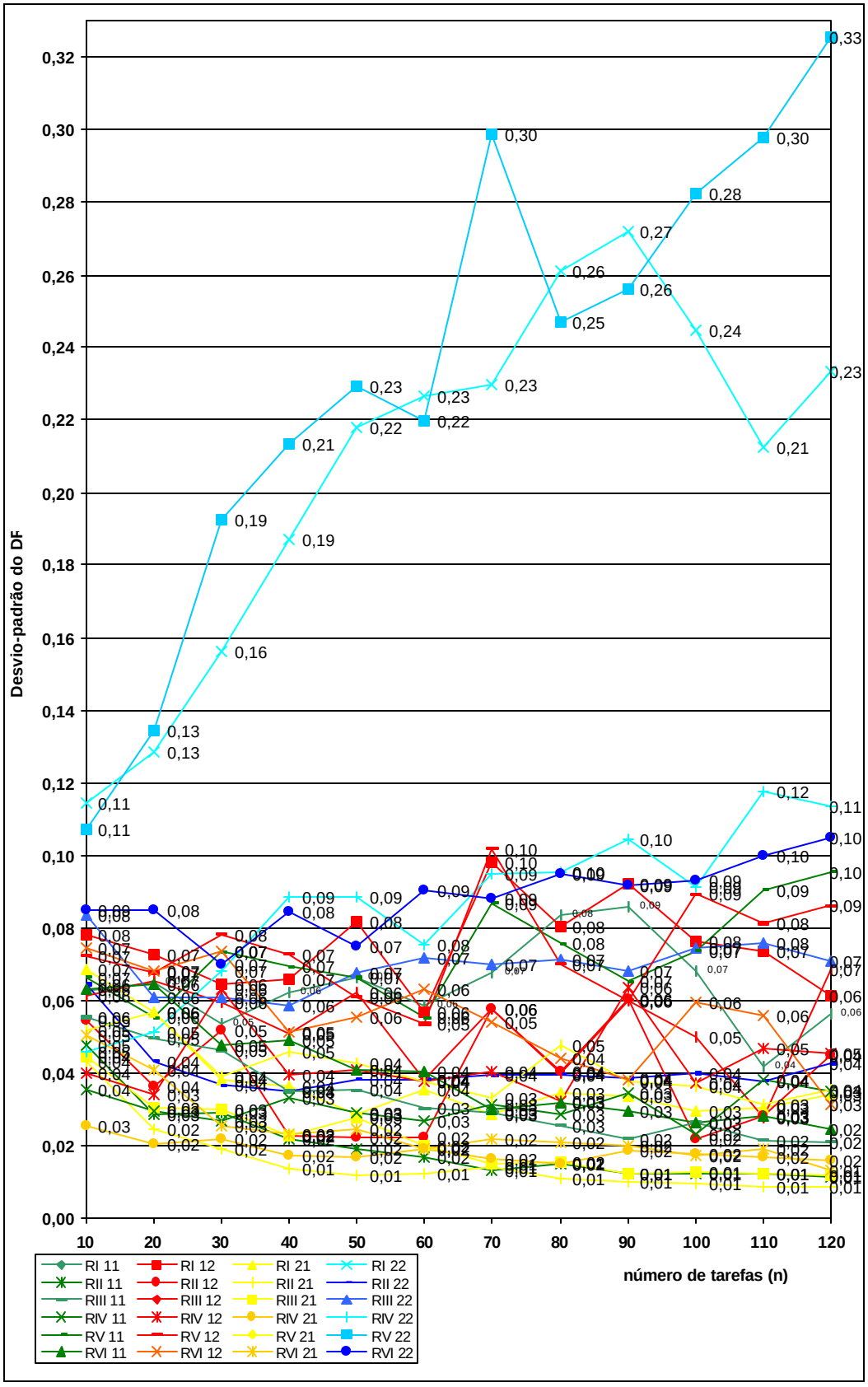


FIGURA C.6 – Comparação do desvio-padrão do desvio relativo para 7 estágios

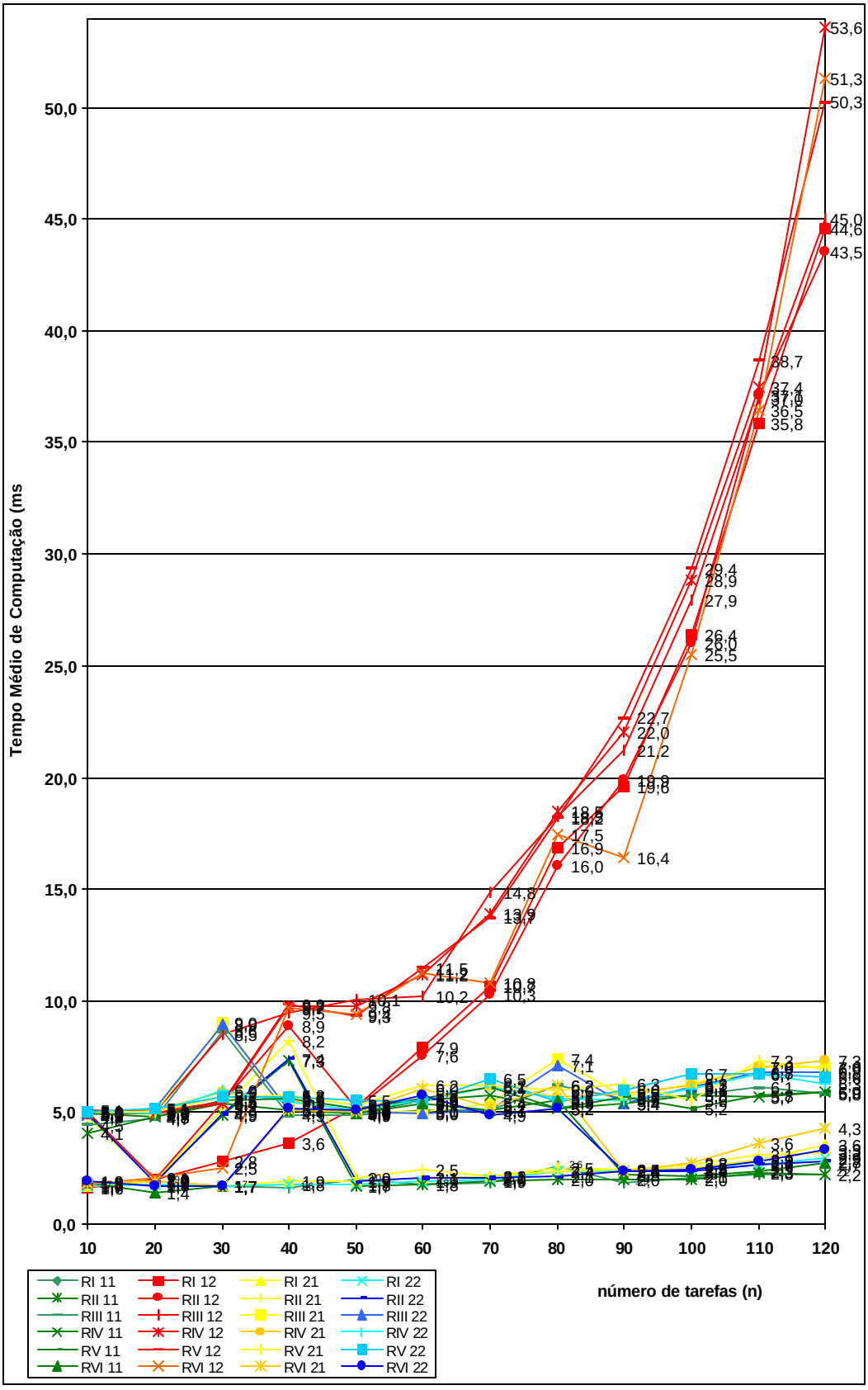


FIGURA C.7 – Comparação do tempo médio de computação (ms) para 4 estágios

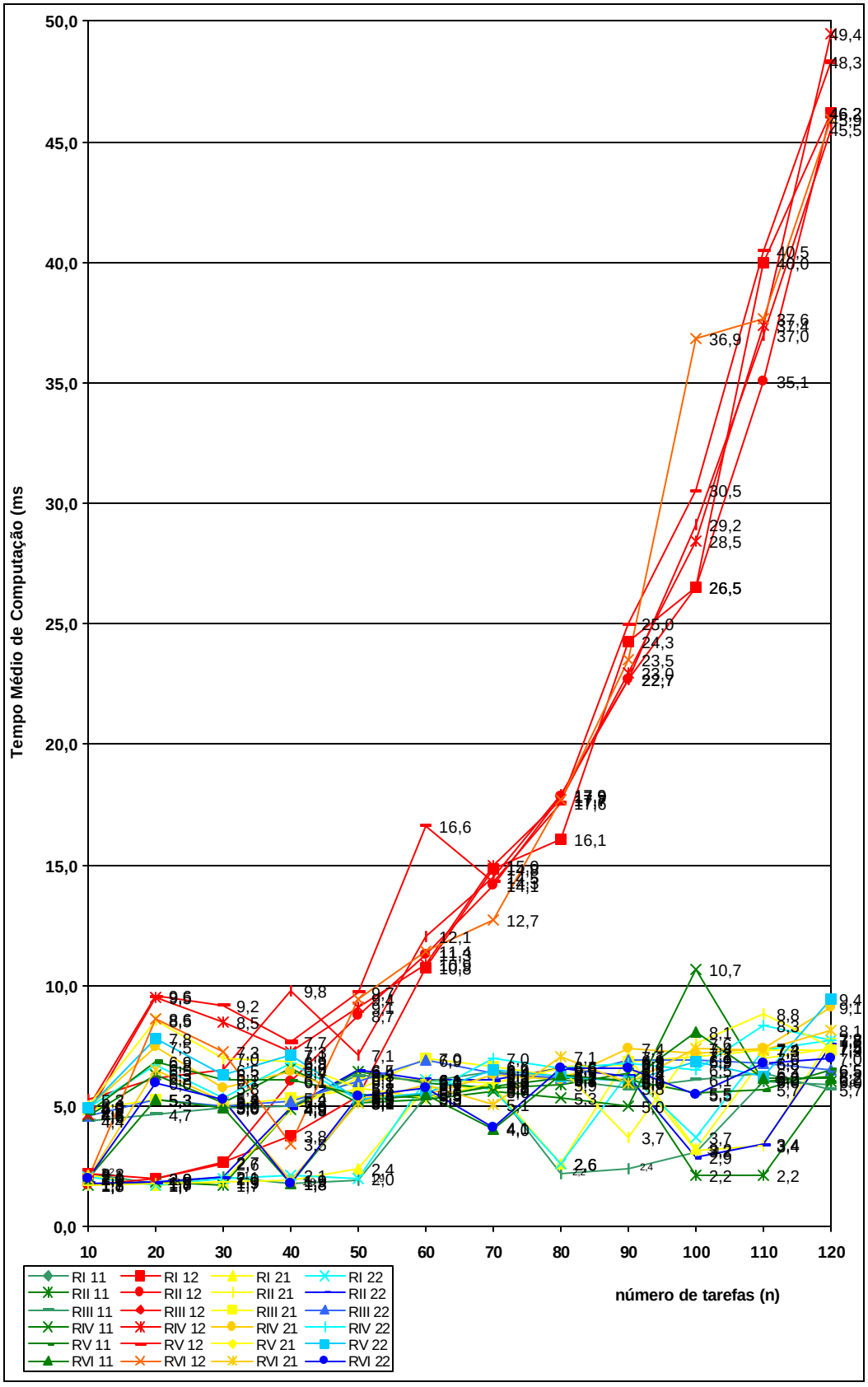


FIGURA C.8 – Comparação do tempo médio de computação (ms) para 7 estágios

APÊNDICE D

FORMATO DOS ARQUIVOS DE DADOS E DE SAÍDA

Os formatos dos arquivos de dados gerados para cada problema e dos arquivos de saída com os resultados serão apresentados a seguir. A extensão de todos os arquivos é *.txt*.

A figura D.1 ilustra um arquivo com os dados de um problema. As informações são separadas por espaço e por linhas. A primeira linha contém o número de tarefas, de máquinas e de estágios do problema. Na segunda linha, há o número de máquinas em cada estágio.

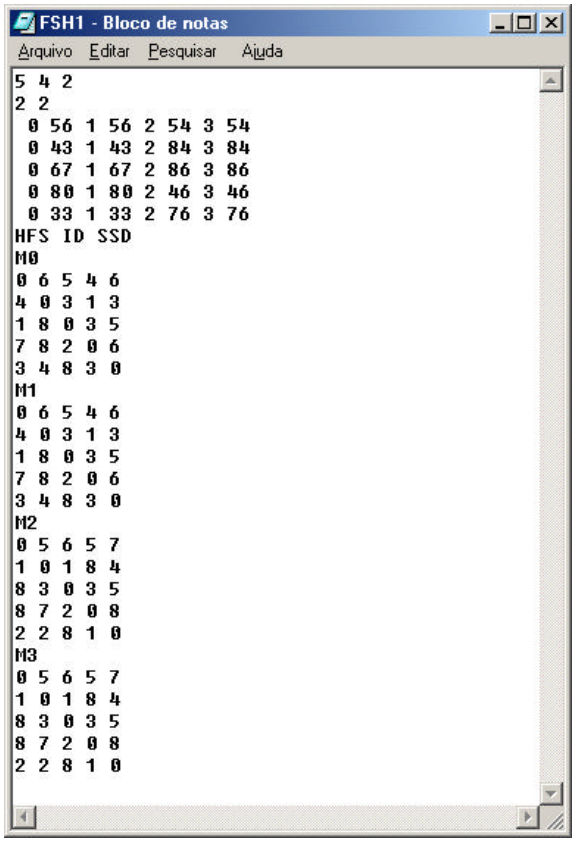


FIGURA D.1 – Formato do arquivo de dados dos problemas

Em seguida, o arquivo contém a matriz de tempos de processamento de cada tarefa em cada máquina. Os tempos de processamento nas máquinas do mesmo estágio serão iguais neste trabalho por se tratar de máquinas paralelas idênticas. O formato desta matriz é o seguinte: colunas com o índice da máquina (iniciando em zero) intercalam com os tempos de processamento das tarefas. Portanto, o número de linhas é igual ao número de tarefas.

A próxima linha contém o tipo de problema: "HFS" significa *Hybrid Flow Shop*, "ID" refere-se a máquinas paralelas idênticas e "SSD" denota o *setup* dependente da seqüência.

Por fim, são apresentadas as matrizes de tempos de *setup* da tarefa $J_{[linha]}$ para a $J_{[coluna]}$ para cada uma das máquinas. As matrizes referentes às máquinas do mesmo estágio serão idênticas.

Um arquivo de saída é ilustrado na figura D.2. Cada linha contém o número do problema, o *makespan* e a programação das máquinas, representadas pela letra "M" seguida do índice da máquina.

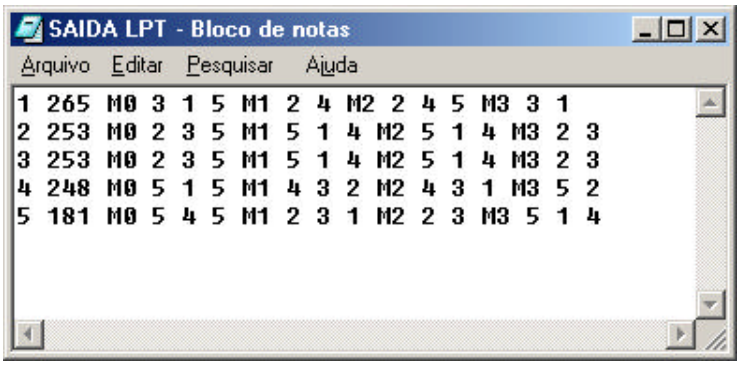


FIGURA D.2 – Arquivo de saída com a programação de cinco problemas pelo Procedimento 1 e Regra de Prioridade LPT

As figuras D.3 e D.4 mostram arquivos de comparação do *makespan* e do tempo de computação, respectivamente, para os quatro métodos de solução. A primeira coluna contém o número do problema, e as colunas seguintes as saídas referentes aos métodos LPT, TOTAL, SCT e SCT/LPST.

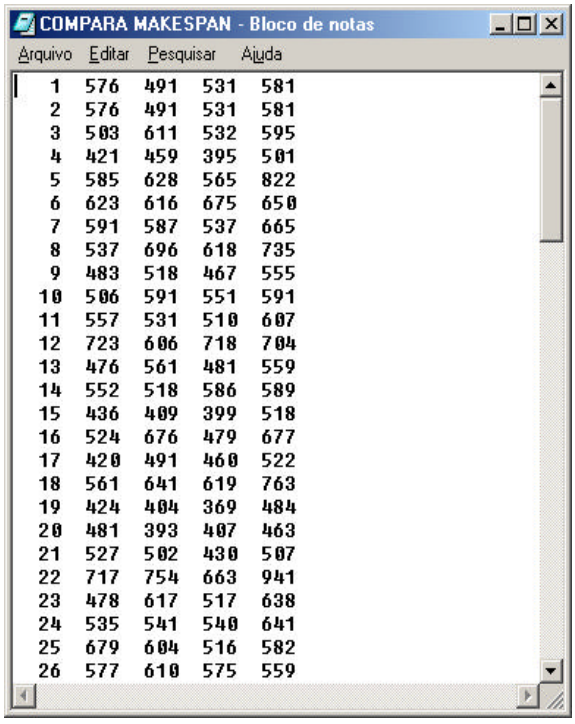


FIGURA D.3 – Arquivo de saída com a comparação do *makespan* dos quatro métodos

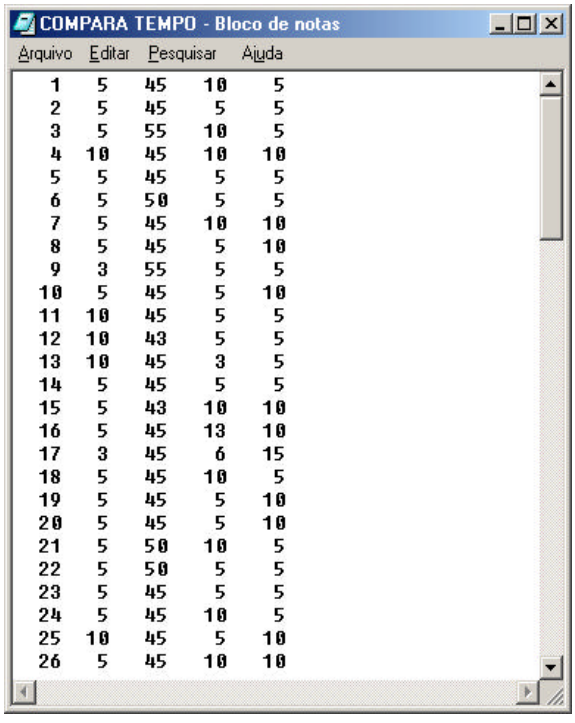


FIGURA D.4 – Arquivo de saída com a comparação do tempo de computação dos quatro métodos

APÊNDICE E

CÓDIGO-FONTE DOS PROGRAMAS COMPUTACIONAIS

```
unit Flow_Shop_Hibrido;

type
  vetordinamico  = array of integer;
  matrizdinamica2 = array of array of integer;
  matrizdinamica3 = array of array of array of integer;

const
  infinito = 32767; //máximo valor para número inteiro
var
  numtarefas, nummaquinas, numestagios, qtdproblemas, numeracao,
  hinicial, hfinal, //para cálculo do tempo de execução
  mk, //número de máquinas de cada estágio
  somap, //soma dos tempos de processamento de cada tarefa
  ordenacao, //seqüência ordenada de tarefas
  ultimatarefa, //índice da última tarefa alocada na máquina "m"
  ultimaposicao: vetordinamico; //índice da última posição alocada na máquina "m"
  tempop, //tempos de processamento das tarefas em cada estágio
  tempoc, //data de término da tarefa "i" no estágio "k"
  programacao, //programação de cada máquin a (posição "i", máquina "m")
  setup: matrizdinamica3; //tempo de setup da tarefa "i" para "j" no estágio "k"

procedure TFormFSH.ProgramacaoFSH(procedimento: integer);
//procedimento: 11 = LPT, 12 = TOTAL, 21 = SCT, 22 = SCT/LPST
var
      i, j, k, kk, m, x, y,//auxiliares para loops
      auxtarefa, makespan: integer;
begin
```

```
        //////// PROGRAMAÇÃO DAS TAREFAS NAS MÁQUINAS
            InicializaVetores;

            case procedimento of
                11: Procedimento1(1);
                12: Procedimento1(2);
                21: Procedimento2(1);
                22: Procedimento2(2);
            end;

        //////// CÁLCULO DO MAKESPAN
            makespan := 0;
            for i := 1 to numtarefas do begin
                if tempoc[i,numestagios] > makespan then
                    makespan := tempoc[i,numestagios];
            end;

        end; //for "x"
end;

procedure TFormFSH.RegraLPT;
var
    i, j, xtarefa: integer;
begin
    for i := 1 to numtarefas-1 do begin
        for j := i+1 to numtarefas do begin
            if somap[ordenacao[i]] < somap[ordenacao[j]] then begin
                xtarefa := ordenacao[i];
                ordenacao[i] := ordenacao[j];
                ordenacao[j] := xtarefa;
            end; //if
        end; //for "j"
    end; //for "i"
end;
```

```
procedure TFormFSH.RegraSRD(estagio: integer);
var
    i, j,
    xtarefa: integer;
begin
    for i := 1 to numtarefas-1 do begin
        for j := i to numtarefas do begin
            if tempoc[ordenacao[i],estagio-1] > tempoc[ordenacao[j],estagio-1] then
begin
                xtarefa := ordenacao[i];
                ordenacao[i] := ordenacao[j];
                ordenacao[j] := xtarefa;
            end; //if
        end; //for "j"
    end; //for "i"
end;

procedure TFormFSH.MetodoSimons;
var
    i, j, kk, l, menor1, menor2, maior, menor, lin, col, tarefa, ant,
    continua: integer; //controla construção da seqüência
    sucessora: vetordinamico; //vai guardando subseqüências das iterações (Simons Jr.)
    total: matrizdinamica2; //matriz com soma dos "p" e "s" de cada tarefa (Simons Jr.)
    procura: boolean;
begin
    //constrói matriz "total[i,j]"
    SetLength(total, numtarefas+1, numtarefas+1);
    for i := 0 to numtarefas do begin //inicializa matriz "total[i,j]"
        for j := 0 to numtarefas do
            total[i,j] := 0;
    end;

    for i := 1 to numtarefas do begin
```

```
        for j := 1 to numtarefas do begin
            //diagonal é zero, outros elementos:
            if i <> j then begin
                for kk := 1 to numestagios do
                    total[i,j] := total[i,j] + tempop[j,kk] + setup[i,j,kk];
            end; //if
        end; //for "j"
    end; //for "i"

    //inicializa matriz "sucessora[i]"
    SetLength(sucessora, numtarefas+1);
    for i := 1 to numtarefas do
        sucessora[i] := 0;

    continua := numtarefas-1;

    repeat begin
        //calcula diferenças dos dois menores elementos (exceto zero)
        //de cada LINHA e guarda na coluna "0"
        menor1 := infinito;
        menor2 := infinito;
        col := 0;

        for i := 1 to numtarefas do begin
            for j := 1 to numtarefas do begin //encontra menor elemento da linha
                if (total[i,j] <> 0) and (total[i,j] < menor1) then begin
                    menor1 := total[i,j];
                    col := j; //para ñ pegar mesmo núm. em "menor2"
                end;//if
            end; //for "j"

            if menor1 = infinito then //linha só tem zero
                menor1 := 0;
```

```
        for l := 1 to numtarefas do begin //encontra 2º menor elemento da linha
            if l <> col then begin
                if (total[i,l] <> 0) and (total[i,l] < menor2) then
                    menor2 := total[i,l];
            end;// if "l"
        end; // for "l"

        if menor2 = infinito then //linha só tem um elemento <> 0
            menor2 := 0;

        total[i,0] := menor1  - menor2;

        if total[i,0] < 0 then
            total[i,0] := total[i,0] * (-1);

        menor1 := infinito;
        menor2 := infinito;
    end; // for "i"

    //calcula diferenças dos dois menores elementos (exceto zero)
    //de cada COLUNA e guarda na linha "0"
    menor1 := infinito;
    menor2 := infinito;
    lin := 0;

    for j := 1 to numtarefas do begin
        for i := 1 to numtarefas do begin //encontra menor elemento da coluna
            if (total[i,j] <> 0) and (total[i,j] < menor1) then begin
                menor1 := total[i,j];
                lin := i; //para ñ pegar mesmo núm. em "menor2"
            end;//if
        end; //for "i"

        if menor1 = infinito then //coluna só tem zero
```

```
            menor1 := 0;

        for l := 1 to numtarefas do begin //encontra 2º menor elemento da coluna
            if l <> lin then begin
                if (total[l,j] <> 0) and (total[l,j] < menor2) then
                    menor2 := total[l,j];
            end;// if "l"
        end; // for "l"

        if menor2 = infinito then //coluna só tem um elemento <> 0
            menor2 := 0;

        total[0,j] := menor1 - menor2;

        if total[0,j] < 0 then
            total[0,j] := total[0,j] * (-1);

        menor1 := infinito;
        menor2 := infinito;
    end; //for "j"

    //identifica maior diferença e sua posição
    maior := 0;

    for i := 1 to numtarefas do begin //verifica coluna "0"
        if total[i,0] > maior then begin
            maior := total[i,0];
            lin := i;
            col := 0;
        end;
    end; //for "i"

    for j := 1 to numtarefas do begin //verifica linha "0"
        if total[0,j] > maior then begin
```

```
            maior := total[0,j];
            lin := 0;
            col := j;
        end;
    end; //for "j"

    //seleciona elemento com menor valor (exceto zero)
    //da linha ou coluna com maior diferença
    menor := infinito;

    if lin <> 0 then begin //se estiver na coluna
        for j := 1 to numtarefas do begin
            if (total[lin,j] <> 0) and (total[lin,j] < menor) then begin
                menor := total[lin,j];
                col := j;
            end;
        end; //for "j"
    end
    else begin //se estiver na linha
        for i := 1 to numtarefas do begin
            if (total[i,col] <> 0) and (total[i,col] < menor) then begin
                menor := total[i,col];
                lin := i;
            end;
        end; //for "i"
    end; //else

    //guarda subseqüência no vetor "sucessora[i]"
    sucessora[lin] := col;

    //atualiza matriz "total[i,j]"
    for i := 1 to numtarefas do begin
        total[lin,i] := 0;
        total[i,col] := 0;
```

```
        end;

        total[col,lin] := 0;
        continua := continua-1;
end //repeat
until continua = 0;

//coloca seqüência obtida no vetor "ordenacao[i]"
tarefa := 1;
while sucessora[tarefa] <> 0 do
        tarefa := tarefa+1;

ordenacao[numtarefas] := tarefa;

for i := numtarefas-1 downto 1 do begin
        ant := 0;
        for j := 1 to numtarefas do begin
                if sucessora[j] = tarefa then begin
                        ant := j;
                        sucessora[j] := 0;
                end;
        end;

        if ant = 0 then begin
                ant := 1;
                procura := true;
                while procura do begin
                        if (sucessora[ant] = 0) and (ant < numtarefas) then
                                ant := ant + 1
                        else begin
                                ant := sucessora[ant];
                                procura := false;
                        end;
                end; //while
```

```
            end; //if "ant"

            ordenacao[i] := ant;
            tarefa := ant;
        end; //for "i"
end;

procedure TFormFSH.Procedimento1(regra: integer);
var
        i, k, m, tarefaatual, auxmaq, maq, preparacao, termino, datatermino: integer;
begin
//////// ORDENAÇÃO INICIAL
        if regra = 1 then begin
            RegraLPT;
        end
        else begin // regra = 2
            MetodoSimons;
        end;

//////// PROGRAMAÇÃO
        auxmaq := 0;
        for k := 1 to numestagios do begin
            if k > 1 then begin// ETAPA 2 - reordena tarefas pela regra SRD
                RegraSRD(k);
            end;

            // designa uma tarefa de cada vez à máquina com data mais cedo de término
            for i := 1 to numtarefas do begin
                tarefaatual := ordenacao[i];
                termino := infinito;
                maq := 0;

                for m := auxmaq to auxmaq+mk[k]-1 do begin
                    if (tempoc[ultimatarefa[m],k] + setup[ultimatarefa[m],tarefaatual,k])
```

```
                                    > tempoc[tarefaatual,k-1] then
                                preparacao := tempoc[ultimatarefa[m],k] +
                                              setup[ultimatarefa[m],tarefaatual,k]
                            else preparacao := tempoc[tarefaatual,k-1];

                            datatermino := preparacao + tempop[tarefaatual,k];

                            if termino > datatermino then begin
                                termino := datatermino;
                                maq := m;
                            end; //if
                        end; //for "m"

                        tempoc[tarefaatual,k] := termino;
                        ultimatarefa[maq] := tarefaatual;
                        ultimaposicao[maq] := ultimaposicao[maq] + 1;
                        programacao[ultimaposicao[maq],maq] := tarefaatual;
                    end; //for "i"

                    auxmaq := auxmaq + mk[k];
                end; //for "k"
            end;

            procedure TFormFSH.Procedimento2(regra: integer);
            var
                i, j, k, m, auxmaq, maq, tar, menorcarga,  maximo, maiort,
                termino, datatermino, proximaliberacao: integer;
                jlinha, //J': conjunto das tarefas não programadas
                liberacao: vetordinamico; //contém datas de término das tarefas no estágio anterior
            begin
            /////// PROGRAMAÇÃO
                auxmaq := 0;
                SetLength(jlinha, numtarefas+1);
                SetLength(liberacao, numtarefas+1);
```

```
if regra = 1 then begin

    for k := 1 to numestagios do begin
        for i := 1 to numtarefas do begin //coloca todas as tarefas no conj. J'
            jlinha[i] := 1;
            liberacao[i] := tempoc[i,k-1];
        end;

        maximo := MaxIntValue(liberacao);
        liberacao[0] := maximo;

        for i := 1 to numtarefas do begin //seleciona par tarefa-máquina
            termino := infinito;
            maq := 0;
            tar := 0;

            proximaliberacao := MinIntValue(liberacao);
            for j := 1 to numtarefas do begin
                if (jlinha[j] <> 0) and (tempoc[j,k-1] <= proximaliberacao)
                    then begin
                    //tarefas não programadas e já liberadas
                    for m := auxmaq to auxmaq+mk[k]-1 do begin
                    //calcula data de término para cada par tarefa-máquina
                        if (tempoc[ultimatarefa[m],k] +
                           setup[ultimatarefa[m],j,k]) < tempoc[j,k-1] then
                               datatermino := tempoc[j,k-1] + tempop[j,k]
                        else datatermino := tempoc[ultimatarefa[m],k] +
                                   setup[ultimatarefa[m],j,k] + tempop[j,k];

                        if termino > datatermino then begin
                            termino := datatermino;
                            maq := m;
                            tar := j;
```

```
                                    end;//if "termino"
                                end;//for "m"
                            end;//if "jlinha"
                        end;//for "j"

                        tempoc[tar,k] := termino;
                        ultimatarefa[maq] := tar;
                        ultimaposicao[maq] := ultimaposicao[maq]+1;
                        programacao[ultimaposicao[maq],maq] := tar;
                        jlinha[tar] := 0; //tira tarefa do conj. J'
                        liberacao[tar] := maximo; //tarefa programada
                    end;//for "i"

                    auxmaq := auxmaq + mk[k];

            end;//for "k"
        end
        else begin//regra = 2
            for k := 1 to numestagios do begin
                for i := 1 to numtarefas do begin //coloca todas as tarefas no conj. J'
                    jlinha[i] := 1;
                    liberacao[i] := tempoc[i,k-1];
                end;

                maximo := MaxIntValue(liberacao);
                liberacao[0] := maximo;

                for i := 1 to numtarefas do begin //seleciona par tarefa-máquina
                    termino := infinito;
                    maq := 0;
                    tar := 0;

                    proximaliberacao := MinIntValue(liberacao);
```

```
if proximaliberacao = maximo then begin //todas as tarefas liberadas
      menorcarga := infinito;
      for m := auxmaq to auxmaq+mk[k]-1 do begin
            if tempoc[ultimatarefa[m],k] < menorcarga then begin
                  menorcarga := tempoc[ultimatarefa[m],k];
                  maq := m;
            end; //if "tempoc"
      end;//for "m"

      maiort := 0;
      for j := 1 to numtarefas do begin //tarefa com maior (sijk+pik)
            if (jlinha[j] <> 0) and
            (maiort < setup[ultimatarefa[maq],j,k] + tempop[j,k]) then
            begin
                  maiort := setup[ultimatarefa[maq],j,k] + tempop[j,k];
                  tar := j;
            end; //if "jlinha"
      end; //for "j"

      if (tempoc[ultimatarefa[maq],k] +
            setup[ultimatarefa[maq],tar,k]) < tempoc[tar,k-1] then
            termino := tempoc[tar,k-1] + tempop[tar,k]
      else termino := tempoc[ultimatarefa[maq],k] +
                  setup[ultimatarefa[maq],tar,k] + tempop[tar,k];
end
else begin//existem tarefas não liberadas
      for j := 1 to numtarefas do begin
            if (jlinha[j] <> 0) and (tempoc[j,k-1] <=
                  proximaliberacao) then begin
                  //tarefas não programadas e já liberadas
                  for m := auxmaq to auxmaq+mk[k]-1 do begin
                  //calcula data de término para cada par tarefa-máquina
                        if (tempoc[ultimatarefa[m],k] +
```

```
                                        setup[ultimatarefa[m],j,k])  <  tempoc[j,k-1]
                                        then
                                         datatermino := tempoc[j,k-1] + tempop[j,k]
                                      else datatermino := tempoc[ultimatarefa[m],k] +
                                                setup[ultimatarefa[m],j,k] +
                                                tempop[j,k];

                                      if termino > datatermino then begin
                                            termino := datatermino;
                                            maq := m;
                                            tar := j;
                                      end;//if "termino"
                                  end;//for "m"
                              end;//if "jlinha"
                         end;//for "j"
                    end;//if-else

                    tempoc[tar,k] := termino;
                    ultimatarefa[maq] := tar;
                    ultimaposicao[maq] := ultimaposicao[maq]+1;
                    programacao[ultimaposicao[maq],maq] := tar;
                    jlinha[tar] := 0; //tira tarefa do conj. J'
                     liberacao[tar] := maximo; //tarefa programada
               end;//for "i"

               auxmaq := auxmaq + mk[k];

          end;//for "k"
     end;
end;

end.
```