

DEDALUS - Acervo - EESC



31100013455

ANÁLISE COMPARATIVA DE ALGORITMOS
GENÉTICOS PARA O PROBLEMA DE
PROGRAMAÇÃO DE OPERAÇÕES FLOW
SHOP PERMUTACIONAL



Autor: Eng. Wagner Lisboa Mota

Dissertação apresentada à área de pós-graduação em Engenharia de Produção da Escola de Engenharia de São Carlos - USP, como parte dos requisitos para obtenção do título de mestre em Engenharia.

Área: Engenharia de Produção.

Orientador: Prof. Titular João Vitor Moccellini

São Carlos
1996

engenharia de Produção

Class. TESE/EESC
Cutt. 113789
Tombo T040/97

st 0746957

Ficha catalográfica preparada pela seção de Tratamento
da informação do Serviço de Biblioteca- EESC-USP

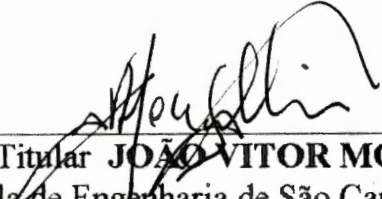
M917a Mota, Wagner Lisboa
Análise comparativa de algoritmos genéticos
para o problema de programação de operações flow
shop permutacional / Wagner Lisboa Mota. -- São
Carlos, 1996.

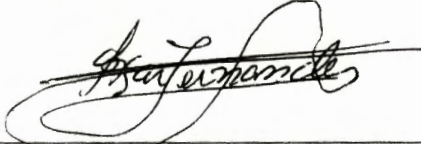
Dissertação (Mestrado). -- Escola de Engenharia
de São Carlos-Universidade de São Paulo, 1996.
Orientador: Prof. Titular João Vitor Moccellin.

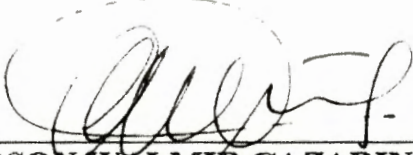
1. Programação. 2. Flow shop permutacional.
3. Algoritmo genético. I. Título

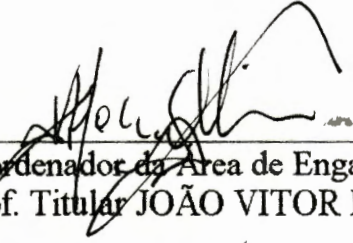
FOLHA DE APROVAÇÃO

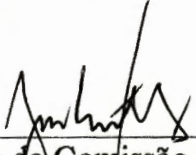
Dissertação defendida e aprovada em 18-12-1996
pela Comissão Julgadora:


Prof. Titular **JOÃO VITOR MOCCELLIN (Orientador)**
(Escola de Engenharia de São Carlos - Universidade de São Paulo)


Prof. Doutor **FLÁVIO CESAR FARIA FERNANDES**
(Universidade Federal de São Carlos - UFSCar)


Prof. Doutor **EDSON WALMIR CAZARINI**
(Escola de Engenharia de São Carlos - Universidade de São Paulo)


Coordenador da Área de Enga. de Produção
Prof. Titular **JOÃO VITOR MOCCELLIN**


Presidente da Comissão de Pós-Graduação
JOSÉ CARLOS A. CINTRA

Dedico este trabalho:

- *À minha mãe, Maria José;
- *Ao Prof. Titular João Vitor Moccellini;
- *À Vanessa;
- *À Ísis;
- *A todas as áreas da engenharia que com sua beleza e utilidade proporcionam uma vida com mais comodidade;
- *Ao CNPq, pelo seu papel fundamental na pesquisa científica brasileira.

AGRADECIMENTOS:

*A Deus por ter me dado forças para prosseguir nesta jornada;

*A minha mãe, pelo seu exemplo de pessoa batalhadora;

*Ao Professor e amigo Dr. João Vitor Moccellin, pela paciência que manteve comigo durante todos esses anos de mestrado e pela orientação;

A todos que de forma positiva ou negativa me incentivaram para o término deste trabalho. Deus abençoe essas pessoas.

SUMÁRIO

RESUMO	iv
ABSTRACT	v
1. INTRODUÇÃO.....	1
1.1. ASPECTOS GERAIS DOS SISTEMAS DE PRODUÇÃO	1
1.2. PLANEJAMENTO E CONTROLE DA PRODUÇÃO	5
2. O PROBLEMA DE PROGRAMAÇÃO DE OPERAÇÕES FLOW-SHOP PERMUTACIONAL.....	11
2.1. PRINCIPAIS HIPÓTESES DO PROBLEMA.....	11
2.2. MÉTODOS HEURÍSTICOS PARA A SOLUÇÃO DO PROBLEMA FLOW-SHOP PERMUTACIONAL.....	13
3. ALGORITMOS GENÉTICOS.....	22
3.1. INTRODUÇÃO	22
3.2. ALGORITMO GENÉTICO SIMPLES	24
3.2.1 - OPERADOR REPRODUÇÃO	24
3.2.2 - OPERADOR CRUZAMENTO (CROSSOVER)	25
3.2.3 - OPERADOR MUTAÇÃO	26
3.3 - ADEQUAÇÃO DO ALGORITMO GENÉTICO AO PROBLEMA DE PROGRAMAÇÃO DE OPERAÇÕES FLOW-SHOP PERMUTACIONAL	26
3.4 - OS OBJETIVOS DO TRABALHO	28
4. ALGORITMOS GENÉTICOS PROPOSTOS	29
4.1. O ALGORITMO GENÉTICO SIMPLES.....	29
4.2. APRESENTAÇÃO DOS ALGORITMOS PROPOSTOS.....	31
4.2.1 - OPERADOR PMX(PARTIALLY MATCHED CROSSOVER).....	32
4.2.2 - OPERADOR MPX(MAXIMAL-PREVENTIVE) :	34
4.2.3 - OPERADOR ORDER CROSSOVER(OX) :	34
4.2.4 - OPERADOR ENHANCE ORDER-CROSSOVER(EOX).....	35
4.2.5 - OPERADOR DE UM CORTE:.....	36
4.2.6 - OPERADOR CYCLE CROSSOVER(CX) :	36
4.2.7 - OPERADOR LINEAR ORDER CROSSOVER (LOX).....	37
5. EXPERIMENTAÇÃO COMPUTACIONAL E ANÁLISE DOS RESULTADOS	39
5.1. EXPERIMENTAÇÃO COMPUTACIONAL	39
5.2. RESULTADOS OBTIDOS.....	43
5.2.1. SEMENTES ALEATÓRIAS.....	43
5.2.2. SEMENTES NÃO-ALEATÓRIAS.....	52
5.2.3. LIMITANTES INFERIORES CALCULADOS.....	60
5.3. ANÁLISE DOS RESULTADOS OBTIDOS.....	67
6. CONCLUSÕES E CONSIDERAÇÕES FINAIS.....	89
REFERÊNCIAS BIBLIOGRÁFICAS	91
APÊNDICE I : PROGRAMAS COMPUTACIONAIS.....	96
APÊNDICE II : SOLUÇÕES DOS PROBLEMAS DO EXPERIMENTO	106

RESUMO

Este trabalho tem como meta avaliar a técnica de Algoritmo Genético aplicada ao problema de programação de operações Flow Shop permutacional, para isso tomou-se da literatura alguns operadores genéticos e com eles criou-se vários algoritmos, com o intuito de avaliar o desempenho destes entre si. As sementes iniciais (soluções iniciais) foram objeto de estudo também deste trabalho, observou-se a performance dos algoritmos, quando da utilização de boas soluções e soluções aleatórias.

ABSTRACT

This work has as a goal evaluate the Genetic Algorithm Technique applied to the Permutation Flow Shop Scheduling Problem, for this we took from the literature some genetics operators and with them created several algorithms trying to evaluate the performance of this operators. The initial seeds (initial solution) was also studied in this work, we one has observed the performance of this algorithms using good initial solution and random initial solution.

1. INTRODUÇÃO

1.1. ASPECTOS GERAIS DOS SISTEMAS DE PRODUÇÃO

A definição de sistema depende do interesse da pessoa que pretenda analisá-lo, mas pode-se esquematizar um sistema utilizando a figura 1.1

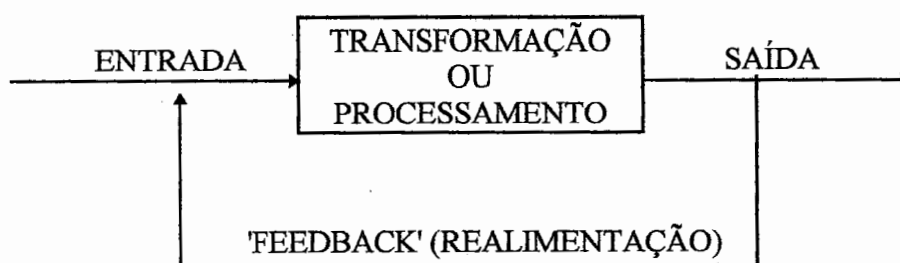


FIGURA 1.1 : Esquema geral de um sistema.

BUFFA(1975) e VASCONCELLOS(1979) definem produção como sendo um processo, através do qual bens e serviços são gerados, por meio de transformação de recursos. Pode-se então utilizar as duas definições acima para concluir que um sistema de produção terá alguns parâmetros que o caracterizam, como:

- a) ENTRADAS ('INPUTS') : Matéria-prima, passageiros, informações, etc;
- b) PROCESSAMENTO ou TRANSFORMAÇÃO('THROUGHPUT') : Fabricação, transporte, consultoria, etc; .
- c) SAÍDAS('OUTPUTS') : Peças acabadas, conjuntos montados, etc;
- d) FEEDBACK(REALIMENTAÇÃO): Visa manter ou aperfeiçoar o desempenho do processo.

Os sistemas de produção podem ser classificados de diversas maneiras. KATZ & KAHN(1970) e MACHLINE et al (1976) fazem uma classificação geral dos sistemas de produção utilizando alguns critérios como : Atividades econômicas, quanto aos bens de consumo, quanto às dimensões dos produtos, etc.

JOHNSON & MONTGOMERY(1974) classificam os sistemas de produção segundo o processo, dividindo em três categorias:

a) Sistema GRANDE PROJETO : Tem como característica a produção de itens complexos e/ou de grande porte;

b) Sistema CONTÍNUO : Tem como característica a produção em larga escala de produtos padronizados; e

c) Sistema INTERMITENTE : Tem como característica a flexibilidade, ou seja, a capacidade de produzir um grande número de produtos. Aqui destacam-se dois sub-itens:

c.1) 'FLOW-SHOP' : Os itens fabricados em uma linha ou célula de manufatura têm a mesma seqüência de operações nas diversas máquinas; e

c.2) 'JOB-SHOP' : A seqüência de operações se modifica de um produto para outro.

Segundo PIRES(1989) : 'Os sistemas de produção se classificam, com base no volume produtivo em dois grandes grupos: produção contínua e produção intermitente'. Um exemplo claro de sistema contínuo é a produção de cerveja e um exemplo de sistema intermitente é a indústria eletro-mecânica. Ainda segundo PIRES : 'Os sistemas de produção intermitente podem-se classificar em produção em massa(componentes/produtos padronizados), em lotes(autopeças) e individual(grandes projetos). Os lotes podem ainda ser classificados em pequenos,

médios e grandes, não existindo, porém, um critério para se fazer essa classificação, ou seja, o que é considerado pequeno lote numa indústria pode ser considerado grande em outra'.

Pode-se caracterizar dentro de qualquer classificação acima de sistema de produção a existência de três subsistemas interativos:

- a) Estrutural : formado pela parte física do sistema, como matéria-prima, instalações, etc;
- b) Social : formado pelas relações sociais entre as pessoas da empresa; e
- c) Organizacional : formado pela hierarquia de decisão, departamentos, etc.

Dentro do terceiro subsistema encontramos o Planejamento e Controle da Produção(P.C.P), que segundo ZACCARELLI(1986) é um conjunto de várias funções que tem por objetivo comandar e coordenar o processo produtivo.

Independentemente do sistema de produção, da tecnologia de processo e do sistema organizacional da fábrica, existem algumas funções que são inerentes ao Planejamento e Controle da Produção.

Essas funções são implementadas de diversas formas dependendo do produto, do layout da fábrica e etc . A influência na forma de implementar as funções do P.C.P., observando apenas os sistemas de produção citados acima, por PIRES, fica clara observando o quadro 1.1.

O sistema de P.C.P. ainda vai variar dependendo do tamanho da empresa, dos produtos fabricados, da organização departamental, etc. Algumas dessas funções serão mais desenvolvidas que outras até mesmo em indústrias de mesmo ramo de negócios.

<i>Características</i>	<i>Sistema de produção Contínuo</i>	<i>Sistema de produção Intermitente repetitivo</i>	<i>Sistema de Produção Intermitente em lotes</i>
<i>- O tempo de preparação do equipamento é em relação ao tempo de operação</i>	pequeno	indeterminado	indeterminado
<i>- A quantidade produzida de artigos iguais é</i>	grande	variável	pequena
<i>- As máquinas são agrupadas em geral de acordo com</i>	o produto que fabricam	o processo que utilizam	o processo que utilizam
<i>- A capacidade das máquinas ser perfeitamente calibradas</i>	precisa	não precisa	não precisa
<i>A quantidade de instruções de serviço necessária é</i>	pequena	grande	grande
<i>- Em relação a quantidade produzida o estoque de matéria-prima e de produtos em elaboração é</i>	pequeno	grande	pequeno
<i>- A capacidade ociosa da fábrica é geralmente</i>	pequena	média	grande
<i>- A produção é feita</i>	para estoques	para estoques	sob encomendas (geralmente)
<i>- O movimento de material dentro da fábrica é</i>	rápido	lento	lento
<i>- O transporte é geralmente movimentado por</i>	transportadores contínuos	carrinhos	carrinhos

Quadro 1.1: Influência de alguns parâmetros nas tarefas a serem comandadas pelo P.C.P.

1.2. PLANEJAMENTO E CONTROLE DA PRODUÇÃO

Como foi dito anteriormente o P.C.P. é um conjunto de várias funções, e não é finalidade deste trabalho detalhar estas funções, portanto, neste capítulo, reporta-se resumidamente algumas destas funções. Estas funções variam de nome de autor para autor, sendo os nomes dados aqui, os que mais aparecem na literatura.

PLANEJAMENTO DE RECURSOS DE LONGO PRAZO

O ponto de partida para qualquer planejamento e controle da produção é o planejamento dos recursos de longo prazo. Toda empresa competitiva deve planejar seu futuro, observando, através de estudos de predição e previsão, a sua capacidade futura e as dimensões de seus negócios à frente.

PLANEJAMENTO AGREGADO DA PRODUÇÃO

De posse do planejamento de longo prazo, elabora-se o que chamamos de planejamento agregado da produção. Este é um plano de médio prazo onde fica estabelecido níveis gerais de produção, dimensões da mão-de-obra e níveis de estoque. O horizonte de planejamento pode variar de 6 a 24 meses dependendo do tipo de indústria considerada.

PLANO MESTRE DE PRODUÇÃO

Aqui tem-se um plano de curto prazo, que é baseado no planejamento agregado de produção. O objetivo do plano mestre é especificar o produto a ser produzido em um certo período e qual o grupo de trabalho que o fabricará. HIGGINS & BROWNE(1992) salientam que o plano mestre de produção é o principal elo de ligação entre as áreas de Marketing, Manufatura e Engenharia.

PLANEJAMENTO DAS NECESSIDADES DE MATERIAIS

Para a execução do plano mestre necessitamos saber quais os materiais a serem utilizados. Com esta informação prepara-se um plano detalhado para aquisição, por meio de compra ou fabricação de todas as matérias-primas e componentes, observando as datas de recebimento ou término de fabricação e as quantidades.

CONTROLE DE ESTOQUE

Segundo ZACCARELLI(1986), as seguintes razões justificam a manutenção de estoques:

- a) Garantir a continuidade ou presteza de fornecimento, evitando situações de demora de fornecimento, suprimento sazonal e riscos de falhas no suprimento;

- b) Possibilitar economias em dinheiro pela compra ou fabricação de lotes econômicos e por possibilitar flexibilidade de processos de manufatura.

PLANEJAMENTO E CONTROLE DA CAPACIDADE

De posse do plano mestre de produção e feito o planejamento das necessidades de materiais, agora deve-se verificar quais os centros produtivos que têm capacidade para executar o plano. Calculada a carga de cada centro de produção, pode-se encontrar gargalos e promover ajustes dos recursos.

O controle da capacidade basicamente cuida das providências para que a capacidade planejada seja realizada (através de ações no chão de fábrica como ativar e desativar máquinas).

LIBERAÇÃO DE ORDENS

A liberação de ordens tem como objetivo emitir instruções para os operadores, desde qual matéria-prima a ser usada, até qual o acabamento final.

ZACCARELLI(1986) afirma que a liberação da produção é um conjunto de funções para:

- a) Verificar a disponibilidade de materiais, ferramentas e instruções técnicas, para as ordens de fabricação serem iniciadas e providenciar para que fiquem à disposição do operador;
- b) Decidir sobre a seqüência de processamento das ordens de fabricação;
- c) Distribuir ordenadamente as vias componentes das ordens de fabricação;
- d) Coletar informações para controle.

PROGRAMAÇÃO/ SEQÜENCIAMENTO DA PRODUÇÃO

BURBIDGE(1986) define programação 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

simplesmente que programação é a alocação de recursos para a execução de tarefas, utilizando uma escala de tempo.

Seqüenciamento se refere à ordenação de tarefas (operações) em uma certa máquina.

Dentro de uma situação de programar as operações nas máquinas disponíveis surgem problemas complexos de programação. Dentro destes problemas, as restrições tecnológicas e a medida de desempenho da programação devem ser especificadas. As restrições tecnológicas são determinadas principalmente pelo fluxo das tarefas nas máquinas. Dentro deste contexto MACCARTHY & LIU (1993) classificaram os problemas de programação de operações da seguinte forma:

- a) Job-Shop: Cada tarefa tem sua própria seqüência de processamento nas máquinas;
- b) Flow-Shop: Todas as tarefas têm a mesma seqüência de processamento nas máquinas;
- c) Open-Shop: Não há uma seqüência específica ou preestabelecida para as tarefas serem processadas nas máquinas;
- d) Flow-Shop Permutacional: É um Flow-Shop no qual a seqüência das tarefas é a mesma em todas as máquinas;
- e) Máquina única: Existe apenas uma única máquina disponível;
- f) Máquinas paralelas: Há disponível k máquinas idênticas;
- g) Job-Shop com máquinas múltiplas: É um Job-Shop no qual existe K_i máquinas idênticas em cada estágio ($i = 1, 2, \dots, m$) e algumas tarefas que necessitam destes estágios sejam processadas em uma e apenas uma dessas

máquinas. A figura 1.2 ilustra esquematicamente a relação entre as diversas classes de problemas.

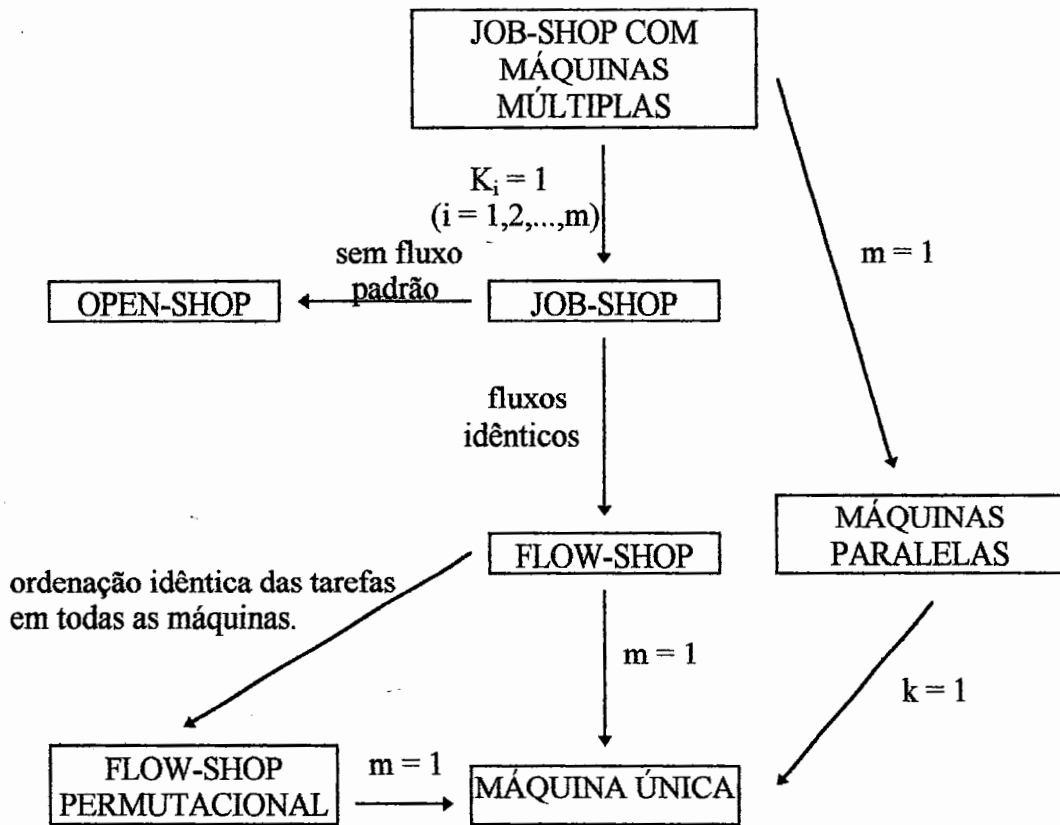


Figura 1.2: Esquema que relaciona as diversas classes de problemas de programação de máquinas (MACCARTHY & LIU, 1993).

O presente trabalho tem por objetivo analisar a utilização da técnica do Algoritmo Genético para a solução do problema de programação de operações Flow-Shop Permutacional.

Dividiu-se o trabalho da seguinte forma: No capítulo 2, definiu-se o problema de programação de operações flow-shop permutacional, as suas principais hipóteses e os métodos mais conhecidos para resolvê-lo. No capítulo 3, conceituou-se o Algoritmo Genético, as suas principais características e os objetivos detalhados deste trabalho. No capítulo seguinte apresenta-se os algoritmos genéticos propostos no âmbito deste trabalho. O capítulo 5 contém a experiência computacional e

análise dos resultados. No capítulo 6 relata-se as conclusões e considerações finais. Logo após o capítulo 6, faz-se as referências bibliográficas. No apêndice I encontram-se os programas computacionais e no apêndice II encontram-se as soluções de todos os problemas utilizados no experimento.

2. O PROBLEMA DE PROGRAMAÇÃO DE OPERAÇÕES FLOW-SHOP PERMUTACIONAL

2.1. PRINCIPAIS HIPÓTESES DO PROBLEMA

Foi visto no capítulo anterior que programação é a alocação de recursos para a execução de tarefas, utilizando uma escala de tempo. A importância de uma boa estratégia de programação num meio de produção não deve ser esquecida nestes dias em que a concorrência acirrada e a globalização dos mercados são uma realidade. A necessidade em responder rapidamente às demandas do mercado e gerenciar a fábrica eficientemente nos geram problemas de programação complexos.

Mesmo observando a importância da programação, é mínima a porcentagem de indústrias que se utilizam da teoria de programação, ou por não confiar ou por desconhecimento. Deve-se deixar claro que em todos os modelos e estudos feitos em programação, sem nenhuma exceção, encontram-se hipóteses que nem sempre são aceitas na fábrica 'real'. Estas hipóteses serão relatadas a seguir.

Foi definido no capítulo anterior o problema de programação de operações Flow-Shop. Colocando em outras palavras, pode-se afirmar que é um problema no qual cada uma das 'n' tarefas deve ser processada, na mesma seqüência, em cada máquina de um conjunto de 'm' máquinas distintas. Quando em cada máquina a ordem de processamento das tarefas é a mesma, tem-se o Flow-Shop Permutacional. Este problema é geralmente encontrado em fábricas onde as peças são movidas de uma máquina para outra através de algum equipamento de movimentação de materiais.

O modelo de programação Flow-Shop Permutacional possui as seguintes hipóteses principais:

- a) As máquinas estão disponíveis e não são previstas eventuais quebras;
- b) Cada máquina pode processar apenas uma tarefa de cada vez;
- c) Cada tarefa pode ser processada por uma máquina de cada vez;
- d) Os tempos de liberação de todas as tarefas são nulos, isto é, todas as tarefas estão disponíveis para o início do processamento;
- e) As operações nas diversas máquinas, uma vez iniciadas não devem ser interrompidas (No Pre-emption);
- f) Os tempos de preparação (Setup Times) das operações nas máquinas são incluídos nos tempos de processamento e independem da programação; e
- g) Os tempos de processamento e as restrições tecnológicas são determinados e fixos.
- h) Não existe sobreposições de operações (No-Overlapping)

A solução deste problema consiste na realidade encontrar dentre as $(n!)$ seqüências viáveis das tarefas, aquela que minimiza o intervalo de tempo entre o início de execução da primeira tarefa na primeira máquina e o término de execução da última tarefa na última máquina, chamado de duração total da programação (MAKESPAN).

Devido a sua natureza combinatorial, o problema Flow-Shop Permutacional é de difícil resolução ótima para casos de médio e grande porte, sendo classificado como 'NP-HARD'. Por exemplo, em um problema envolvendo 10 tarefas, o número de programações viáveis é de 3.628.800.

Em relação a problemas de duas máquinas, JOHNSON (1954) apresenta um algoritmo que fornece a solução ótima. Para o caso de 3 máquinas, a solução ótima pode ser obtida, sob certas restrições, quanto aos tempos de processamento.

O problema de programação Flow-Shop Permutacional tem sido estudado e pesquisado há quatro décadas e várias técnicas de programação matemática, tais como programação linear inteira (SELEN & HOTT, 1986; WILSON, 1989) e técnicas de enumeração do tipo Branch-and-Bound (IGNALL & SCHRAGE, 1965 ; LAGEWEG et al, 1978; POTTS, 1980) têm sido empregadas para encontrar a solução ótima, sendo estas técnicas nada eficientes, em termos computacionais, em problemas de médio e grande porte. Assim sendo, vários métodos heurísticos têm sido propostos para a solução do problema em questão.

Este tipo de método, chamado heurístico, não pode garantir a qualidade da solução, sendo aceitável se a resposta for satisfatória em relação ao esforço computacional e à distância da solução ótima. Um método heurístico, em geral, é fortemente dependente das particularidades do problema e da distribuição dos dados, que neste caso são os tempos de processamento das tarefas. Estes métodos podem ser classificados em dois tipos:

- a) Algoritmos construtivos: Têm a característica de gerar uma única solução;
- b) Algoritmos melhorativos: Têm a característica de iniciar com uma solução viável e ir melhorando, até não haver mudanças significativas na solução que possam justificar o esforço computacional.

2.2. MÉTODOS HEURÍSTICOS PARA A SOLUÇÃO DO PROBLEMA FLOW-SHOP PERMUTACIONAL

A seguir, são apresentadas as notações usuais e descritos resumidamente os métodos heurísticos mais conhecidos e referenciados na literatura.

Seja $J = \{J_1, J_2, \dots, J_i, \dots, J_n\}$ o conjunto das n tarefas que serão processadas, na mesma seqüência, por um conjunto de m máquinas distintas. O tempo de processamento de uma tarefa J_i na máquina k é p_{ki} ($i = 1, 2, \dots, n$ e $k = 1, 2, \dots, m$). Zero será o tempo de processamento da tarefa que não tiver operação em uma certa máquina.

PALMER (1965) sugeriu um fator chamado 'slope index', o qual estabelecia a seqüência de processamento das tarefas nas máquinas. Este índice é calculado de forma que as tarefas cujos tempos de processamento tendem a crescer na seqüência das máquinas, devem receber maior prioridade na programação, ou seja, devem ocupar as primeiras posições na ordem de execução das tarefas. O 'slope index' para uma tarefa J_i é dado por:

$$S_i = \sum_{k=1}^m (2k - m - 1) p_{ki} \quad \text{para } i = 1, 2, \dots, n$$

De acordo com a ordenação não-crescente dos fatores, através dos valores de S_i , estabelece-se a seqüência de programação das tarefas.

CAMPBELL, DUDEK & SMITH (1970) definiram um algoritmo conhecido por CDS, que é basicamente uma extensão do algoritmo de Johnson para a solução ótima do problema com duas máquinas ($m = 2$). O CDS usa a regra de Johnson em $(m - 1)$ estágios, onde cada estágio é considerado um problema com duas máquinas, com tempos de processamento 'artificiais' P'_{1i} e P'_{2i} ($i = 1, 2, \dots, n$). No primeiro estágio temos $P'_{1i} = p_{1i}$ e $P'_{2i} = p_{mi}$, que é a regra de Johnson aplicada somente considerando a primeira e a última máquinas, sendo as outras desprezadas. No estágio 2, $P'_{1i} = p_{1i} + p_{2i}$ e $P'_{2i} = p_{(m-1)i} + p_{mi}$, utiliza-se a regra de Johnson à soma dos tempos de processamento da primeira com a segunda máquina e da penúltima com a última máquina. No estágio t , então, os tempos de processamento artificiais serão:

$$p'_{1i} = \sum_{k=1}^i p_{ki} \text{ e } p'_{2i} = \sum_{k=1}^i p_{(m-k+1)i}$$

Em cada um dos $(m - 1)$ estágios, a seqüência de tarefas obtida pela regra de Johnson é utilizada para calcular a duração total da programação ('makespan') do problema original. A seqüência que fornecer a menor duração é escolhida como solução do problema.

GUPTA (1971) observou que o algoritmo de Johnson para o problema com 2 ou 3 máquinas é, na verdade, um método de ordenação a partir da definição de um fator para cada tarefa, sequenciando-as de acordo com a ordem crescente de tais fatores. Gupta generalizou o fator, para o caso de $m \geq 4$ máquinas, definindo, para cada tarefa J_i , o seguinte fator:

$$f(i) = \frac{A}{\min_{1 \leq k \leq m-1} (p_{ki} + p_{k+1,i})}$$

$$\text{onde } i = 1, 2, \dots, n \text{ e } A = \begin{cases} 1 & \text{se } p_{ki} \leq p_{li} \\ -1 & \text{se } p_{ki} > p_{li} \end{cases}$$

DANNENBRING (1977) apresentou um procedimento que procura combinar as vantagens do fator 'Slope Index' de Palmer e do método CDS, chamado de 'Rapid Access' (RA), apresentando uma boa solução de maneira simples e rápida. Ao invés de resolver $(m - 1)$ problemas artificiais com 2 máquinas, o método RA resolve um único problema, no qual os tempos de processamento do problema artificial, são determinados através de um método de ponderação, o qual é:

$$p'_{1i} = \sum_{k=1}^m (m - k + 1) p_{ki} \text{ e } p'_{2i} = \sum_{k=1}^m (K) p_{ki} \text{ para } K = 1, 2, \dots, n$$

Utilizando a solução fornecida pelo procedimento RA, Dannenbring propôs dois procedimentos, chamados RACS e RAES, que se encaixam no tipo melhorativo. Estes procedimentos procuram encontrar entre as seqüências 'vizinhas' daquela

fornecida pela solução inicial, uma seqüência cuja programação forneça uma menor duração total. Seqüência 'vizinha' é definida por Dannenbring como uma nova seqüência das tarefas, obtida a partir da solução inicial, fazendo-se uma troca de posições entre duas tarefas adjacentes. Utilizando-se o processo RACS, são examinadas $(n - 1)$ novas seqüências (vizinhas), adotando como solução a seqüência associada ao menor makespan, desde que tal makespan seja menor que o da solução inicial. O processo RAES, ao invés de terminar o procedimento de melhoria após examinar $(n - 1)$ seqüências vizinhas da solução inicial, usa a melhor seqüência vizinha para gerar as suas respectivas vizinhas, e assim sucessivamente até se encontrar uma seqüência cujas vizinhas não apresentam uma melhora na duração total da programação.

NAWAZ, ENSCORE Jr. & HAM (1983) desenvolveram um algoritmo conhecido por NEH baseado na hipótese de que a prioridade de programação de cada tarefa deve ser diretamente proporcional à soma dos seus tempos de processamento nas m máquinas. Pode-se descrever este algoritmo em quatro passos principais:

Passo 1 - Para cada tarefa, calcular a soma dos tempos de processamento em todas as máquinas;

Passo 2 - Ordenar as n tarefas de acordo com os valores não-crescentes das somas dos tempos de processamento;

Passo 3 - Selecionar as duas primeiras tarefas da ordenação, sequenciando-as de maneira a minimizar a duração total da programação, considerando-se somente essas duas tarefas;

Passo 4 - Para $K = 3$ a n , fazer:

- Selecionar a tarefa que ocupa a K -ésima posição na ordenação obtida no passo 2;

- Examinar as K possibilidades de acrescentar a tarefa na seqüência até então obtida, adotando aquela que leva a uma menor duração total da programação parcial

HUNDAL & RAJGOPAL (1988) observaram que o algoritmo de Palmer ignora a máquina $(m + 1)/2$ quando m é ímpar, o que pode afetar a qualidade da solução, especialmente quando o número de tarefas é grande. Então desenvolveram uma extensão dele. A extensão do algoritmo de Palmer é considerada a partir de dois novos conjuntos de fatores dados por:

$$S_i = \sum_{k=1}^m (2k - m) p_{ki} \quad e$$

$$S_i = \sum_{k=1}^m (2k - m - 2) p_{ki} \quad \text{para } i = 1, 2, \dots, n$$

Pode-se observar, que duas outras seqüências são obtidas, sendo utilizada a melhor.

WIDMER & HERTZ (1989) desenvolveram um método, do tipo melhorativo, denominado SPIRIT que a partir de uma solução inicial, busca obter uma melhor solução, examinando uma série de outras seqüências de tarefas, obtidas através do conceito de 'vizinhança'. A solução inicial do método SPIRIT é obtida através de uma analogia com o problema do caixeiro viajante, onde a distância entre duas tarefas (cidades) J_u e J_v é calculada pela seguinte expressão:

$$d_{uv} = p_{1u} + \sum_{j=2}^m (m - j) | p_{ju} - p_{j-1,v} | + p_{mv},$$

onde m = número de máquinas e p_{ji} = tempo de processamento da tarefa J_i na máquina j .

A segunda etapa do algoritmo, de melhoria da solução inicial, utiliza uma técnica denominada 'BUSCA TABU'. Este tipo de técnica é útil para encontrar uma boa solução, ou possivelmente a solução ótima, de problemas do tipo: 'Minimizar $c(x)$, sujeito a $x \in X$ ', onde $c(x)$ é uma função de uma variável discreta x , e X é o conjunto de soluções viáveis. O processo de 'BUSCA TABU' se inicia com uma solução viável atual de $x \in X$. A solução x se transforma em x' através da aplicação de uma função $m \in M(x)$, sendo x' uma nova solução viável ($x' = m(x)$). Esta transformação é chamada de mudança e o conjunto $\{x' : x' = m(x); x, x' \in X; m \in M(x)\}$ é denominado 'vizinhança' de x .

A fim de se evitar, o tanto quanto possível, a formação de ciclos nas mudanças, um fator t é associado a ' x ' e ' m '. Este fator define um conjunto de mudanças que são ditas 'TABU' (proibidas) e que são guardadas em uma lista denominada 'LISTA TABU'. O tamanho da lista tabu não pode crescer indefinidamente, sendo limitada por um parâmetro ' s ', chamado de 'tamanho da lista tabu'. Para se incluir um novo elemento na lista deve-se remover um outro, que é geralmente o mais antigo.

OSMAN & POTTS(1989) apresentaram alguns métodos heurísticos baseados na técnica denominada 'Simulated Annealing', advinda da física estatística. KIRKPATRICK et al. (1983) foram os primeiros a utilizar essa técnica na solução de problemas de otimização combinatorial. Os métodos heurísticos propostos são algoritmos do tipo melhorativo, onde a partir de uma seqüência inicial das tarefas, são examinadas uma série de outras seqüências 'vizinhas', à semelhança dos algoritmos RACS, RAES e SPIRIT. A aplicação da técnica 'Simulated Annealing' basicamente consiste em gerar uma seqüência σ' vizinha de σ (seqüência atual) e avaliar a diferença Δ entre as durações totais das programações $D(\sigma)$ e $D(\sigma')$. Se $\Delta = D(\sigma') - D(\sigma) \leq 0$, então σ' passa a ser a nova seqüência atual. Se $\Delta > 0$, então a aceitação de σ' como nova seqüência atual é associada a uma probabilidade dada por $e^{-\Delta/T}$, onde T é um parâmetro determinado empiricamente, denominado 'temperatura'. O processo se repete através de um número K de iterações, dado por:

$$K = \max\{ 3.300 \ln(n) + 7.500 \ln(m) - 18.250, 2.000\}$$

onde n = número de tarefas e m = número de máquinas.

HO & CHANG (1991) apresentaram um algoritmo melhorativo, no qual a solução inicial é obtida utilizando-se métodos já existentes, tais como PALMER, CDS, GUPTA, DANNENBRING(RA) e HUNDAL. Numa segunda etapa, Ho & Chang propõem um processo de melhoria da solução inicial baseado em relações entre os tempos de processamento das tarefas, consideradas em pares, ou seja, mais especificamente, utilizando-se as diferenças:

$$d_{ij}^k = p_{k+1,i} - p_{kj}$$

onde $i, j = 1, 2, \dots, n$ $k = 1, 2, \dots, (m - 1)$ e $i \neq j$

$p_{k+1,i}$ = tempo de processamento da tarefa J_i na máquina $(k+1)$

p_{kj} = tempo de processamento da tarefa J_j a máquina K

MOCCELLIN (1993) desenvolveu um novo método heurístico para a solução do problema de programação de operações Flow-Shop permutacional denominado FSHOPH, semelhante ao SPIRIT. A diferença básica entre eles se encontra na obtenção da solução inicial do problema.

No método FSHOPH assim como no algoritmo de WIDMER & HERTZ (1989), se faz uma analogia com o problema do caixeiro viajante. A diferença está no cálculo das distâncias, que é feita da seguinte forma:

$$UBG_{uv} = \max(0, UBX_{uv}^{m-1} + (p_{m-1,v} - p_{mu}))$$

onde

$$UBX_{uv}^{k+1} = \max(0, UBX_{uv}^k + (p_{k,v} - p_{k+1,u}))$$

$$UBX_{uv}^1 = 0.$$

para $k = 1, 2, \dots, m - 2$; $u = 1, 2, 3, \dots, n + 1$; $v = 1, 2, 3, \dots, n + 1$ e $u \neq v$

Sendo :

$UBG_{u v}$ = Limitante superior do tempo de espera, na última máquina, entre as operações das tarefas adjacentes J_u e J_v .

$p_{k u}$ = tempo de processamento da operação da tarefa J_u na máquina k ($k = 1, 2, \dots, m$)

$p_{k v}$ = tempo de processamento da operação da tarefa J_v na máquina k ($k = 1, 2, \dots, m$)

$UBX_{u v}^k = UBX_{j+1}^k$: Limitante superior do intervalo de tempo entre o término da operação da tarefa que ocupa a j -ésima posição na seqüência de tarefas e o início da operação da tarefa que ocupa a $(j+1)$ -ésima posição da seqüência, na máquina k);

No segundo passo, à semelhança do método SPIRIT, melhora-se a solução inicial, utilizando-se a técnica de 'BUSCA TABU'.

REEVES (1993) desenvolveu um método melhorativo de busca tabu. A idéia principal era fazer com que, no procedimento de busca tabu, a vizinhança de uma seqüência não fosse totalmente avaliada, mas somente um certa quantidade fixa, dada por uma relação que estabelecia o número de vizinhos a serem examinados (p) em função de ϕ (0,1; 0,2; 0,3; 0,5 e 1,0) e n (número de tarefas). Essa relação é dada por $\phi = \frac{p}{n}$.

NAGANO (1995) apresentou um trabalho utilizando a técnica de BUSCA TABU para a solução do problema de programação de operações Flow-Shop permutacional onde foram avaliados os desempenhos de dez procedimentos alternativos de Busca Tabu, incluindo-se o do método FSHOPH. Os procedimentos foram propostos a partir da combinação de diferentes estruturas de vizinhança, formas de busca na vizinhança e condições de parada. Partindo da solução inicial fornecida pelo FSHOPH, os desempenhos dos procedimentos foram avaliados em termos de qualidade da solução e esforço computacional.

REEVES(1995) elaborou um algoritmo, baseado na técnica do Algoritmo Genético, idealizado por John Holland, para resolver problemas de programação de operações flow shop permutacional. Neste trabalho, Reeves compara o algoritmo genético elaborado por ele com o Simulated Annealing é um algoritmo de busca na vizinhança simples. Este algoritmo é descrito no capítulo seguinte.

3. ALGORITMOS GENÉTICOS

3.1. INTRODUÇÃO

Os Algoritmos Genéticos são algoritmos de busca baseados nos mecanismos da teoria da evolução, que considera como fatores evolutivos as mutações, a recombinação gênica, as migrações, o tamanho da população, o isolamento reprodutivo e a seleção natural.

O que os Algoritmos Genéticos fazem é simular esses fatores em forma de sistemas artificiais como programas de computador.

Pode-se observar três tipos de processos de busca: baseado em cálculos, enumerativos e aleatórios.

- a) Baseado em cálculos: Tem a característica da necessidade da função-objetivo ser contínua e diferenciável;
- b) Enumerativos: A característica aqui é a de olhar para todos os pontos do espaço-solução, um de cada vez. É apropriado para um espaço-solução finito ou infinito discreto;
- c) Aleatórios: Possui a característica da escolha da busca ser aleatória como guia na exploração do espaço-solução.

O algoritmo genético (A.G.) é um exemplo de processo de busca aleatória. Deve-se salientar que uma escolha aleatória não, necessariamente, implica em uma busca sem direção. Pode-se citar ainda, como processos de busca aleatória, o

'Simulated Annealing' e o 'Busca Tabu'. Davis (1987) explora uma conexão entre o 'Simulated Annealing' e o 'Algoritmo Genético'.

Algumas diferenças fundamentais entre o algoritmo genético e os processos de otimização tradicional são:

- a) O A.G. trabalha com uma codificação do conjunto de parâmetros e não com eles;
- b) O A.G. busca em uma população de pontos e não pontos únicos;
- c) O A.G. usa informações da função-objetivo que não são diferenciáveis; e
- d) O A.G. usa regras de transição probabilísticas e não determinísticas.

O primeiro a desenvolver a idéia foi John Holland da Universidade de Michigan através da monografia *Adaptation in Natural and Artificial Systems*(1975). Vários textos e dissertações têm dado validade à técnica para otimização, aplicações de controle e formação de famílias em Tecnologia de Grupo(VENUGOPAL & NARENDRAN, 1992). Devido a essa validade o Algoritmo Genético tem aumentado a sua faixa de utilização.

O presente trabalho se fixará em um Algoritmo Genético simples, devido a sua alta simplicidade e eficiência/eficácia, envolvendo não mais do que copiar filas('strings') e trocar pedaços de filas. Ele é composto por três componentes, chamados operadores: Reprodução, Cruzamento('Crossover') e Mutação. Antes de se considerar estes operadores, deve-se mencionar a nomenclatura a ser usada, fazendo uma comparação com a usada na genética:

GENÉTICA	ALGORITMO GENÉTICO
Cromossomo	Fila (String)
Gene	Característica
Alelo	Valor da Característica
Locus	Posição na Fila

Figura 3.1 : Comparação das nomenclaturas usadas na genética e no Algoritmo Genético.

3.2. ALGORITMO GENÉTICO SIMPLES

Como foi mencionado, o algoritmo genético simula alguns elementos da genética e quando são usados apenas os operadores reprodução, cruzamento e mutação, ele é chamado de algoritmo genético simples. As definições destes três operadores são apresentadas a seguir.

3.2.1 - OPERADOR REPRODUÇÃO

A reprodução dos seres vivos acontece para que as espécies se perpetuem. A reprodução, chamada sexuada depende da ação combinada (ou até mesmo isolada) de células especializadas, denominadas gametas. De alguma forma essas células se encontram, portanto devem vir de algum lugar, esse lugar, no nosso caso, chamaremos de população.

Então para se iniciar a reprodução deve-se ter uma população inicial e cada indivíduo (fila), desta população, avaliado por uma função-objetivo (também chamada de função 'fitness'). Com o objetivo de aumentar a variabilidade dos

indivíduos, parte-se de duas populações denominadas subpopulações. A construção destas subpopulações observa-se adiante.

A definição que pode-se dar de reprodução é de que é um processo pelo qual filas individuais são escolhidas, da população, para reciclar e 'melhorar' essa população. Em outras palavras, a reprodução faz o papel da seleção natural da evolução, pois para cada fila há um valor associado (valor da função-objetivo). No caso de programação de operações toma-se, para serem os pais, as filas com o menor (mínimo) valor da função-objetivo, em cada subpopulação.

3.2.2 - OPERADOR CRUZAMENTO (CROSSOVER)

Após a reprodução, o operador cruzamento entra em ação. Este operador pode ser aplicado em duas etapas. Primeiro, cada membro escolhido na reprodução é 'cortado' em uma certa posição. Segundo, cada par de pais tem as partes cortadas trocadas entre si. Assim, o cruzamento tem a função de trocar informações (partes) entre um par de filas, chamadas de pais. o cruzamento pode ser feito de diversas formas, ou seja, utilizando-se diversos operadores. O de mais fácil utilização é o de um corte, sendo o mais utilizado o PMX (Partially Matched Crossover) proposto por Goldberg (1989). Um dos tipos de cruzamento é ilustrado na figura 3.2.

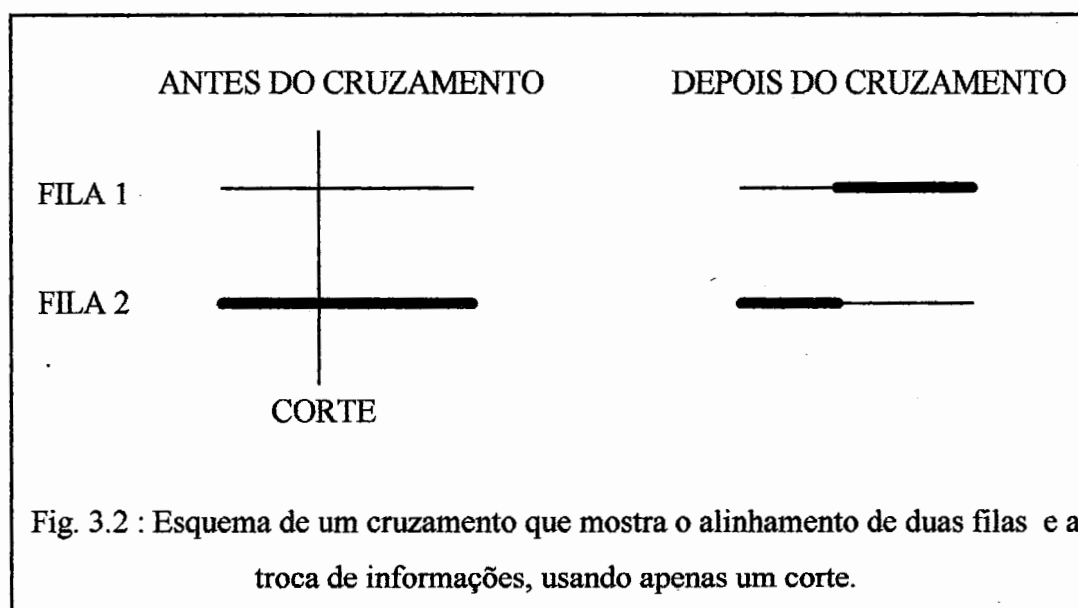


Fig. 3.2 : Esquema de um cruzamento que mostra o alinhamento de duas filas e a troca de informações, usando apenas um corte.

3.2.3 - OPERADOR MUTAÇÃO

O operador mutação tem um papel secundário no algoritmo genético. Ele é necessário, segundo Goldberg (1989), porque os operadores anteriores podem fazer com que algum material genético útil se perca, ou seja, alguma seqüência importante, e com a mutação isso pode ser reparado. Como na mutação dos seres vivos, aqui ela pode trazer resultados desejáveis ou manifestações desconcertantes. O operador mutação é ativado ocasionalmente e com uma taxa muito pequena.

3.3 - ADEQUAÇÃO DO ALGORITMO GENÉTICO AO PROBLEMA DE PROGRAMAÇÃO DE OPERAÇÕES FLOW-SHOP PERMUTACIONAL

Para se obter a máxima eficácia/eficiência do Algoritmo Genético é necessário codificar adequadamente os integrantes do problema. No caso de programação de operações flow-shop permutacional, pode-se codificar da seguinte maneira:

- a) As seqüências viáveis das tarefas são as filas (cromossomos);
- b) Cada tarefa faz o papel do gene;
- c) A posição que cada tarefa ocupa na seqüência é chamada de locus.
- d) A função-objetivo, que medirá a capacidade de sobrevivência da seqüência, poderá ser: o Makespan, Tempo médio de permanência(mean flow-time), etc.

Quando utiliza-se um certo operador, deve-se verificar se este produziu uma nova seqüência viável. Caso contrário deve-se modificá-lo a fim de ser possível a sua utilização. Um outro detalhe a ser colocado é em relação à otimização. Em programação de operações em máquinas, normalmente se quer minimizar a função-

objetivo. Desta forma, na escolha dos pais o fator(elemento) de interesse é o menor valor desta função.

REEVES(1995) se utiliza de um algoritmo genético simples, onde aparece os três operadores: reprodução(chamado por ele de mecanismo de seleção), cruzamento e mutação.

No mecanismo de seleção, ele se utiliza de um sistema simples de ranqueamento de acordo com uma distribuição de probabilidade

$$P([k]) = \frac{2k}{M(M+1)} \quad \text{onde,}$$

[k]: é o k-ésimo cromossomo da população, sendo esta colocada em ordem não-crescente dos valores dos makespan(duração total da programação).

M: é o tamanho da população

O operador de cruzamento utilizado é o de um corte, como descrito acima.

O operador mutação adotado foi chamado por ele de shift, que nada mais é do que escolher aleatoriamente um gene e permutá-lo com um dos seus dois vizinhos.

Um dos indivíduos da população inicial foi obtido pelo algoritmo NEH(1983), enquanto os (M-1) indivíduos restantes foram gerados aleatoriamente.

3.4 - OS OBJETIVOS DO TRABALHO

Um dos objetivos deste trabalho é o de comparar o desempenho dos operadores entre si, aplicados ao problema de programação de operações Flow-Shop permutacional. Ainda, analisa-se a performance do algoritmo genético utilizando-se soluções iniciais (sementes) não-aleatórias e aleatórias, observando a necessidade ou não de dar mais atenção para a construção das populações iniciais (soluções iniciais).

4. ALGORITMOS GENÉTICOS PROPOSTOS

4.1. O ALGORITMO GENÉTICO SIMPLES

No capítulo anterior mencionou-se a utilização de um algoritmo genético simples. Ele se compõe de um operador reprodução, cruzamento e mutação. Pode-se genericamente esquematizar um algoritmo genético da seguinte maneira:

Inicializar uma população $G(0)$ com N soluções viáveis

$i = 0$.

REPETIR

 Selecionar os pais de $G(i)$ observando o valor da função objetivo.

 Aplicar o operador de cruzamento nos pais.

 Aplicar o operador mutação e obter a geração $G(i + 1)$.

$i = i + 1$

ATÉ UMA CONDIÇÃO DE PARADA

Para observar como o algoritmo genético funciona, toma-se o seguinte exemplo: 'Maximizar a função $f(x) = -x^2 + 31x$ para x inteiro e $x \in [0,31]$ '.

Primeiro deve-se codificar o problema de maneira convenientes. Os valores possíveis de x (os cromossomos) colocaremos como uma seqüência de 5 algarismos binários (0,1) (os genes), então, $x = 31$ será $x = 11111$ e $x = 0$ será $x = 00000$. A função (função fitness) que ditará o fator de sobrevivência ou não do cromossomo será $f(x) = -x^2 + 31x$.

Em segundo lugar deve-se criar a população inicial e escolher os operadores reprodução, cruzamento e mutação. A população inicial pode ser criada de várias formas. Criou-se, neste exemplo, os cromossomos(população) iniciais de forma aleatória, sendo ao todo seis cromossomos. O operador reprodução, que representa a

forma de escolher os cromossomos a serem cruzados, escolherá os cromossomos(pais) aleatoriamente. O operador de cruzamento que trocará os materiais genéticos será o de um corte, ele consiste em escolher um mesmo ponto em cada cromossomo e permutar os genes entre si. No exemplo não colocou-se o operador mutação pois a sua eficácia é comprovada quando há muitas gerações. A tabela 4.1 mostra um exemplo.

No.	FILA	X	f(x)	CRUZAMENTOS
01	11011	27	108	(01) 110 11 11010
02	00011	3	84	(05) 110 10 11011
03	10101	21	210	(02) 000 1 00010
04	11100	28	84	(04) 1110 0 11101
05	11010	26	130	(03) 1010 1 10100
06	01010	10	210	(06) 0101 0 01011
01	11010	26	130	(01) 1101 0 11011
02	00010	2	58	(05) 1101 1 11010
03	10100	20	220	(02) 000 10 00001
04	11101	29	58	(04) 111 01 11110
05	11011	27	108	(03) 1 0100 11011
06	01011	11	220	(06) 0 011 00100
01	11011	27	108	(01) 11 011 11100
02	00001	1	30	(06) 00 100 00011
03	11011	27	108	(02) 0 0001 01110
04	11110	30	30	(04) 1 1110 10001
05	11010	13	234	(03) 11 011 11010
06	00100	4	108	(05) 11 010 11011
01	11100	28	84	
02	01110	14	238	
03	11010	26	130	
04	10001	17	238	
05	11011	27	108	
06	00011	3	84	

Tabela 4.1: Exemplo de utilização do conceito do algoritmo genético.

4.2. APRESENTAÇÃO DOS ALGORITMOS PROPOSTOS

Como visto no exemplo anterior, deve-se mostrar quais são os operadores a serem utilizados pelos algoritmos genéticos. Aqui apresenta-se os algoritmos utilizados na resolução dos problemas gerados aleatoriamente.

Além das hipóteses já mencionadas no capítulo 1, deve-se apresentar o cenário com que este trabalho se utilizará. Seja então:

A) A programação será em um ambiente flow-shop permutacional;

B) Usa-se um Algoritmo Genético Simples;

C) As subpopulações iniciais serão construídas de duas formas:

C.1 - Aleatoriamente: A partir de duas seqüências viáveis geradas aleatoriamente, os integrantes de cada subpopulação serão obtidos através do rotacionamento, no sentido anti-horário, de cada seqüência (com este procedimento construímos duas subpopulações);

C.2 - Utilizando boas soluções iniciais: A partir de uma seqüência gerada pelo algoritmo NEH, encontram-se os outros componentes desta subpopulação da mesma forma que acima. A outra subpopulação é gerada a partir da solução inicial do algoritmo FSHOPH (Moccellin,1993). Neste caso, a subpopulação também é gerada como acima. O tamanho da população será igual ao número de tarefas do problema.

D) Os operadores de cruzamento que serão utilizados:

D.1 - PMX (Goldberg,1989; Biegel & Davern,1990);

D.2 - MPX (Gorges, 1989);

D.3 - OX (Goldberg, 1989);

D.4 - EOX (Shang and Li, 1991);

D.5 - UM CORTE (Goldberg, 1989);

D.6 - CX (Goldberg, 1989);

D.7 - LOX (Falkenauer & Bouffouix, 1991).

E) Com o objetivo de aumentar a variabilidade dos cromossomos, o operador mutação, que apenas trocará de lugar duas tarefas aleatoriamente, terá uma frequência de 100% e será colocado em ação no início e no fim do algoritmo, ou seja, em todas as gerações haverá mutação.

F) O número de gerações será em princípio de 100, podendo ser mudado dependendo do desempenho do algoritmo.

G) A função-objetivo usada para avaliar os cromossomos será o Makespan.

O único elemento a ser mudado de um algoritmo para outro é o operador de cruzamento, portanto os algoritmos apresentam-se com os nomes de seus operadores de cruzamento.

Os operadores de cruzamento são descritos a seguir.

4.2.1 - OPERADOR PMX(Partially Matched Crossover)

PASSO 01 : Escolha um intervalo para ser trocado.

PASSO 02 : Crie um mapa do intervalo selecionado;

PASSO 03 : Permute os dois intervalos;

PASSO 04 : Use o mapa para alterar a nova solução para torná-la viável.

EXEMPLO:

$\leq n/2$

A: 9 8 4 5 6 7 1 3 2 10

B: 8 7 1 2 3 10 9 5 4 6

PASSO 01 : Escolha 4 a 6 ;

PASSO 02 :	A	B
	5	2
	6	3
	7	10

PASSO 03 :

A': 9 8 4 2 3 10 1 3 2 10

B': 8 7 1 5 6 7 9 5 4 6

PASSO 04 :

A': 9 8 4 2 3 10 1 6 5 7

B': 8 10 1 5 6 7 9 2 4 3

4.2.2 - OPERADOR MPX(MAXIMAL-PREVENTIVE) :

PASSO 01 : Escolha uma subsequência do pai A; coloque-a na solução filho, o mais a direita possível;

PASSO 02 : As posições não preenchidas o são com tarefas do pai B, da seguinte forma:

PASSO 2.1: As tarefas que estão nas mesmas posições, da solução filho, já preenchidas, são colocadas, na ordem inversa, logo antes da subsequência colocada, evitar conflito;

PASSO 2.2 : As posições ainda não preenchidas, o são, colocando as tarefas que sobraram, nas suas posições relativas.

EXEMPLO:

A: 1 2 3 4 5 6 7 8 9 10

B: 8 4 9 3 7 5 1 10 2 6

C: 8 9 7 2 10 1 3 4 5 6

4.2.3 - OPERADOR ORDER CROSSOVER(OX):

PASSO 01 : Escolha uma subsequência do pai A; coloque-a na solução filho na mesma posição que na original;

PASSO 02 : As demais posições serão ocupadas pelas tarefas do pai B, com um movimento correção das tarefas(eliminando conflito) até encaixar-se na solução filho.

EXEMPLO :

A : 1 2 3 4 5 6 7 8 9

B : 6 9 5 1 4 8 2 3 7

PASSO 01 : Escolha de 4 a 7;

C : _ _ _ 4 5 6 7 _ _ _

PASSO 02 :

B : 6 9 5 | 1 4 8 2 | 3 7

B : 6 9 5 | 1 8 2 4 | 3 7

B : 6 9 1 | 8 2 4 5 | 3 7

B : 9 1 8 | 2 4 5 6 | 3 7

B : 1 8 2 4 5 6 7 | 3 9

C : 1 8 2 4 5 6 7 3 9

4.2.4 - OPERADOR ENHANCE ORDER-CROSSOVER(EOX)

O EOX trabalha da mesma forma que o OX. A diferença é apenas, que após ter escolhido uma subsequência do Pai A, o Pai B é rotacionado até que as tarefas que estiverem na última posição da subsequência do Pai A coincidirem.

EXEMPLO :

A : 2 5 3 7 10 6 11 12 1 9 4 8

B : 1 2 3 4 5 6 7 8 9 10 11 12

B rotacionado : 5 6 7 8 9 10 11 12 1 2 3 4

4.2.5 - OPERADOR DE UM CORTE:

PASSO 01 : Escolher um ponto para se partir, cada pai, em duas subsequências;

PASSO 02 : A solução filho é gerada através da união de duas subsequências, uma de cada pai, na sua posição absoluta;

PASSO 03 : Para se evitar conflito, troca-se tarefas duplicadas por tarefas que faltam para completar a solução filho.

EXEMPLO:

A:	1	2	3	4	5	6	7	8	9	10
B:	4	9	5	2	3	8	10	1	7	6

A':	1	2	3	4	5	6	7	1	7	6
B':	4	9	5	2	3	8	10	8	9	10

A': 8 2 3 4 5 10 9 1 7 6

B': 4 7 5 2 3 1 6 8 9 10

4.2.6 - OPERADOR CYCLE Crossover (CX) :

PASSO 01 : Escolha a tarefa na 1a. posição do Pai A; essa tarefa será a 1a. tarefa da solução filho;

PASSO 02 : Toma-se a tarefa, no Pai B, que se encontra na 1a. posição e verifica-se a posição que esta tarefa se encontra no Pai A e coloca-se nesta posição, na solução filho, esta tarefa ;

PASSO 03 : Toma-se a tarefa, no Pai B, que se encontra na posição da tarefa escolhida no passo 02 e verifica-se a posição que esta tarefa se encontra no Pai A e coloca-se nesta posição, na solução filho, esta tarefa ;

PASSO 04 : Toma-se a tarefa, no Pai B, que se encontra na posição da tarefa escolhida no passo anterior e verifica-se a posição que esta tarefa se encontra no Pai A e coloca-se nesta posição, na solução filho, esta tarefa ;

PASSO 05 : Repetir o passo 04 até completar o círculo ;

PASSO 06 : As posições ainda não preenchidas na solução filho, são preenchidas com as tarefas do Pai B em suas posições absolutas .

EXEMPLO:

A: 1 2 3 4 5 6 7 8 9 10

B: 6 9 5 7 3 4 1 2 10 8

C: 1 9 5 4 3 6 7 2 10 8

4.2.7 - OPERADOR LINEAR ORDER Crossover (LOX)

PASSO 01 : Escolha uma subsequência do pai A; coloque-a na solução filho na mesma posição que na original;

PASSO 02 : As demais posições serão ocupadas pelas tarefas do pai B, com um movimento correção das tarefas(eliminando conflito) até encaixar-se na solução filho.

OBSERVAÇÃO : A única diferença para o OX é a de que no passo 02 as tarefas permanecem em suas posições relativas.

EXEMPLO :

A: 2 5 3 7 | 10 6 11 12 | 1 9 4 8
B: 1 2 3 4 | 5 6 7 8 | 9 10 11 12

C: 1 2 3 4 | 10 6 11 12 | 5 7 8 9

5. EXPERIMENTAÇÃO COMPUTACIONAL E ANÁLISE DOS RESULTADOS

5.1. EXPERIMENTAÇÃO COMPUTACIONAL

Na experimentação computacional foram resolvidos 420 problemas, subdivididos em três grupos. O primeiro grupo de problemas, considerados de pequeno porte, totalizou 60, divididos em 6 classes de acordo com o número de tarefas 'n' e o número de máquinas 'm', para $n = 10$ tarefas e $m = 4, 7, 10, 15, 20$ ou 25 máquinas.

O segundo grupo, de problemas considerados de médio porte, foi constituído de 12 classes de problemas com $n = 30$ ou 50 e $m = 4, 7, 10, 15, 20$ ou 25, compondo 120 problemas ao todo.

O terceiro e último grupo, para problemas considerados de grande porte, tem-se 24 classes, sendo $n = 70, 90, 100$ ou 110 e $m = 4, 7, 10, 15, 20$ ou 25.

Os problemas em questão foram gerados aleatoriamente, sendo os tempos de processamento das operações, números inteiros uniformemente distribuídos no intervalo $[0,100]$.

O equipamento utilizado foi um microcomputador 486 DX2, com 8 Mb de memória RAM, disco rígido de 320 Mb. Os algoritmos foram codificados em linguagem Turbo Pascal para Windows.

Para a solução mais otimizada dos problemas acima, desenvolveu-se um software chamado 'Controlador de Programação de Operações Flow Shop Permutacional'.

As figuras 5.1.1 a 5.1.5 mostram as estruturas principais do software.

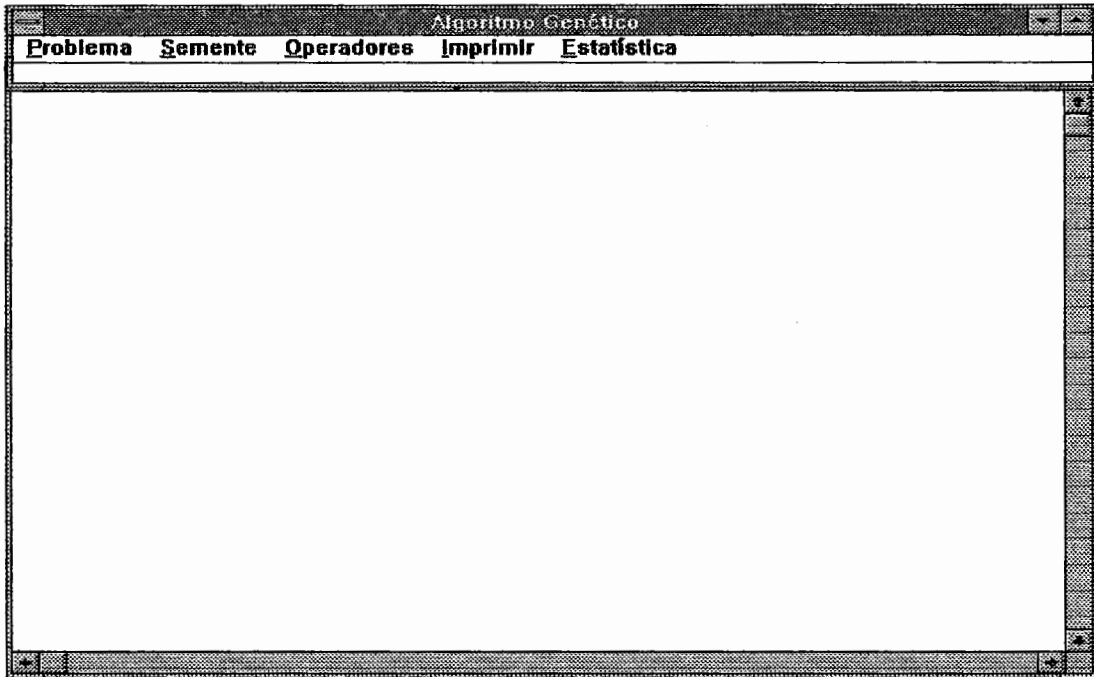


Figura 5.1.1: Tela principal do software.

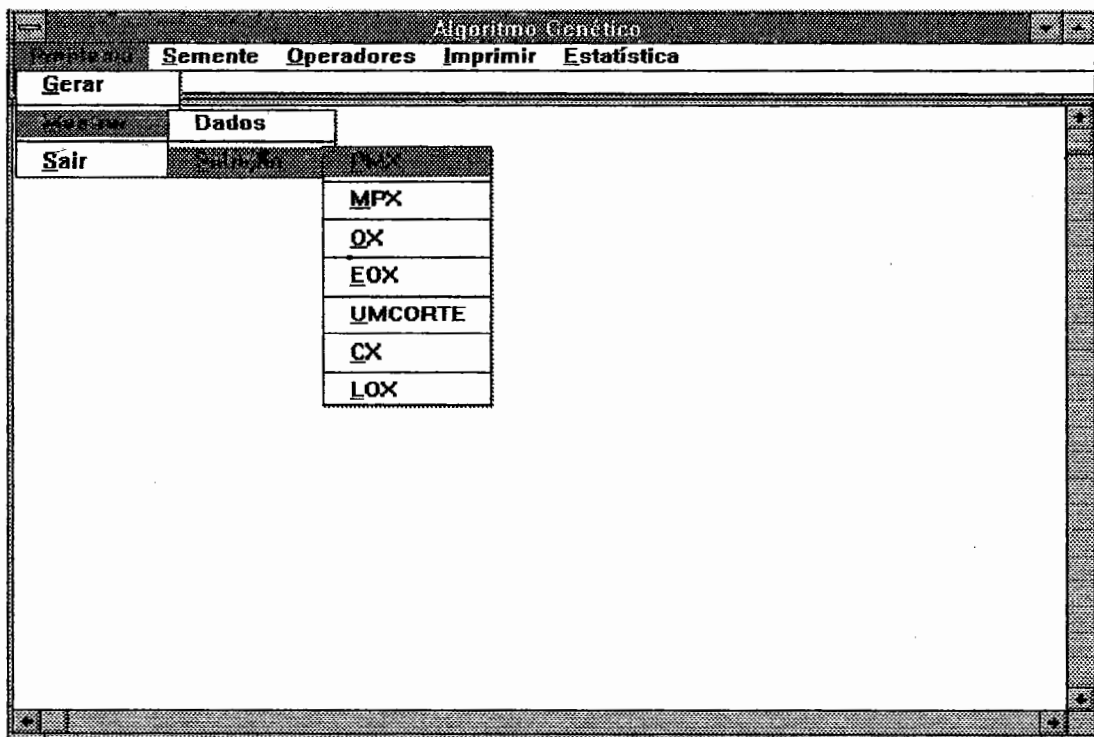


Figura 5.1.2: Ramificações da opção PROBLEMA.

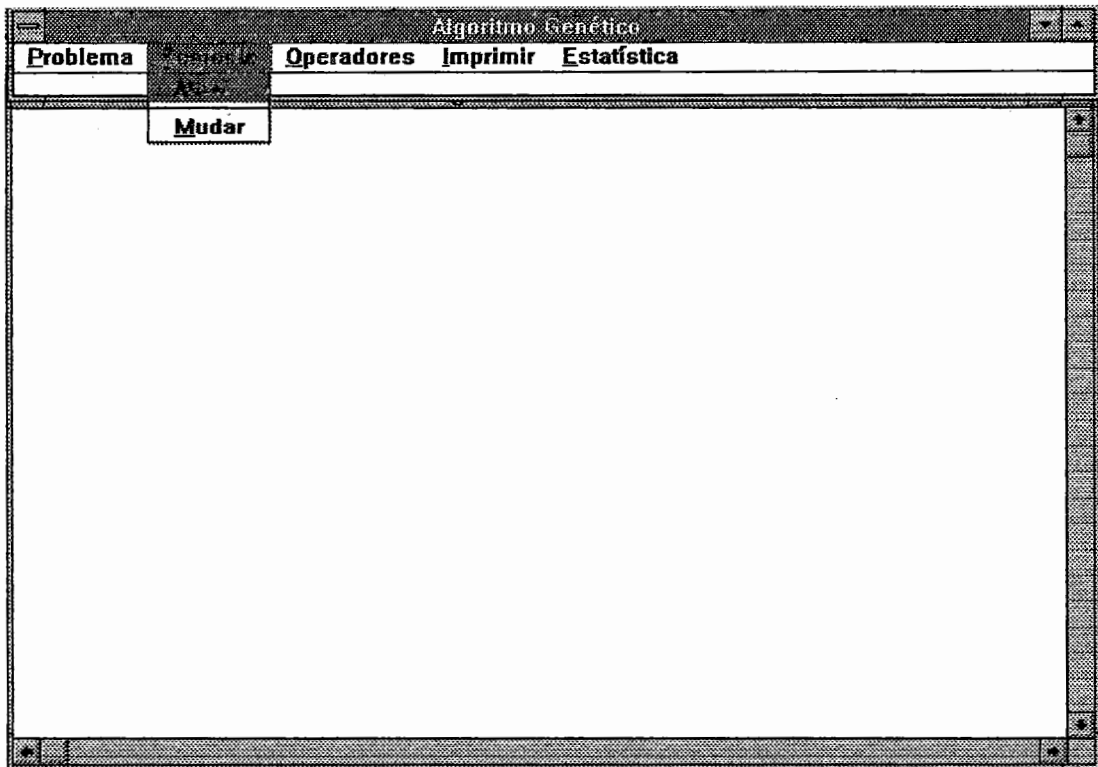


Figura 5.1.3: Ramificações da opção SEMENTE.

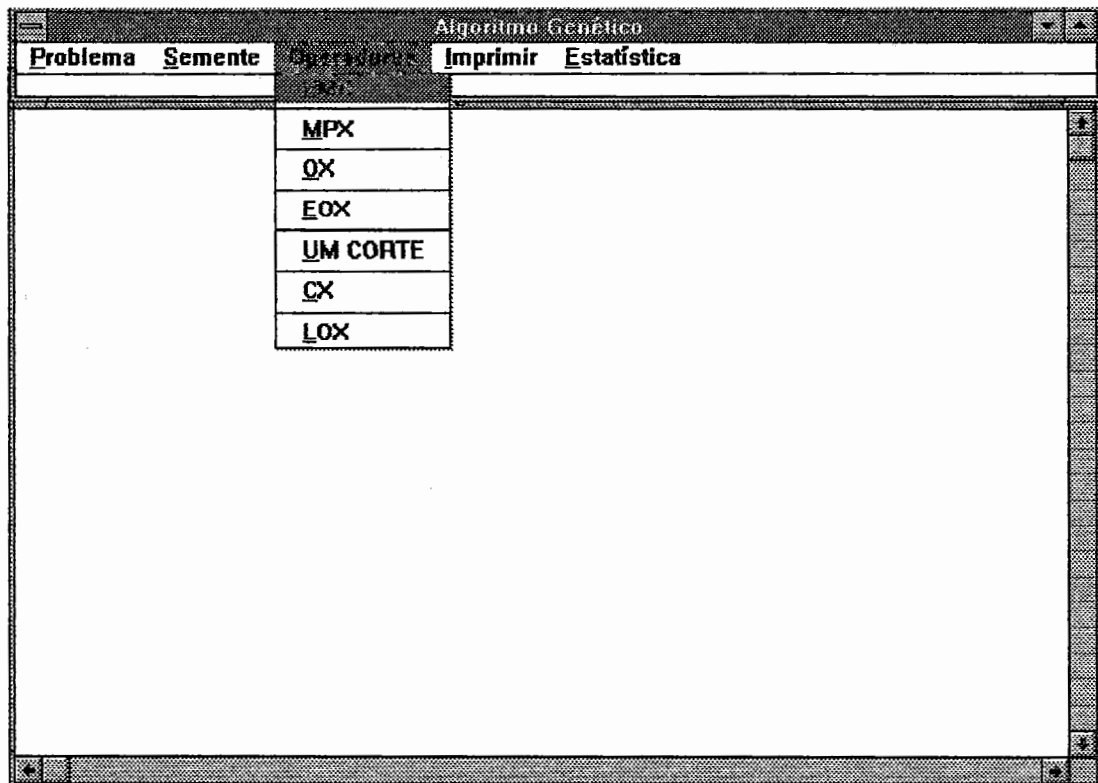


Figura 5.1.4: Ramificações da opção OPERADORES.

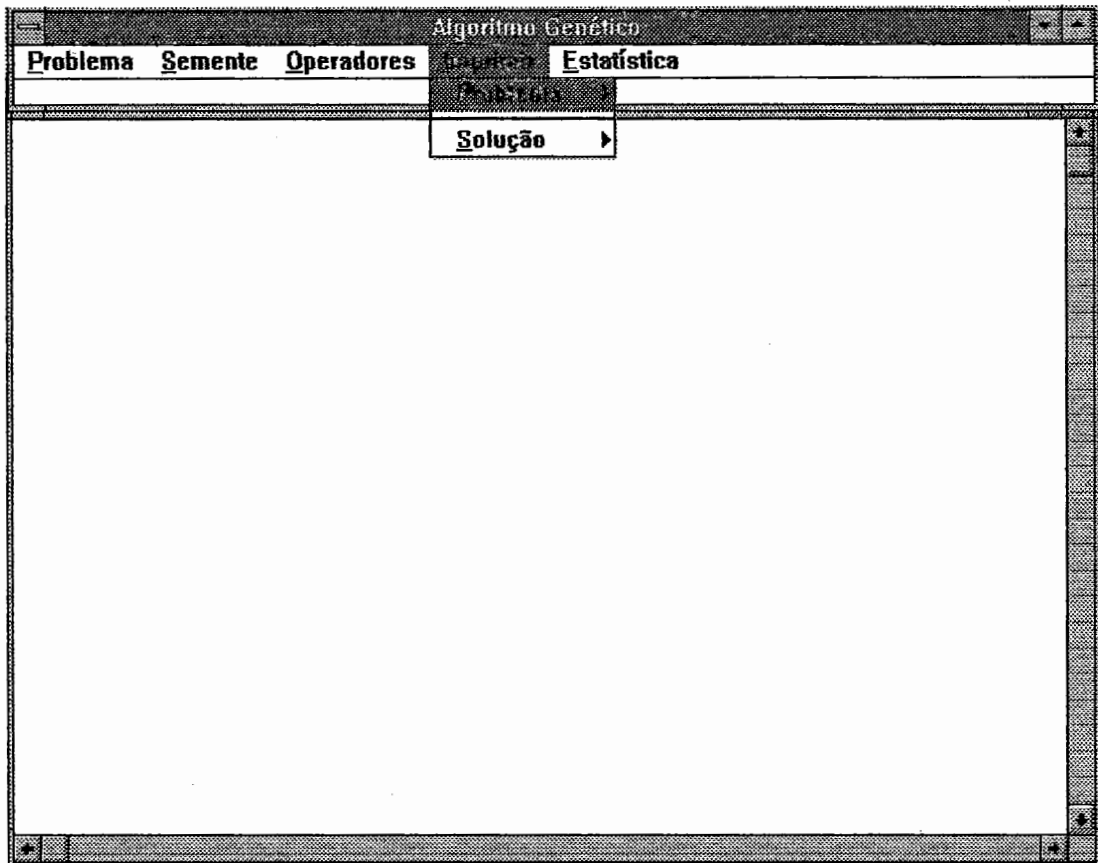


Figura 5.1.5: Ramificações da opção IMPRIMIR.

Para comparar os diversos operadores de cruzamento usou-se a Porcentagem de Sucesso definido como o número de vezes que o algoritmo obteve o menor makespan, isoladamente ou não dividido pelo número total de problemas resolvidos.

$$\%SUCESSO = \frac{\text{No. de vezes que o algoritmo obteve o melhor resultado}}{\text{Total de problemas resolvidos}} \cdot 100\%$$

Para uma outra forma de análise, calculou-se um desvio relativo(D.R.), que tem como referência um Limitante inferior (L.B.), o qual é descrito a seguir.

$$\text{D.R.} = \frac{\text{Sol. obtida} - \text{L.B.}}{\text{L.B.}} \cdot 100\% , \text{ onde}$$

$$L.B. = \text{MAX} \left\{ \begin{array}{l} \sum_{j=1}^n t_{1j} + \text{MIN}_{1 \leq j \leq n} \sum_{i=2}^m t_{ij} \\ \text{MIN}_{1 \leq j \leq n} \sum_{i=1}^{k-1} t_{ij} + \sum_{j=1}^n t_{kj} + \text{MIN}_{1 \leq j \leq n} \sum_{i=k+1}^m t_{ij} \text{ para } 2 \leq k \leq m-1 \\ \text{MIN}_{1 \leq j \leq n} \sum_{i=1}^{m-1} t_{ij} + \sum_{j=1}^n t_{mj} \end{array} \right.$$

n = Número de tarefas e m = Número de máquinas.

t_{ij} = Tempo de processamento da tarefa i na máquina j

5.2. RESULTADOS OBTIDOS

Os resultados obtidos na experimentação, são apresentados nas tabelas 5.2.1 a 5.2.34. Os números entre parêntesis indicam tempos médios de computação, em segundos. Nas tabelas de resultados globais (Tabelas 5.2.3, 5.2.8, 5.2.17, 5.2.20, 5.2.25 e 5.2.34), a primeira linha indica a porcentagem de sucesso e a segunda linha apresenta a média dos desvios relativos.

5.2.1. SEMENTES ALEATÓRIAS

As tabelas 5.2.1 a 5.2.17 apresentam os resultados obtidos pelos algoritmos utilizando-se sementes iniciais aleatórias.

Tabela 5.2.1: Porcentagem de sucesso, para problemas com $n = 10$ tarefas e sementes aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X10	10 (0.75)	20 (0.76)	40 (0.76)	20 (0.76)	40 (0.74)	40 (0.71)	20 (0.76)
07X10	0 (1.30)	20 (1.29)	30 (1.27)	10 (1.31)	40 (1.25)	20 (1.23)	10 (1.28)
10X10	50 (1.80)	20 (1.80)	0 (1.80)	10 (1.82)	10 (1.77)	0 (1.74)	10 (1.80)
15X10	20 (2.69)	10 (2.66)	30 (2.66)	30 (2.70)	30 (2.66)	10 (2.63)	30 (2.69)
20X10	20 (3.58)	30 (3.62)	10 (3.59)	20 (3.61)	30 (3.57)	0 (3.54)	20 (3.57)
25X10	20 (4.46)	0 (4.48)	20 (4.49)	30 (4.49)	20 (4.46)	10 (4.43)	20 (4.48)

Tabela 5.2.2: Média dos desvios relativos para problemas com $n = 10$ tarefas e sementes aleatórias

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X10	6.11	5.64	5.59	6.03	4.97	5.59	6.20
07X10	15.31	15.13	15.40	15.32	14.30	15.01	15.93
10X10	18.91	18.50	19.69	18.28	19.23	19.03	18.68
15X10	24.56	24.56	24.24	24.33	24.06	24.34	23.77
20X10	26.37	26.17	26.50	26.39	26.60	27.24	26.21
25X10	23.73	23.67	23.62	23.37	23.59	23.99	24.40

Tabela 5.2.3: Resultados para problemas de pequeno porte, utilizando sementes aleatórias.

Número de tarefas	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
10	20.00	16.67	21.67	20	28.33	13.33	18.33
	19.16	18.95	19.17	18.95	18.79	19.2	19.2
	(2.46)	(2.47)	(2.44)	(2.48)	(2.44)	(2.4)	(2.46)

Tabela 5.2.4: Porcentagem de sucesso para problemas com n = 30 tarefas e sementes aleatórias

Classe de problema m x n	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X30	10 (5.35)	0 (5.38)	10 (5.36)	40 (5.56)	20 (5.36)	10 (5.31)	40 (5.34)
07X30	0 (9.95)	10 (9.99)	10 (9.96)	20 (10.16)	50 (9.97)	0 (9.89)	20 (9.97)
10X30	40 (14.67)	10 (14.71)	30 (14.68)	10 (14.88)	0 (14.68)	0 (14.64)	10 (14.71)
15X30	0 (22.47)	0 (22.51)	30 (22.48)	30 (22.69)	20 (22.46)	20 (22.43)	20 (22.49)
20X30	20 (30.52)	0 (30.54)	20 (30.53)	20 (30.73)	10 (30.53)	10 (30.47)	40 (30.55)
25X30	10 (38.49)	0 (38.52)	20 (38.5)	10 (38.7)	10 (38.5)	40 (37.45)	10 (38.5)

Tabela 5.2.5: Porcentagem de sucesso para problemas com $n = 50$ tarefas e sementes aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X50	30 (14.53)	40 (14.57)	30 (14.54)	20 (15.13)	20 (14.59)	20 (14.46)	40 (14.56)
07X50	20 (27.25)	0 (27.35)	10 (27.27)	10 (27.84)	10 (27.32)	20 (27.19)	50 (27.73)
10X50	10 (40.27)	10 (40.34)	20 (40.30)	0 (40.87)	10 (40.34)	50 (40.24)	0 (40.30)
15X50	20 (61.99)	0 (62.06)	10 (62.02)	10 (62.58)	20 (62.06)	20 (61.96)	20 (62.04)
20X50	0 (84.60)	0 (84.67)	10 (84.64)	40 (85.18)	0 (84.67)	20 (84.55)	30 (84.64)
25X50	20 (106.82)	0 (106.88)	20 (106.84)	10 (107.34)	20 (106.85)	30 (106.74)	10 (107.07)

Tabela 5.2.6: Médias dos desvios relativos para problemas com $n = 30$ tarefas e sementes aleatórias

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X30	1.77	1.49	1.98	1.66	1.96	2.08	1.67
07X30	6.63	6.74	6.91	7.50	6.44	6.85	6.31
10X30	10.05	10.84	9.90	11.10	11.29	11.37	10.63
15X30	17.73	17.53	16.36	17.02	16.43	17.56	16.57
20X30	23.05	23.56	22.67	22.50	22.44	23.81	22.46
25X30	27.98	28.98	28.39	28.68	27.94	27.29	28.44

Tabela 5.2.7: Médias dos desvios relativos para problemas com $n = 50$ tarefas e sementes aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X50	0.49	0.42	0.62	0.64	0.62	0.70	0.53
07X50	3.70	3.69	3.78	3.78	3.47	3.65	3.33
10X50	9.47	10.11	8.68	9.49	9.51	8.85	10.19
15X50	15.27	16.23	15.53	16.10	15.84	15.94	15.33
20X50	20.95	21.35	20.64	20.02	20.83	20.21	20.14
25X50	25.54	26.00	25.12	25.42	25.58	25.90	25.89

Tabela 5.2.8: Resultados globais para problemas de médio porte e sementes aleatórias

Número de tarefas	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
30	13.33	3.33	20	21.67	18.33	13.33	23.33
	14.53	14.86	14.37	14.74	14.42	14.83	14.34
	(20.69)	(20.72)	(20.71)	(20.91)	(20.72)	(20.64)	(20.7)
50	18.33	8.33	13.33	15	11.67	25	28.33
	12.73	13.51	12.87	12.81	12.79	12.75	12.62
	(57.69)	(57.76)	(57.72)	(58.3)	(57.76)	(57.65)	(57.75)
valores globais	15.83	5.83	16.67	18.34	15	19.17	25.83
	13.63	14.19	13.62	13.78	13.61	13.79	13.48

Tabela 5.2.9: Porcentagem de sucesso para problemas com $n = 70$ tarefas e sementes aleatória.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X70	30 (28.16)	10 (28.29)	0 (28.21)	20 (29.34)	20 (28.31)	30 (28.13)	10 (28.25)
07X70	10 (53.05)	0 (53.16)	10 (53.10)	40 (54.21)	10 (53.20)	10 (53.01)	20 (53.14)
10X70	20 (78.67)	20 (78.82)	10 (78.75)	10 (79.80)	10 (78.82)	20 (78.63)	10 (78.75)
15X70	10 (121.2)	0 (121.4)	10 (121.3)	30 (122.3)	20 (121.4)	0 (121.2)	30 (121.3)
20X70	20 (165.8)	20 (165.9)	10 (165.9)	10 (166.8)	30 (165.8)	10 (165.7)	0 (165.8)
25X70	30 (209.6)	10 (209.8)	30 (209.7)	10 (210.7)	10 (209.7)	0 (209.6)	10 (209.7)

Tabela 5.2.10: Porcentagem de sucesso para problemas com $n = 90$ tarefas e sementes aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X90	20 (46.41)	20 (46.61)	30 (46.48)	30 (48.34)	10 (46.64)	10 (46.36)	10 (46.57)
07X90	20 (87.34)	20 (87.87)	20 (87.40)	0 (89.69)	20 (87.58)	0 (87.28)	20 (87.49)
10X90	20 (129.8)	10 (130)	0 (129.9)	0 (131.6)	30 (130.1)	30 (129.7)	20 (129.9)
15X90	10 (200.9)	0 (201.1)	10 (201.0)	20 (202.7)	30 (201.1)	30 (200.9)	10 (201.0)
20X90	20 (274.6)	0 (274.8)	0 (274.8)	30 (276.3)	10 (274.9)	10 (274.7)	30 (274.8)
25X90	20 (347.7)	0 (347.8)	20 (347.7)	10 (349.4)	30 (347.9)	20 (347.6)	0 (347.8)

Tabela No.5.2.11: Porcentagem de sucesso para problemas com $n = 100$ tarefas e sementes aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X100	40 (57.3)	30 (57.5)	10 (57.4)	30 (59.7)	20 (57.6)	30 (57.2)	0 (57.4)
07X100	30 (107.9)	20 (108.2)	20 (108.1)	10 (110.3)	10 (108.3)	10 (107.9)	0 (108.1)
10X100	10 (161.3)	10 (161.5)	10 (161.3)	0 (163.6)	30 (161.6)	10 (161.3)	30 (161.4)
15X100	40 (238.2)	0 (238.4)	10 (238.3)	0 (240.3)	20 (238.5)	20 (238.2)	10 (238.4)
20X100	10 (339.4)	10 (339.6)	0 (339.5)	20 (341.7)	40 (339.6)	20 (339.4)	0 (339.5)
25X100	10 (427.0)	10 (427.3)	10 (427.2)	10 (429.3)	40 (427.3)	20 (427.0)	0 (427.2)

Tabela 5.2.12: Porcentagem de sucesso para problemas com $n = 110$ tarefas e sementes aleatórias

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X110	40 (69.5)	20 (69.8)	0 (69.6)	20 (72.4)	30 (69.9)	10 (69.7)	10 (69.7)
07X110	10 (130.5)	0 (130.7)	0 (130.6)	20 (133.3)	20 (130.8)	40 (130.4)	10 (130.6)
10X110	20 (194.1)	0 (194.4)	20 (194.3)	30 (196.9)	0 (194.4)	10 (194.1)	20 (194.3)
15X110	10 (300.5)	10 (300.7)	30 (300.6)	0 (303.5)	10 (300.8)	30 (300.5)	10 (300.9)
20X110	0 (410.8)	10 (410.9)	10 (410.9)	10 (413.4)	40 (411.1)	10 (410.8)	20 (410.9)
25X110	20 (519.1)	10 (519.5)	20 (519.3)	10 (522.3)	10 (520.0)	20 (519.7)	10 (519.9)

Tabela 5.2.13: Média dos desvios relativos para problemas com $n = 70$ tarefas e sementes aleatórias

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X70	0.85	1.16	1.10	0.87	1.04	0.84	0.92
07X70	4.06	4.02	4.35	3.61	4.41	4.50	4.18
10X70	7.63	7.44	7.75	8.00	7.52	7.11	7.59
15X70	13.58	14.30	13.60	13.21	13.29	13.90	13.77
20X70	20.74	21.38	20.43	20.94	20.50	21.06	21.29
25X70	21.30	22.66	21.63	21.86	21.88	21.63	21.92

Tabela 5.2.14: Média dos desvios relativos para problemas com $n = 90$ tarefas e sementes aleatórias

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X90	1.07	0.87	1.22	1.07	1.36	1.41	1.18
07X90	2.34	2.74	2.47	2.52	2.48	2.95	2.75
10X90	5.59	5.81	6.12	6.59	5.81	5.94	5.64
15X90	10.84	11.64	11.29	11.31	10.91	11.52	11.58
20X90	14.98	15.44	14.79	15.02	15.22	15.06	14.85
25X90	18.94	20.96	19.26	19.81	19.36	19.35	19.81

Tabela 5.2.15: Média dos desvios relativos para problemas com $n = 100$ tarefas e sementes aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X100	0.64	0.58	0.91	0.73	0.87	0.89	0.91
07X100	2.71	2.53	2.96	3.02	2.72	2.98	2.66
10X100	5.89	5.90	5.34	5.98	5.32	5.61	5.52
15X100	10.71	11.66	10.95	10.95	10.75	11.09	11.17
20X100	14.69	15.65	15.21	15.13	14.40	15.04	15.04
25X100	19.34	20.28	19.51	19.15	18.36	19.11	19.85

Tabela 5.2.16: Média dos desvios relativos para problemas com $n = 110$ tarefas e sementes aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X110	0.70	0.52	0.81	0.62	0.63	0.67	0.90
07X110	3.13	2.95	3.22	2.83	2.72	2.85	3.00
10X110	5.29	5.15	5.34	4.99	5.36	5.13	5.17
15X110	11.40	11.38	11.20	11.32	11.29	10.88	10.88
20X110	14.99	15.40	14.85	15.10	14.69	15.00	14.87
25X110	17.90	18.83	18.52	17.98	18.18	18.35	18.30

Tabela 5.2.17: Resultados globais para problemas de grande porte e sementes aleatórias.

Número de tarefas	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
70	20	10	11.67	20	16.67	11.67	13.33
	11.36	11.83	11.48	11.42	11.44	11.51	11.61
	(114.14)	(114.26)	(114.18)	(115.33)	(114.27)	(114.08)	(114.22)
90	18.33	8.33	13.33	15	21.67	16.67	15
	8.96	9.58	9.19	9.39	9.19	9.37	9.3
	(190.88)	(191.06)	(190.95)	(192.84)	(191.11)	(190.81)	(191.02)
100	23.33	13.33	10	11.67	26.67	18.33	6.67
	9	9.43	9.15	9.16	8.74	9.12	9.19
	(236.98)	(237.21)	(237.09)	(239.47)	(237.31)	(236.95)	(237.21)
110	16.67	8.33	13.33	15	18.33	20	13.33
	8.9	9.04	8.99	8.81	8.81	8.81	8.85
	(288.34)	(288.64)	(288.49)	(291.23)	(288.66)	(288.23)	(288.5)
valores globais	19.58	10	12.08	15.42	20.84	12.5	12.08
	9.56	9.97	9.7	9.7	9.55	9.7	9.74

5.2.2. SEMENTES NÃO-ALEATÓRIAS

As tabelas 5.2.18 a 5.2.34 apresentam os resultados obtidos pelos algoritmos utilizando-se sementes iniciais não-aleatórias, obtidas pelos métodos NEH e FShopH.

Tabela 5.2.18: Porcentagem de sucesso para problemas com $n = 10$ tarefas e sementes não-aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X10	70 (0.77)	70 (0.79)	50 (0.76)	60 (0.785)	50 (0.768)	80 (0.732)	60 (0.765)
07X10	40 (1.30)	50 (1.29)	60 (1.28)	70 (1.30)	50 (1.29)	60 (1.24)	60 (1.30)
10X10	90 (1.83)	80 (1.85)	90 (1.81)	80 (1.84)	90 (1.80)	90 (1.75)	80 (1.81)
15X10	50 (2.72)	50 (2.72)	50 (2.70)	70 (2.74)	60 (2.68)	70 (2.65)	40 (2.72)
20X10	90 (3.61)	80 (3.63)	70 (3.60)	80 (3.65)	70 (3.61)	60 (3.57)	100 (3.62)
25X10	40 (4.5)	70 (4.5)	50 (4.5)	30 (4.5)	50 (4.5)	30 (4.5)	40 (4.5)

Tabela 5.2.19: Média dos desvios relativos para problemas com $n = 10$ tarefas e sementes não-aleatórias.

Classe de problema $m \times m$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X10	4.57	4.64	4.89	4.74	4.67	4.43	4.66
07X10	12.80	12.84	12.36	12.82	12.86	12.86	12.59
10X10	16.34	16.63	16.25	16.28	16.35	16.25	16.53
15X10	22.69	22.83	22.74	22.83	22.84	22.76	22.94
20X10	25.37	25.50	25.66	25.54	25.59	25.65	25.36
25X10	23.47	22.92	23.27	23.68	23.37	23.47	23.76

Tabela 5.2.20: Resultados globais para problemas de pequeno porte e sementes não-aleatórias.

Número de tarefas	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
10	63.33 17.54 (2.43)	66.67 17.56 (2.43)	61.67 17.53 (2.43)	65 17.65 (2.45)	61.67 17.61 (2.41)	65 17.57 (2.38)	63.33 17.64 (2.43)

Tabela 5.2.21: Porcentagem de sucesso para problemas com $n = 30$ tarefas e sementes não-aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X30	70 (5.5)	80 (5.5)	70 (5.5)	60 (5.7)	70 (5.5)	80 (5.5)	70 (5.5)
07X30	30 (10.2)	20 (10.2)	20 (10.2)	30 (10.4)	20 (10.2)	30 (10.1)	40 (10.2)
10X30	10 (15.0)	0 (15.0)	0 (15.0)	20 (15.2)	40 (15.0)	30 (14.9)	10 (15.0)
15X30	0 (22.9)	0 (23.0)	30 (23.0)	10 (23.2)	30 (23.1)	20 (22.9)	40 (23.0)
20X30	10 (31.2)	0 (31.2)	10 (31.2)	20 (31.4)	40 (31.2)	40 (31.1)	0 (31.2)
25X30	30 (39.3)	10 (39.3)	20 (39.3)	0 (39.5)	20 (39.3)	30 (39.3)	20 (39.3)

Tabela 5.2.22: Porcentagem de sucesso para problemas com $n = 50$ tarefas e sementes não-aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X50	90 (15.1)	80 (15.1)	90 (15.1)	80 (15.7)	80 (15.2)	90 (15.0)	80 (15.2)
07X50	30 (28.1)	40 (28.2)	40 (28.1)	30 (28.7)	40 (28.2)	20 (28.1)	40 (28.2)
10X50	50 (41.6)	20 (41.6)	30 (41.6)	30 (42.2)	40 (41.6)	40 (41.5)	20 (41.6)
15X50	10 (63.9)	10 (64.0)	10 (64.0)	20 (64.6)	20 (64.0)	20 (63.9)	10 (64.0)
20X50	30 (87.1)	0 (87.2)	20 (87.2)	20 (87.7)	30 (87.2)	0 (87.1)	20 (87.2)
25X50	50 (110.3)	20 (110.3)	20 (110.3)	10 (110.9)	40 (110.3)	10 (110.2)	30 (110.3)

Tabela 5.2.23: Média dos desvios relativos para problemas com $n = 30$ tarefas e sementes não-aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X30	0.49	0.38	0.37	0.44	0.39	0.46	0.37
07X30	4.28	4.78	4.58	4.58	4.75	4.37	4.30
10X30	8.36	8.52	8.14	8.23	8.10	7.90	8.59
15X30	13.59	13.90	13.43	13.56	13.50	13.42	13.26
20X30	19.92	20.16	19.81	19.75	19.65	19.77	19.88
25X30	24.55	24.92	24.69	24.89	24.53	24.77	24.72

Tabela 5.2.24: Média dos desvios relativos para problemas com $n = 50$ tarefas e sementes não-aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X50	0.07	0.13	0.07	0.09	0.13	0.09	0.11
07X50	1.81	1.92	1.80	1.80	1.77	1.94	1.84
10X50	5.54	5.69	5.56	5.59	5.58	5.56	5.64
15X50	11.74	11.89	11.76	11.55	11.57	11.57	11.71
20X50	15.46	15.74	15.33	15.33	15.47	15.47	15.45
25X50	20.43	20.76	20.55	20.63	20.51	20.62	20.47

Tabela 5.2.25: Resultados globais para problemas de médio porte e sementes não-aleatórias.

Número de tarefas	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
30	25	18.33	25	23.33	36.67	38.33	30
	11.86 (20.24)	12.11 (20.28)	11.84 (20.25)	11.91 (20.45)	11.82 (20.25)	11.78 (20.2)	11.85 (20.26)
50	43.33	28.33	35	31.67	41.67	30	33.33
	9.18 (50.63)	9.35 (50.7)	9.18 (50.66)	9.17 (51.21)	9.17 (50.7)	9.21 (50.59)	9.2 (50.74)
valores globais	34.17 10.52	23.33 10.73	30 10.51	27.5 10.54	39.17 10.5	34.17 10.5	31.67 10.53

Tabela 5.2.26: Porcentagem de sucesso para problemas com $n = 70$ tarefas e sementes não-aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X70	70 (29.5)	80 (29.6)	60 (29.6)	80 (30.8)	70 (29.7)	60 (29.5)	70 (29.6)
07X70	20 (55.4)	30 (55.5)	30 (55.4)	30 (56.6)	20 (55.5)	50 (55.3)	40 (55.5)
10X70	40 (82.1)	20 (82.2)	20 (82.1)	50 (83.3)	50 (82.2)	40 (82.0)	40 (82.2)
15X70	40 (126.4)	20 (126.6)	20 (126.5)	30 (127.6)	20 (126.6)	20 (126.4)	10 (126.5)
20X70	20 (172.7)	0 (172.8)	20 (172.7)	20 (173.9)	20 (172.8)	0 (172.6)	30 (172.7)
25X70	30 (218.7)	0 (218.8)	10 (218.7)	20 (219.9)	30 (218.8)	30 (218.6)	0 (218.8)

Tabela 5.2.27: Porcentagem de sucesso para problemas com $n = 90$ tarefas e sementes não-aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X90	100 (49.2)	100 (49.4)	100 (49.3)	90 (51.2)	90 (49.4)	90 (49.1)	90 (49.3)
07X90	60 (92.3)	80 (92.6)	60 (92.4)	50 (94.3)	40 (92.6)	50 (92.3)	40 (92.5)
10X90	60 (136.9)	50 (137.0)	50 (137.0)	60 (138.9)	50 (137.1)	50 (136.8)	40 (137.0)
15X90	30 (211.7)	20 (211.8)	30 (211.7)	50 (213.6)	60 (211.9)	30 (211.6)	40 (211.8)
20X90	10 (288.9)	10 (289.0)	20 (288.9)	0 (290.8)	40 (289.1)	20 (288.8)	20 (289.0)
25X90	40 (366.4)	0 (366.5)	0 (366.4)	20 (368.2)	20 (366.6)	0 (366.3)	20 (366.6)

Tabela 5.2.28: Porcentagem de sucesso para problema com $n = 100$ tarefas e sementes não-aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X100	100 (61.0)	100 (61.2)	100 (61.1)	100 (63.4)	100 (61.3)	100 (61.0)	100 (61.1)
07X100	60 (114.3)	60 (114.5)	70 (114.3)	70 (116.7)	70 (114.6)	90 (114.2)	70 (114.4)
10X100	50 (170.4)	30 (170.6)	50 (170.5)	50 (172.9)	80 (170.8)	60 (170.4)	60 (170.7)
15X100	20 (262.9)	0 (263.1)	20 (263)	50 (265.3)	10 (263.1)	10 (262.8)	20 (263.1)
20X100	20 (358.6)	0 (358.9)	10 (358.8)	20 (361.2)	20 (359)	20 (358.6)	10 (358.9)
25X100	20 (454.8)	10 (455)	10 (454.9)	20 (457.3)	0 (455.1)	20 (454.8)	30 (455.1)

Tabela 5.2.29: Porcentagem de sucesso para problemas com $n = 110$ tarefas e sementes não-aleatórias.

Classe de problema $m \times n$	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X110	80 (74.1)	100 (74.37)	80 (74.24)	90 (77.08)	80 (74.49)	90 (74.04)	80 (74.32)
07X110	70 (139)	40 (139.2)	60 (139.1)	70 (141.9)	60 (139.3)	70 (138.9)	70 (139.1)
10X110	70 (206.8)	50 (207.1)	70 (206.8)	70 (209.7)	60 (207.1)	60 (206.7)	70 (206.9)
15X110	30 (319.8)	20 (320)	40 (320)	10 (322.6)	20 (320)	30 (319.6)	30 (320)
20X110	40 (436)	0 (436.3)	10 (436.2)	10 (439)	20 (436.4)	0 (436)	20 (436.3)
25X110	30 (554.4)	0 (554.8)	20 (554.6)	40 (557.1)	10 (554.6)	30 (554.2)	10 (554.3)

Tabela 5.2.30: Média dos desvios relativos para problemas com $n = 70$ tarefas e sementes não-aleatórias.

Classe de problema m x n	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X70	0.07	0.09	0.13	0.06	0.12	0.10	0.12
07X70	1.41	1.43	1.41	1.38	1.37	1.28	1.35
10X70	3.48	3.63	3.58	3.49	3.41	3.59	3.51
15X70	8.09	8.18	8.10	8.11	8.12	8.10	8.11
20X70	14.60	14.76	14.56	14.56	14.59	14.54	14.46
25X70	15.77	15.86	15.73	15.77	15.71	15.81	15.83

Tabela 5.2.31: Média dos desvios relativos para problemas com $n = 90$ tarefas e sementes não-aleatórias.

Classe de problema m x n	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X90	0.07	0.07	0.07	0.07	0.08	0.08	0.08
07X90	0.65	0.61	0.63	0.64	0.67	0.65	0.68
10X90	2.37	2.40	2.44	2.36	2.42	2.39	2.45
15X90	6.28	6.31	6.27	6.21	6.15	6.31	6.21
20X90	8.92	8.95	8.94	9.01	8.91	8.93	8.98
25X90	12.92	13.02	12.98	12.96	12.89	13.01	12.94

Tabela 5.2.32: Média dos desvios relativos para problemas com $n = 100$ tarefas e sementes não-aleatórias.

Classe de problema m x n	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X100	0.13	0.13	0.13	0.13	0.13	0.13	0.13
07X100	0.70	0.70	0.69	0.65	0.66	0.61	0.62
10X100	1.88	2.01	1.92	1.94	1.84	1.88	1.94
15X100	4.90	5.06	4.98	4.86	4.95	4.87	4.89
20X100	8.39	8.44	8.40	8.38	8.37	8.41	8.38
25X100	12.43	12.45	12.44	12.47	12.43	12.41	12.30

Tabela 5.2.33: Média dos desvios relativos para problemas com n = 110 tarefas e sementes não-aleatórias.

Classe de problema m x n	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
04X110	0.06	0.03	0.06	0.05	0.05	0.05	0.06
07X110	0.61	0.64	0.63	0.60	0.62	0.60	0.61
10X110	1.78	1.83	1.79	1.73	1.81	1.76	1.80
15X110	5.41	5.52	5.43	5.48	5.41	5.45	5.40
20X110	8.53	8.71	8.63	8.67	8.66	8.64	8.60
25X110	10.79	10.83	10.76	10.79	10.79	10.75	10.79

Tabela 5.2.34: Resultados globais para problemas de grande porte e sementes não-aleatórias.

No. de tarefas	% PMX	% MPX	% OX	% EOX	% UM CORTE	% CX	% LOX
70	36.67	25	26.67	38.33	35	33.33	31.67
	7.24	7.32	7.25	7.23	7.22	7.24	7.23
	(109.42)	(109.56)	(109.48)	(110.52)	(109.54)	(109.38)	(109.48)
90	50	43.33	43.33	45	50	40	41.67
	5.2	5.23	5.22	5.21	5.18	5.23	5.22
	(181.13)	(181.37)	(181.21)	(183.03)	(181.36)	(181.09)	(181.25)
100	45	33.33	43.33	51.67	46.67	50	48.33
	4.74	4.8	4.76	4.74	4.73	4.72	4.71
	(221.84)	(222.08)	(221.98)	(224.14)	(222.15)	(221.84)	(222)
110	53.33	35	46.67	48.33	41.67	46.67	46.67
	4.53	4.59	4.55	4.55	4.56	4.54	4.54
	(270.75)	(270.99)	(270.88)	(273.6)	(271.16)	(270.85)	(271.05)
valores globais	46.25	34.17	40	45.83	43.34	42.5	42.09
	5.43	5.49	5.45	5.43	5.42	5.43	5.43

5.2.3. LIMITANTES INFERIORES CALCULADOS

A seguir apresentam-se as tabelas com os limitantes inferiores calculados para os problemas gerados aleatoriamente neste trabalho (Tabelas 5.2.35 a 5.2.41). As tabelas são divididas pelo número de tarefas do problema. Na tabela 5.2.42 observa-se os valores médios dos limitantes inferiores para cada tamanho de problema.

Tabela 5.2.35: Limitantes inferiores calculados para os problemas com $n = 10$ tarefas.

PROBLEMA	L.B.	PROBLEMA	L.B.	PROBLEMA	L.B.
04X010P01	556	07X010P01	814	10X010P01	931
04X010P02	758	07X010P02	750	10X010P02	992
04X010P03	681	07X010P03	737	10X010P03	887
04X010P04	677	07X010P04	758	10X010P04	902
04X010P05	717	07X010P05	843	10X010P05	878
04X010P06	622	07X010P06	727	10X010P06	887
04X010P07	765	07X010P07	763	10X010P07	1006
04X010P08	621	07X010P08	783	10X010P08	855
04X010P09	688	07X010P09	797	10X010P09	843
04X010P10	798	07X010P10	753	10X010P10	821
média	688.3	média	772.5	média	900.2
PROBLEMA	L.B.	PROBLEMA	L.B.	PROBLEMA	L.B.
15X010P01	1082	20X010P01	1385	25X010P01	1778
15X010P02	1139	20X010P02	1230	25X010P02	1549
15X010P03	1221	20X010P03	1334	25X010P03	1667
15X010P04	1230	20X010P04	1365	25X010P04	1633
15X010P05	1100	20X010P05	1443	25X010P05	1639
15X010P06	1067	20X010P06	1430	25X010P06	1650
15X010P07	1188	20X010P07	1592	25X010P07	1624
15X010P08	1232	20X010P08	1216	25X010P08	1605
15X010P09	1229	20X010P09	1267	25X010P09	1575
15X010P10	984	20X010P10	1343	25X010P10	1736
média	1147.2	média	1360.5	média	1645.6
MÉDIA TOTAL: 1085.7					

Tabela 5.2.36: Limitantes inferiores calculados para os problemas com $n = 30$ tarefas.

PROBLEMA	L.B.	PROBLEMA	L.B.	PROBLEMA	L.B.
04X030P01	1658	07X030P01	1840	10X030P01	2139
04X030P02	1595	07X030P02	1826	10X030P02	2068
04X030P03	1602	07X030P03	1729	10X030P03	2024
04X030P04	1662	07X030P04	1954	10X030P04	1978
04X030P05	1648	07X030P05	1750	10X030P05	2045
04X030P06	1573	07X030P06	2007	10X030P06	1951
04X030P07	1895	07X030P07	1743	10X030P07	1906
04X030P08	1585	07X030P08	1691	10X030P08	2145
04X030P09	1542	07X030P09	1878	10X030P09	1909
04X030P10	1721	07X030P10	1754	10X030P10	1854
média	1648.1	média	1817.2	média	2001.9
PROBLEMA	L.B.	PROBLEMA	L.B.	PROBLEMA	L.B.
15X030P01	2141	20X030P01	2488	25X030P01	2450
15X030P02	2197	20X030P02	2353	25X030P02	2556
15X030P03	2165	20X030P03	2433	25X030P03	2513
15X030P04	2356	20X030P04	2410	25X030P04	2624
15X030P05	2084	20X030P05	2383	25X030P05	2626
15X030P06	2278	20X030P06	2384	25X030P06	2760
15X030P07	2197	20X030P07	2284	25X030P07	2608
15X030P08	2194	20X030P08	2648	25X030P08	2622
15X030P09	2265	20X030P09	2421	25X030P09	2649
15X030P10	2154	20X030P10	2423	25X030P10	2585
média	2203.1	média	2422.7	média	2599.3
MÉDIA PARCIAL: 2115.4					

Tabela 5.2.37: Limitantes inferiores calculados para os problemas com $n = 50$ tarefas.

PROBLEMA	L.B.	PROBLEMA	L.B.	PROBLEMA	L.B.
04X050P01	2641	07X050P01	2947	10X050P01	2981
04X050P02	2815	07X050P02	3218	10X050P02	2925
04X050P03	2686	07X050P03	2991	10X050P03	3098
04X050P04	2797	07X050P04	2765	10X050P04	3065
04X050P05	2862	07X050P05	2832	10X050P05	3050
04X050P06	2883	07X050P06	2898	10X050P06	2887
04X050P07	2784	07X050P07	2936	10X050P07	3055
04X050P08	2853	07X050P08	3050	10X050P08	3077
04X050P09	2676	07X050P09	2841	10X050P09	3164
04X050P10	2752	07X050P10	3029	10X050P10	2826
média	2774.9	média	2950.7	média	3012.8
PROBLEMA	L.B.	PROBLEMA	L.B.	PROBLEMA	L.B.
15X050P01	3178	20X050P01	3473	25X050P01	3642
15X050P02	3425	20X050P02	3574	25X050P02	3599
15X050P03	3112	20X050P03	3402	25X050P03	3694
15X050P04	3329	20X050P04	3314	25X050P04	3533
15X050P05	3477	20X050P05	3539	25X050P05	3638
15X050P06	3191	20X050P06	3360	25X050P06	3533
15X050P07	3268	20X050P07	3590	25X050P07	3802
15X050P08	3130	20X050P08	3366	25X050P08	3685
15X050P09	2996	20X050P09	3457	25X050P09	3681
15X050P10	3077	20X050P10	3484	25X050P10	3733
média	3218.3	média	3455.9	média	3654
MÉDIA PARCIAL: 3177.8					

Tabela 5.2.38: Limitantes inferiores calculados para os problemas com $n = 70$ tarefas.

PROBLEMA	L.B.	PROBLEMA	L.B.	PROBLEMA	L.B.
04X070P01	3687	07X070P01	3847	10X070P01	4037
04X070P02	3576	07X070P02	3607	10X070P02	4130
04X070P03	3777	07X070P03	3953	10X070P03	3985
04X070P04	3595	07X070P04	3766	10X070P04	3911
04X070P05	3695	07X070P05	3849	10X070P05	4324
04X070P06	3547	07X070P06	3921	10X070P06	3884
04X070P07	3608	07X070P07	4094	10X070P07	4022
04X070P08	3729	07X070P08	3993	10X070P08	3967
04X070P09	3956	07X070P09	3869	10X070P09	4036
04X070P10	4042	07X070P10	4011	10X070P10	4046
média	3721.2	média	3891	média	4034.2
PROBLEMA	L.B.	PROBLEMA	L.B.	PROBLEMA	L.B.
15X070P01	4006	20X070P01	4311	25X070P01	4899
15X070P02	4277	20X070P02	4363	25X070P02	4799
15X070P03	4298	20X070P03	4479	25X070P03	4815
15X070P04	4270	20X070P04	4361	25X070P04	4746
15X070P05	4550	20X070P05	4458	25X070P05	4906
15X070P06	4219	20X070P06	4346	25X070P06	4572
15X070P07	4427	20X070P07	4500	25X070P07	4665
15X070P08	4331	20X070P08	4347	25X070P08	4822
15X070P09	4312	20X070P09	4313	25X070P09	4528
15X070P10	4136	20X070P10	4426	25X070P10	4776
média	4282.6	média	4390.4	média	4752.8
MÉDIA PARCIAL: 4178.7					

Tabela 5.2.39: Limitantes inferiores calculados para os problemas com $n = 90$ tarefas.

PROBLEMA	L.B.	PROBLEMA	L.B.	PROBLEMA	L.B.
04X090P01	4760	07X090P01	4835	10X090P01	5202
04X090P02	4856	07X090P02	5000	10X090P02	5252
04X090P03	4912	07X090P03	5077	10X090P03	4858
04X090P04	4912	07X090P04	4741	10X090P04	5127
04X090P05	5007	07X090P05	5022	10X090P05	5085
04X090P06	5029	07X090P06	4888	10X090P06	5353
04X090P07	4770	07X090P07	5076	10X090P07	4879
04X090P08	4752	07X090P08	5195	10X090P08	5357
04X090P09	4656	07X090P09	4875	10X090P09	5104
04X090P10	5130	07X090P10	5095	10X090P10	5305
média	4878.4	média	4980.4	média	5152.2
PROBLEMA	L.B.	PROBLEMA	L.B.	PROBLEMA	L.B.
15X090P01	5439	20X090P01	5345	25X090P01	5909
15X090P02	5282	20X090P02	5750	25X090P02	5768
15X090P03	5681	20X090P03	5594	25X090P03	5691
15X090P04	5345	20X090P04	5489	25X090P04	5942
15X090P05	5390	20X090P05	5430	25X090P05	5898
15X090P06	5580	20X090P06	5588	25X090P06	5884
15X090P07	5437	20X090P07	5922	25X090P07	5704
15X090P08	5316	20X090P08	5974	25X090P08	5674
15X090P09	5374	20X090P09	5687	25X090P09	5789
15X090P10	4995	20X090P10	5620	25X090P10	5994
média	5383.9	média	5639.9	média	5825.3
MÉDIA PARCIAL: 5310.0					

Tabela 5.2.40: Limitantes inferiores calculados para os problemas com $n = 100$ tarefas.

PROBLEMA	L.B.	PROBLEMA	L.B.	PROBLEMA	L.B.
04X100P01	4965	07X100P01	5255	10X100P01	5481
04X100P02	5531	07X100P02	5624	10X100P02	5476
04X100P03	5220	07X100P03	5338	10X100P03	5993
04X100P04	5723	07X100P04	5779	10X100P04	5478
04X100P05	5334	07X100P05	5443	10X100P05	5701
04X100P06	5287	07X100P06	5556	10X100P06	5993
04X100P07	5280	07X100P07	5814	10X100P07	5856
04X100P08	5155	07X100P08	5377	10X100P08	5803
04X100P09	5283	07X100P09	5455	10X100P09	5776
04X100P10	5194	07X100P10	5594	10X100P10	5702
média	5297.2	média	5523.5	média	5725.9
PROBLEMA	L.B.	PROBLEMA	L.B.	PROBLEMA	L.B.
15X100P01	5685	20X100P01	6296	25X100P01	6280
15X100P02	5859	20X100P02	6406	25X100P02	6448
15X100P03	6030	20X100P03	6224	25X100P03	6279
15X100P04	6006	20X100P04	5940	25X100P04	6271
15X100P05	5774	20X100P05	6148	25X100P05	6214
15X100P06	6050	20X100P06	6506	25X100P06	6241
15X100P07	5797	20X100P07	6107	25X100P07	6435
15X100P08	5749	20X100P08	5964	25X100P08	6327
15X100P09	6113	20X100P09	6067	25X100P09	6455
15X100P10	6008	20X100P10	6123	25X100P10	6467
média	5907.1	média	6178.1	média	6341.7
MÉDIA PARCIAL: 5828.9					

Tabela 5.2.41: Limitantes inferiores calculados para os problemas com $n = 110$ tarefas.

PROBLEMA	L.B.	PROBLEMA	L.B.	PROBLEMA	L.B.
04X110P01	6264	07X110P01	6128	10X110P01	6266
04X110P02	5945	07X110P02	6059	10X110P02	6349
04X110P03	5926	07X110P03	6184	10X110P03	6102
04X110P04	6103	07X110P04	6411	10X110P04	5873
04X110P05	6152	07X110P05	5996	10X110P05	6038
04X110P06	5438	07X110P06	5892	10X110P06	6750
04X110P07	5776	07X110P07	6045	10X110P07	6007
04X110P08	5782	07X110P08	5759	10X110P08	6335
04X110P09	6098	07X110P09	5916	10X110P09	6025
04X110P10	5792	07X110P10	5759	10X110P10	6385
média	5927.6	média	6014.9	média	6213
PROBLEMA	L.B.	PROBLEMA	L.B.	PROBLEMA	L.B.
15X110P01	6375	20X110P01	6722	25X110P01	7154
15X110P02	6188	20X110P02	6742	25X110P02	6848
15X110P03	6478	20X110P03	6174	25X110P03	6861
15X110P04	6110	20X110P04	6635	25X110P04	7178
15X110P05	6666	20X110P05	6883	25X110P05	6752
15X110P06	6233	20X110P06	6594	25X110P06	6898
15X110P07	6293	20X110P07	6603	25X110P07	6961
15X110P08	6434	20X110P08	6596	25X110P08	6598
15X110P09	6382	20X110P09	6615	25X110P09	7009
15X110P10	6206	20X110P10	6834	25X110P10	6466
média	6336.5	média	6639.8	média	6872.5
MÉDIA PARCIAL: 6334.1					

Tabela 5.3.42: Médias totais dos limitantes inferiores para problemas de pequeno, médio e grande portes.

	Pequeno	Médio	Grande
L.B.(Calculado)	1085.72	2646.58	5412.92

5.3. ANÁLISE DOS RESULTADOS OBTIDOS

Um resultado que marca todas as tabelas (sementes aleatórias e não-aleatórias) é em relação ao tempo de computação. Dentro de uma mesma classe de problema os tempos de computação são quase idênticos, mostrando que a escolha do melhor algoritmo será determinada exclusivamente pelo seu desempenho em relação a solução final.

De forma globalizada para problemas de pequeno porte e sementes aleatórias (Tabela 5.2.3) o algoritmo UM CORTE se destacou na porcentagem de sucesso alcançando 28.33% , 6.66% acima do segundo melhor(OX) e 15% acima do último(CX). Na média dos desvios relativos não há vantagem de nenhum algoritmo, mostrando que ao longo do tempo, os algoritmos apresentados neste trabalho tem desempenhos iguais.

Para problemas de pequeno porte e sementes não-aleatórias, a maior porcentagem de sucesso é do operador MPX com 66.67%, que corresponde a 5.0% melhor que os piores (OX e UM CORTE, 61.67%) e 1.67% acima dos segundos melhores (EOX e CX, 65.0%). Olhando para as médias dos desvios relativos não é encontrado novamente um operador que se destaque.

Em problemas de médio porte e sementes aleatórias observa-se que o algoritmo MPX deixou a desejar na porcentagem de sucessos em todas as classes de problema (Tabelas 5.2.4 e 5.2.5). De forma globalizada (Tabela 5.2.8) encontra-se o algoritmo LOX se destacando com 25.83% de sucesso, 20.0% acima do último colocado (MPX) e 6.66% acima do segundo melhor (CX).

Para problemas de médio porte com sementes não-aleatórias, de novo o MPX se mostrou ineficiente em relação aos outros algoritmos em quase todas as classes de problemas (Tabelas 5.2.21 e 5.2.22). Da tabela 5.2.25, onde encontram-se a globalização dos problemas de médio porte e sementes não-aleatórias, observa-se o melhor desempenho, novamente, do algoritmo UM CORTE com 39.17%, 5% acima dos segundos melhores (PMX e CX, 34.17%) e 15.84% acima do último colocado

(MPX, 23.33%). Em relação aos desvios relativos não encontra-se de novo nenhum algoritmo que se destaque.

Em problemas de grande porte e sementes aleatórias novamente encontra-se o mau desempenho do algoritmo MPX em todas as classes de problemas (Tabelas 5.2.9 a 5.2.11), mas também os maus desempenhos dos algoritmos OX, CX e LOX. De forma globalizada (Tabela 5.2.17) se destacam dois algoritmos PMX (19.58%) e o UM CORTE (20.84%). Novamente não se observa destaque entre os algoritmos no que diz respeito aos desvios relativos, entretanto se for considerada a média dos L.B. para estes problemas, a diferença do PMX (9.55%) para o MPX (9.97%) que é de 0.42% representa no final da programação das tarefas uma diferença de 22.73 unidades de tempo (normalmente o minuto) o que pode ser de muito custo para a fábrica. Quando observa-se essa diferença para o problema de 25 máquinas, 110 tarefas e sementes aleatórias ela se eleva para aproximadamente 1h (considerando a unidade de tempo minuto).

Nos problemas de grande porte e sementes não-aleatórias os algoritmos MPX e LOX tiveram desempenhos baixos em relação aos outros observando as tabelas 5.2.26 a 5.2.29. De forma globalizada (Tabela 5.2.34) novamente o algoritmo PMX se destacou com 46.25% de sucesso, 0.42% acima do EOX, que também teve um desempenho a se destacar com 45.84% de sucesso. Ainda o PMX ficou 12.08% acima do último colocado (MPX, 34.17%). No que diz respeito aos desvios relativos, não encontramos nenhum algoritmo se destacando.

Para analisar os desempenhos das sementes aleatórias e não-aleatórias construiu-se as tabelas 5.3.1 a 5.3.3 contendo a média dos desvios relativos correspondente ao número de tarefas. Essas tabelas são colocadas em gráficos que mostram a relação existente entre número de tarefas e a média dos desvios relativos (Gráficos 5.3.1 a 5.3.7) para sementes aleatórias e não-aleatórias. Para observar a distribuição dos desvios construiu-se um outro tipo de gráfico (Gráficos 5.3.8 a 5.3.14, sementes aleatórias e Gráficos 5.3.15 a 5.3.21, sementes não aleatórias) que mostram a dispersão dos desvios em relação ao número de tarefas.

Tabela 5.3.1: Média dos desvios relativos em função do número de tarefas para os algoritmos PMX, MPX e OX.

n	PMX		MPX		OX	
	não-aleatória	aleatória	não-aleatória	aleatória	não-aleatória	aleatória
10	17.54%	19.16%	17.56%	18.95%	17.53%	19.17%
30	11.86%	14.53%	12.11%	14.86%	11.84%	14.37%
50	9.18%	12.57%	9.35%	12.97%	9.18%	12.40%
70	7.24%	11.36%	7.32%	11.83%	7.25%	11.48%
90	5.20%	8.96%	5.23%	9.58%	5.22%	9.19%
100	4.74%	9.00%	4.80%	9.43%	4.76%	9.15%
110	4.53%	8.90%	4.59%	9.04%	4.55%	8.99%

Tabela 5.3.2: Média dos desvios relativos em função do número de tarefas para os algoritmos EOX, UM CORTE e CX.

n	EOX		UM CORTE		CX	
	não-aleatória	aleatória	não-aleatória	aleatória	não-aleatória	aleatória
10	17.65%	18.95%	17.61%	18.79%	17.57%	19.20%
30	11.91%	14.74%	11.82%	14.42%	11.78%	14.83%
50	9.17%	12.58%	9.17%	12.64%	9.21%	12.54%
70	7.23%	11.42%	7.22%	11.44%	7.24%	11.51%
90	5.21%	9.39%	5.18%	9.19%	5.23%	9.37%
100	4.74%	9.16%	4.73%	8.74%	4.72%	9.12%
110	4.55%	8.81%	4.56%	8.81%	4.54%	8.81%

Tabela 5.3.3: Média dos desvios relativos em função do número de tarefas para o algoritmo LOX.

LOX		
n	não-aleatória	aleatória
10	17.64%	19.20%
30	11.85%	14.34%
50	9.20%	12.57%
70	7.23%	11.61%
90	5.22%	9.30%
100	4.71%	9.19%
110	4.54%	8.85%

Gráfico 5.3.1.: Relação entre número de tarefas e as média dos desvios relativos para o algoritmo PMX.

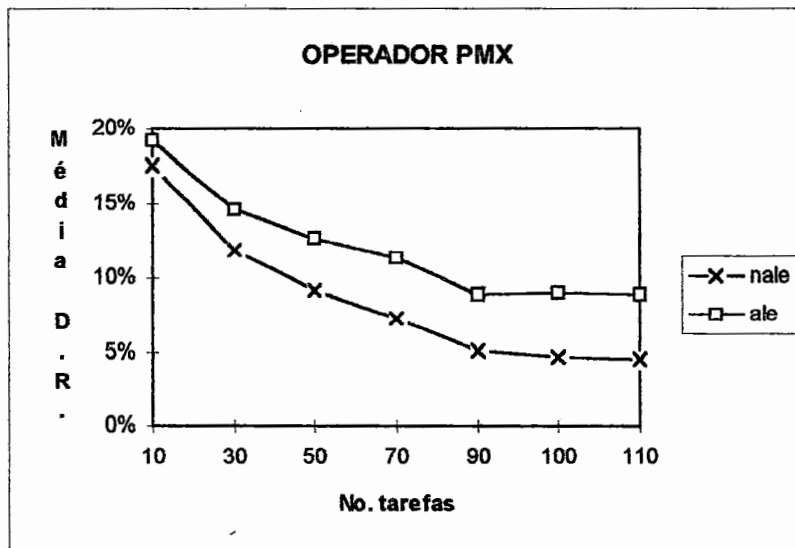


Gráfico 5.3.2.: Relação entre o número de tarefas e as médias dos desvios relativos para o algoritmo MPX

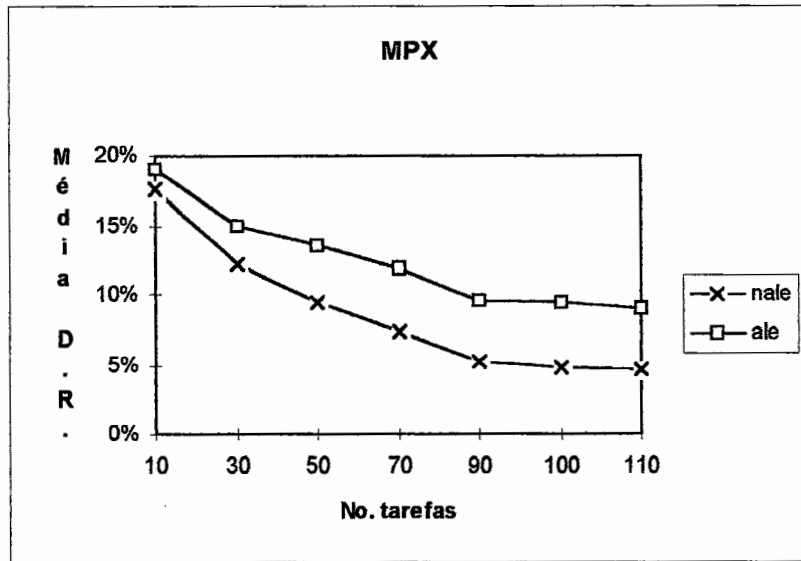


Gráfico 5.3.3.: Relação entre o número de tarefas e as médias dos desvios relativos para o algoritmo OX.

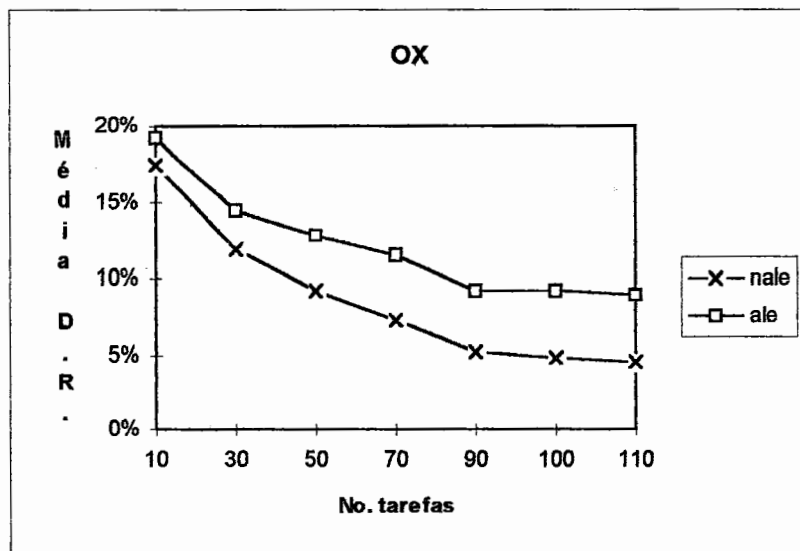


Gráfico 5.3.4: Relação entre o número de tarefas e as médias dos desvios relativos para o algoritmo EOX

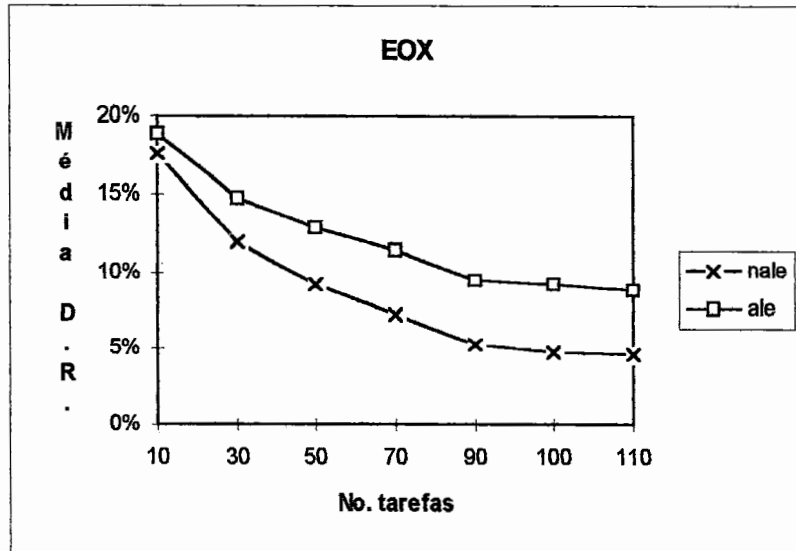


Gráfico 5.3.5.: Relação entre o número de tarefas e as média dos desvios relativos para o algoritmo UM CORTE.

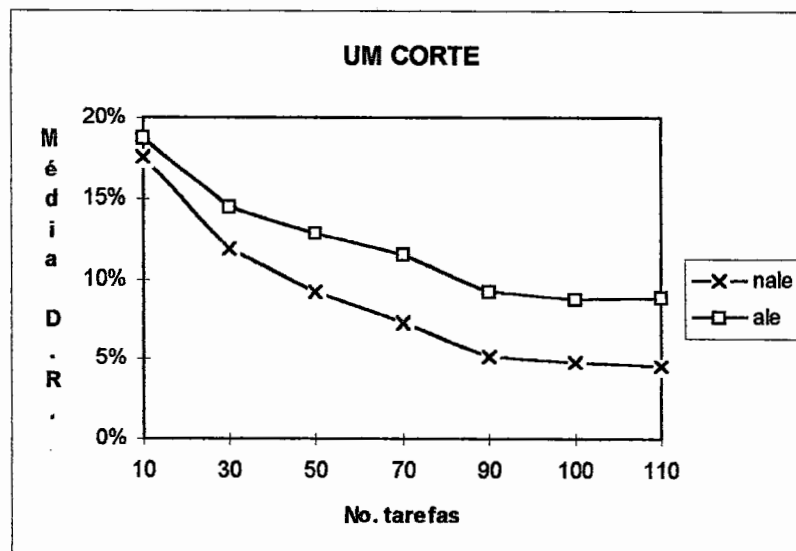


Gráfico 5.3.6.: Relação entre o número de tarefas e as médias dos desvios relativos para o algoritmo CX.

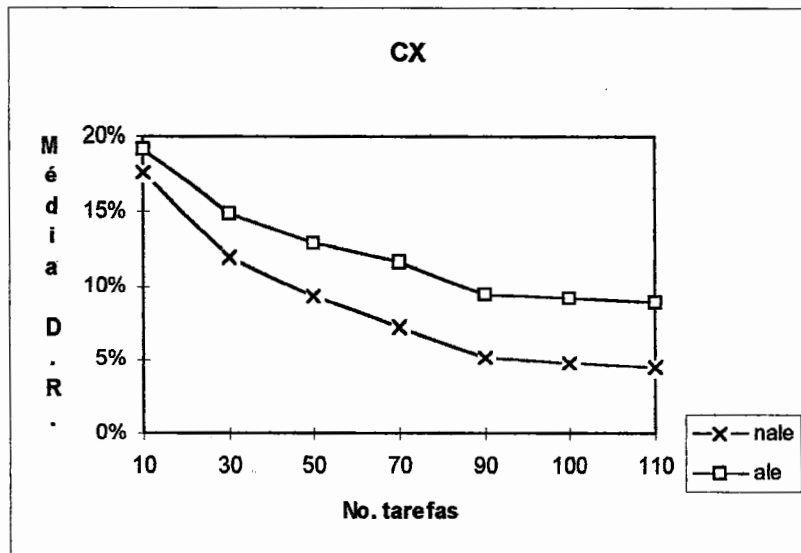
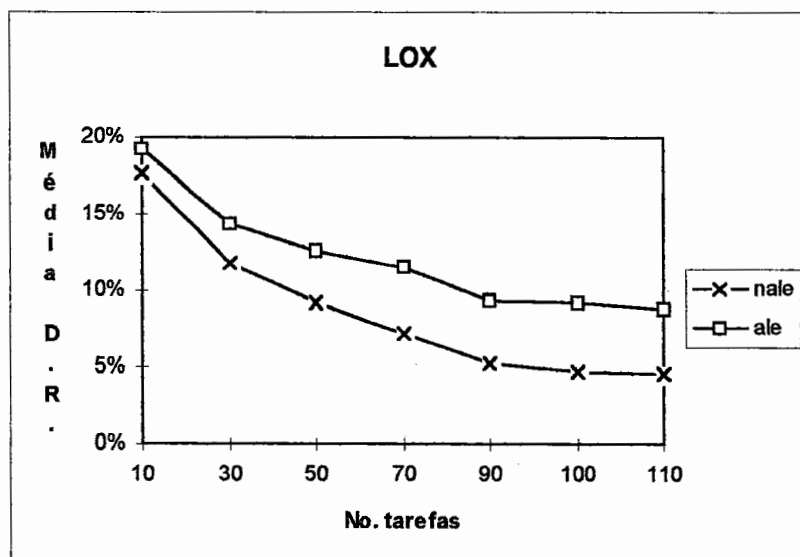


Gráfico 5.3.7.: Relação entre o número de tarefas e as médias dos desvios relativos para o algoritmo LOX.



SEMENTES ALEATÓRIAS

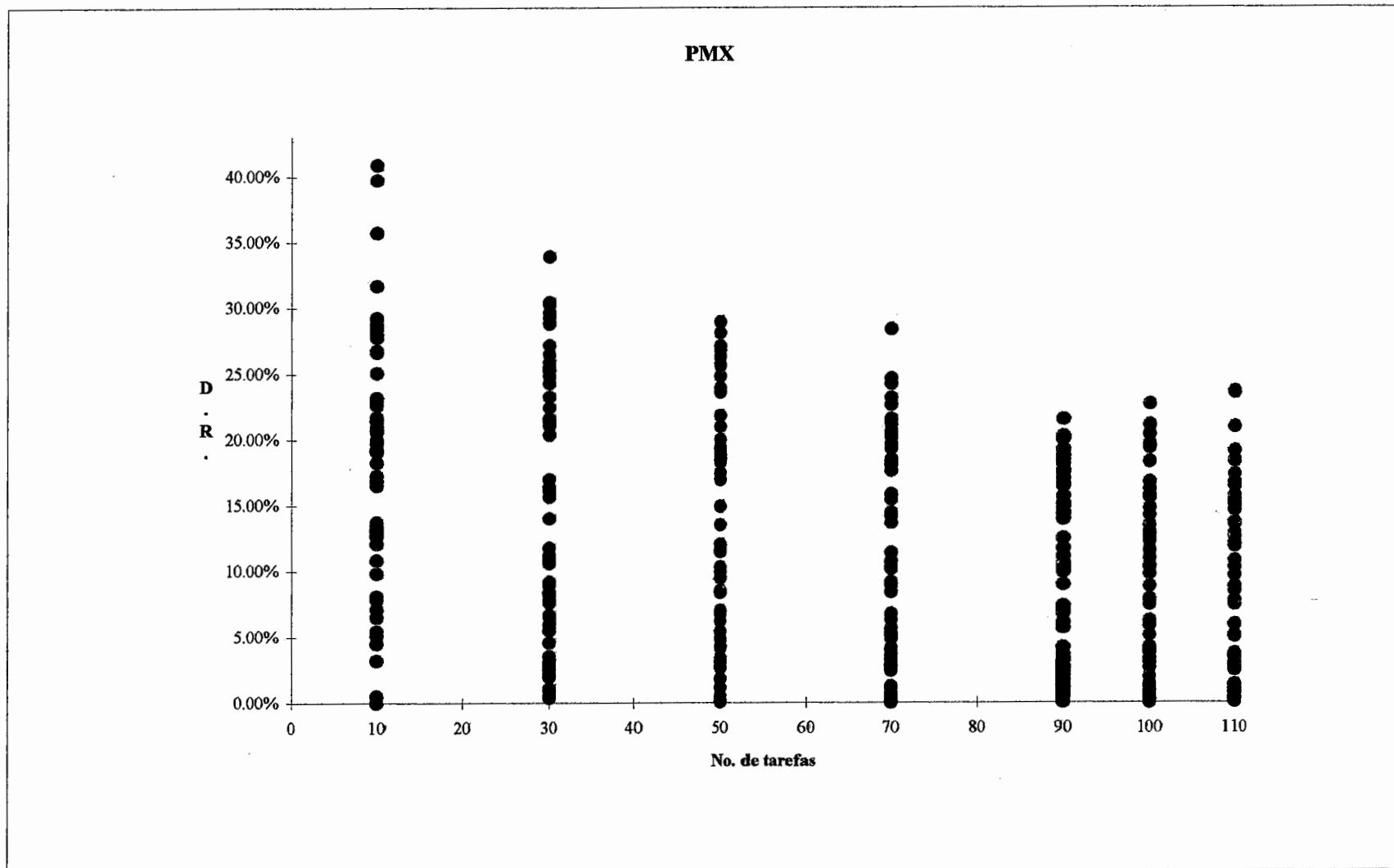


Gráfico 5.3.8: Dispersão dos desvios em relação ao número de tarefas para o operador PMX.

SEMENTES ALEATÓRIAS

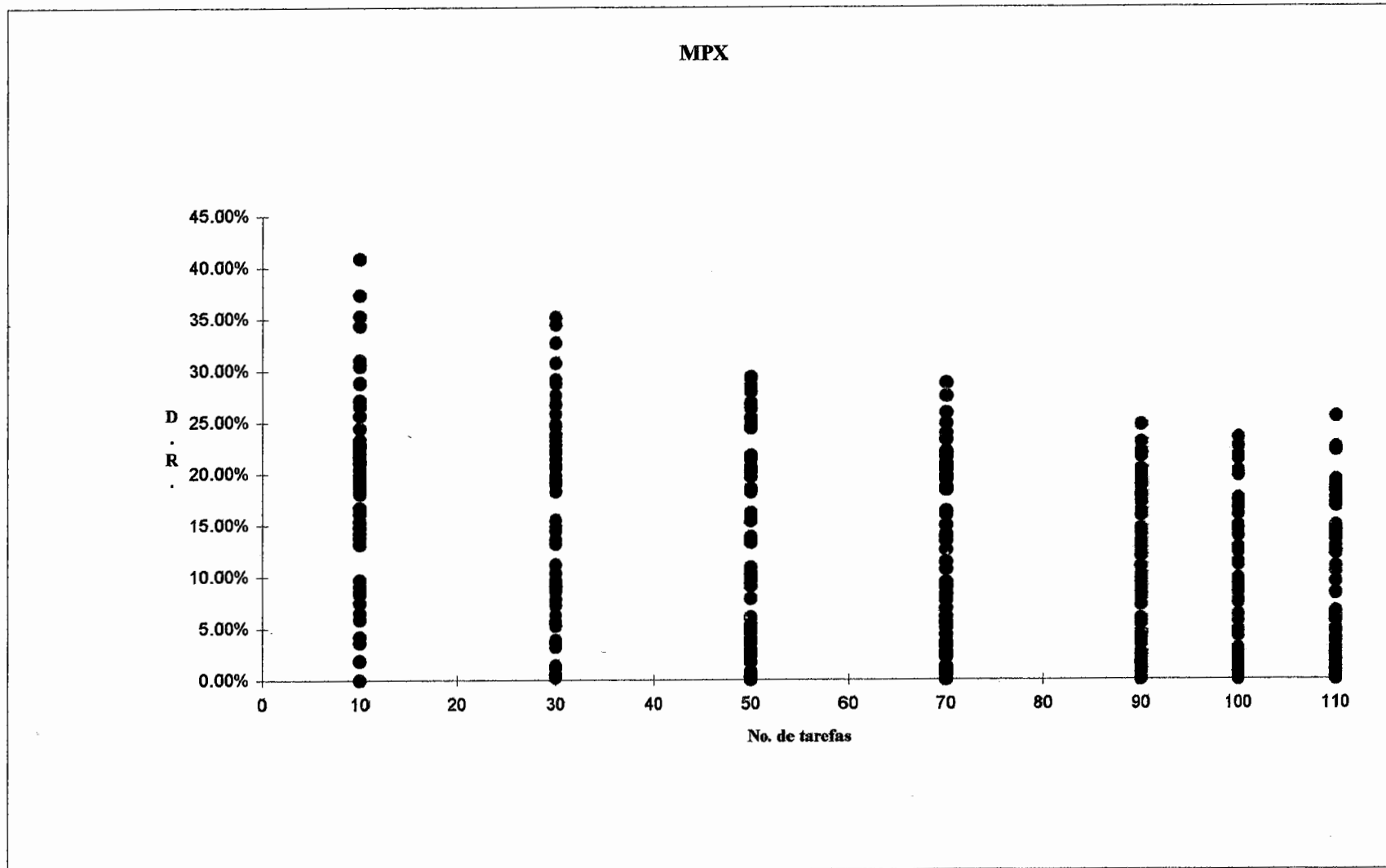


Gráfico 5.3.9: Dispersão dos desvios em relação ao número de tarefas para o operador MPX.

SEMENTES ALEATÓRIAS

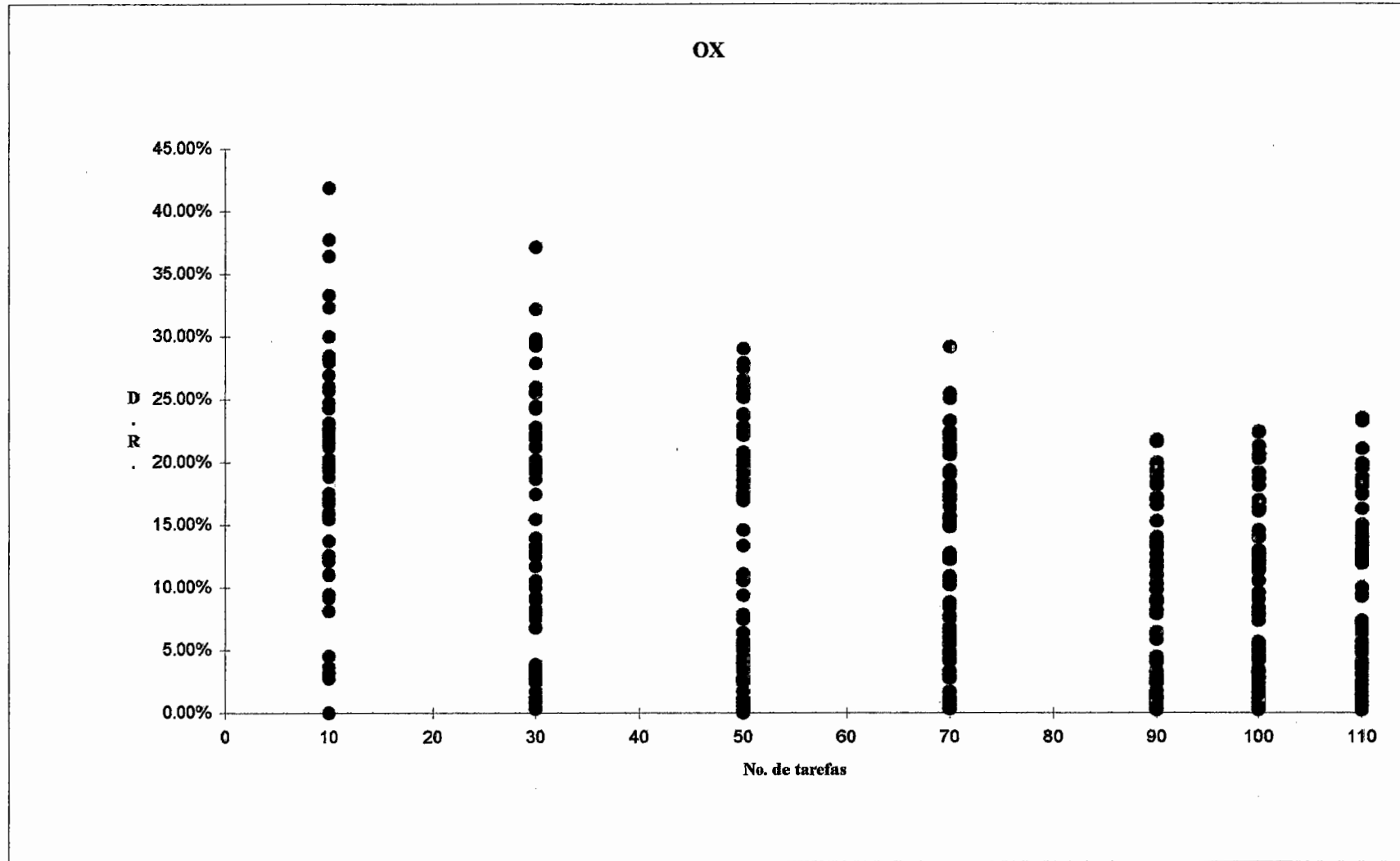


Gráfico 5.3.10: Dispersão dos desvios em relação ao número de tarefas para o operador OX.

SEMENTES ALEATÓRIAS

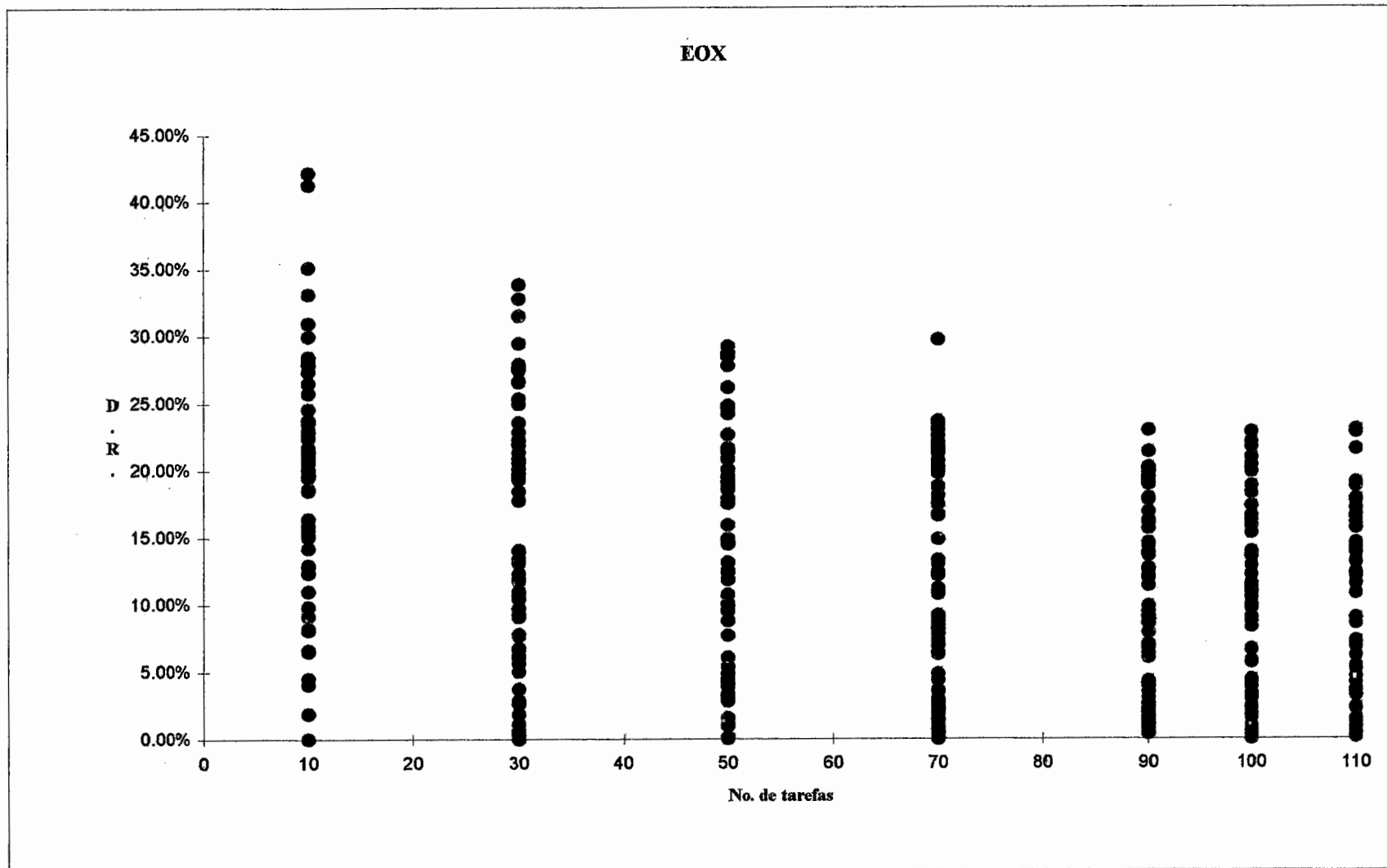


Gráfico 5.3.11: Dispersão dos desvios em relação ao número de tarefas para o operador EOX.

SEMENTES ALEATÓRIAS

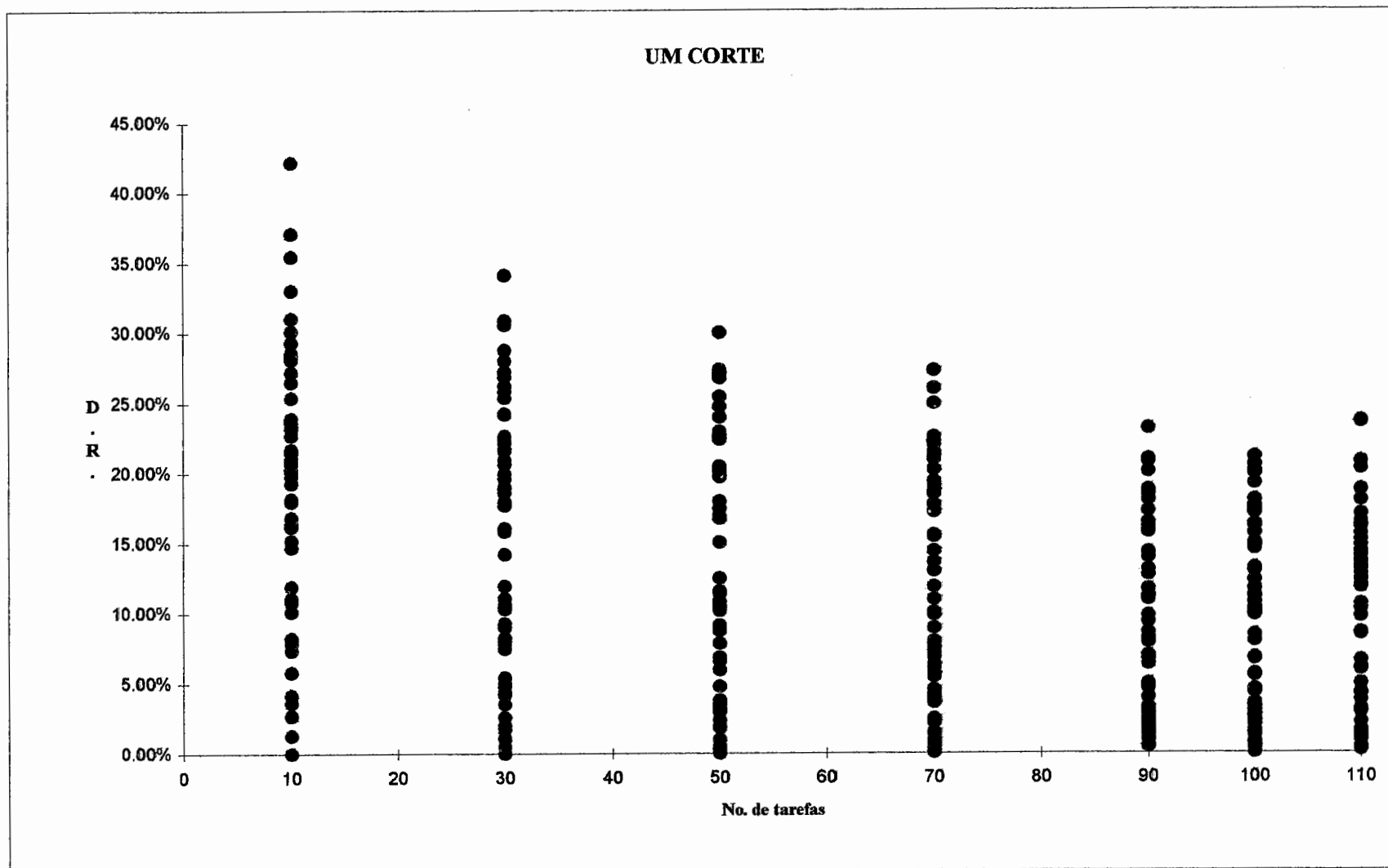


Gráfico 5.3.12: Dispersão dos desvios em relação ao número de tarefas para o operador UM CORTE.

SEMENTES ALEATÓRIAS

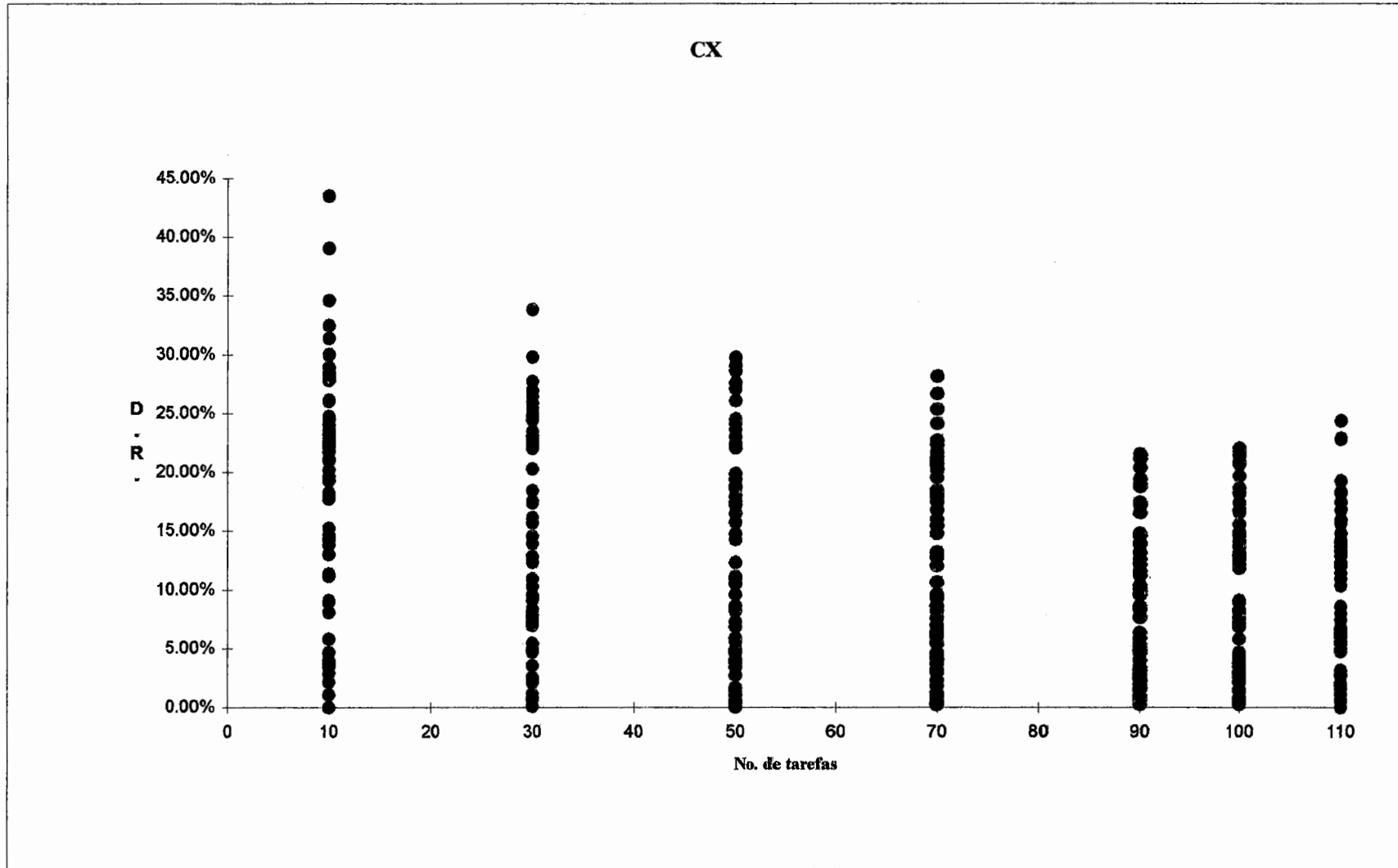


Gráfico 5.3.13: Dispersão dos desvios em relação ao número de tarefas para o operador CX.

SEMENTES ALEATÓRIAS

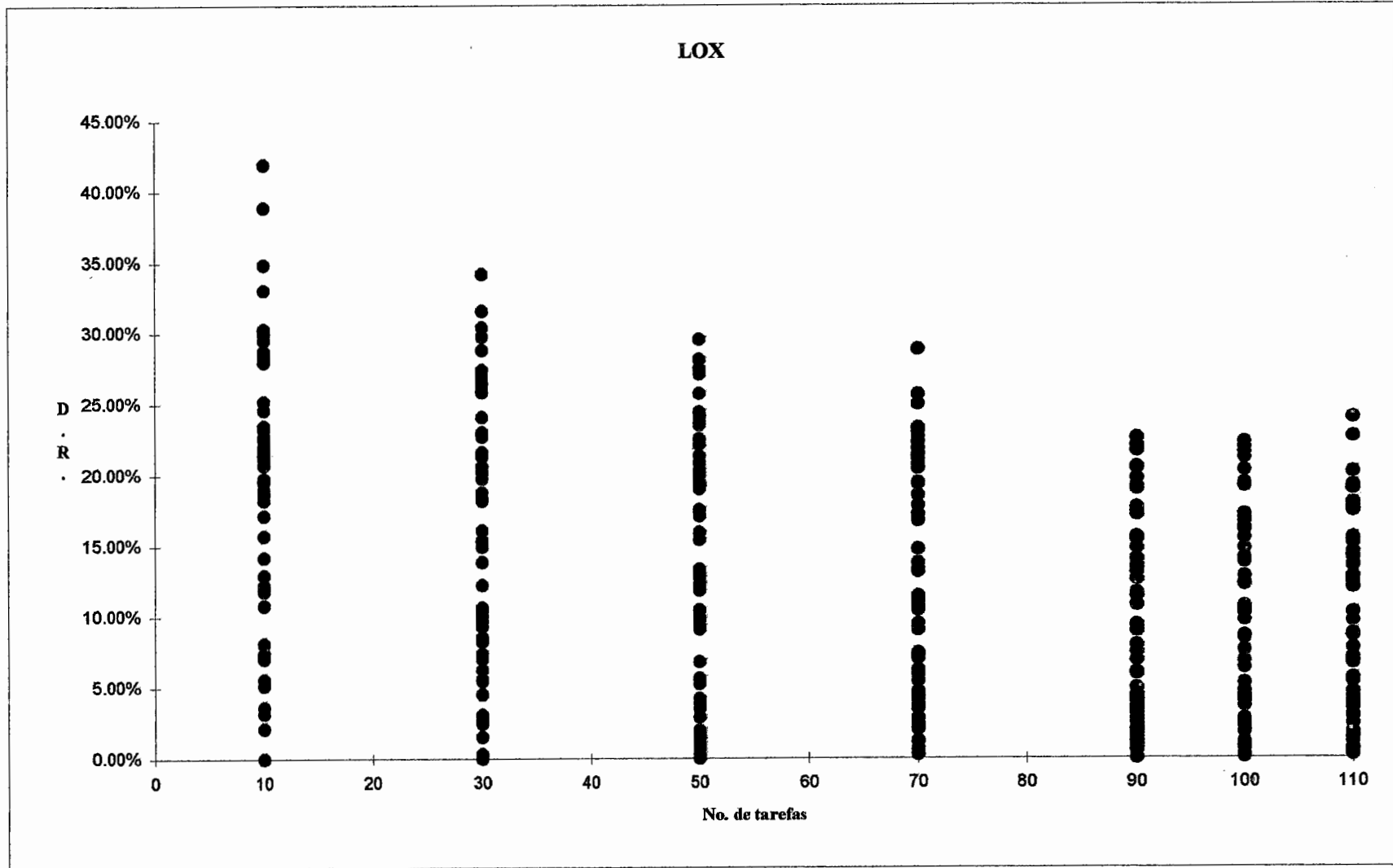


Gráfico 5.3.14: Dispersão dos desvios em relação ao número de tarefas para o operador LOX.

SEMENTES NÃO-ALEATÓRIAS

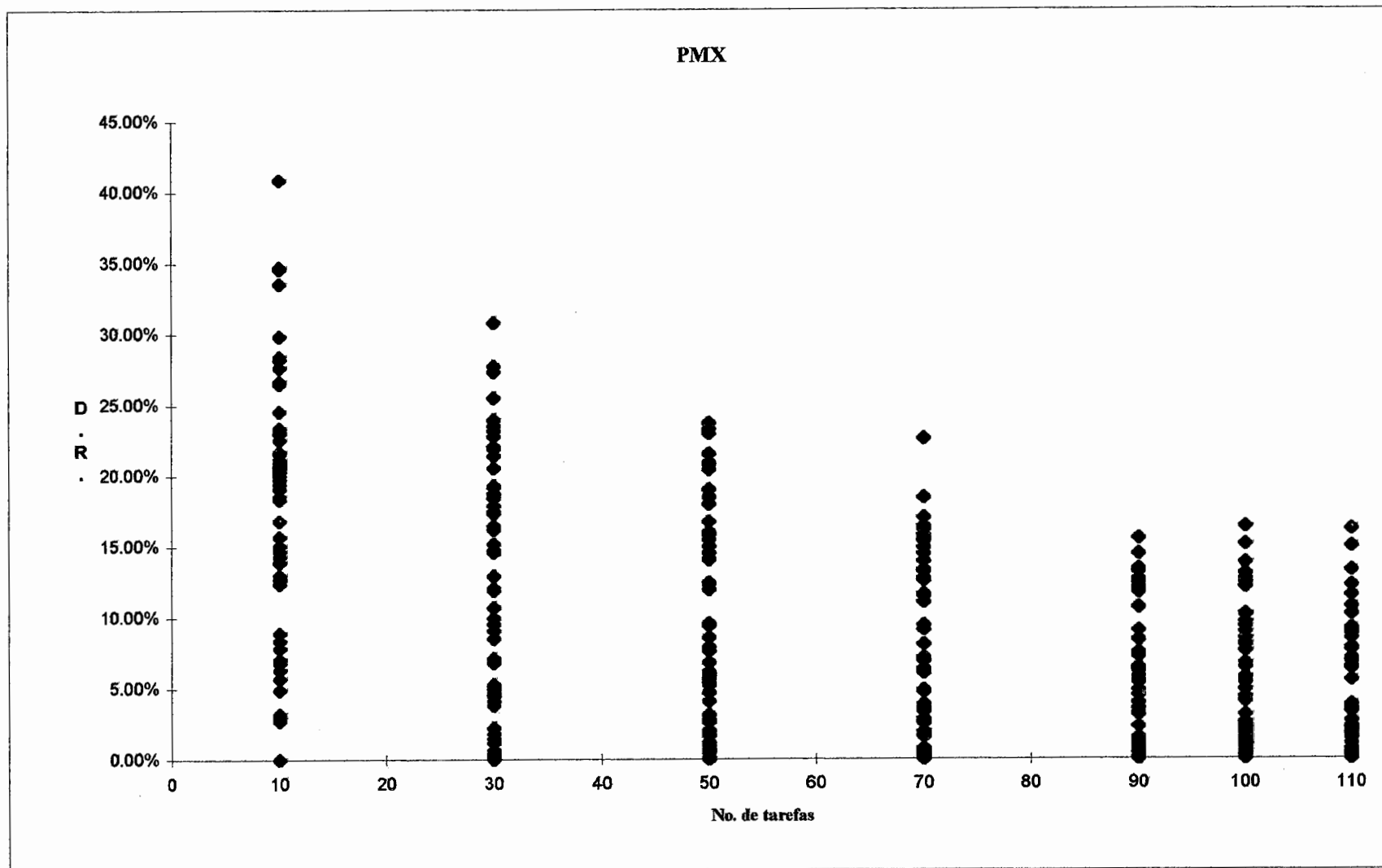


Gráfico 5.3.15: Dispersão dos desvios em relação ao número de tarefas para o operador PMX.

SEMENTES NÃO-ALEATÓRIAS

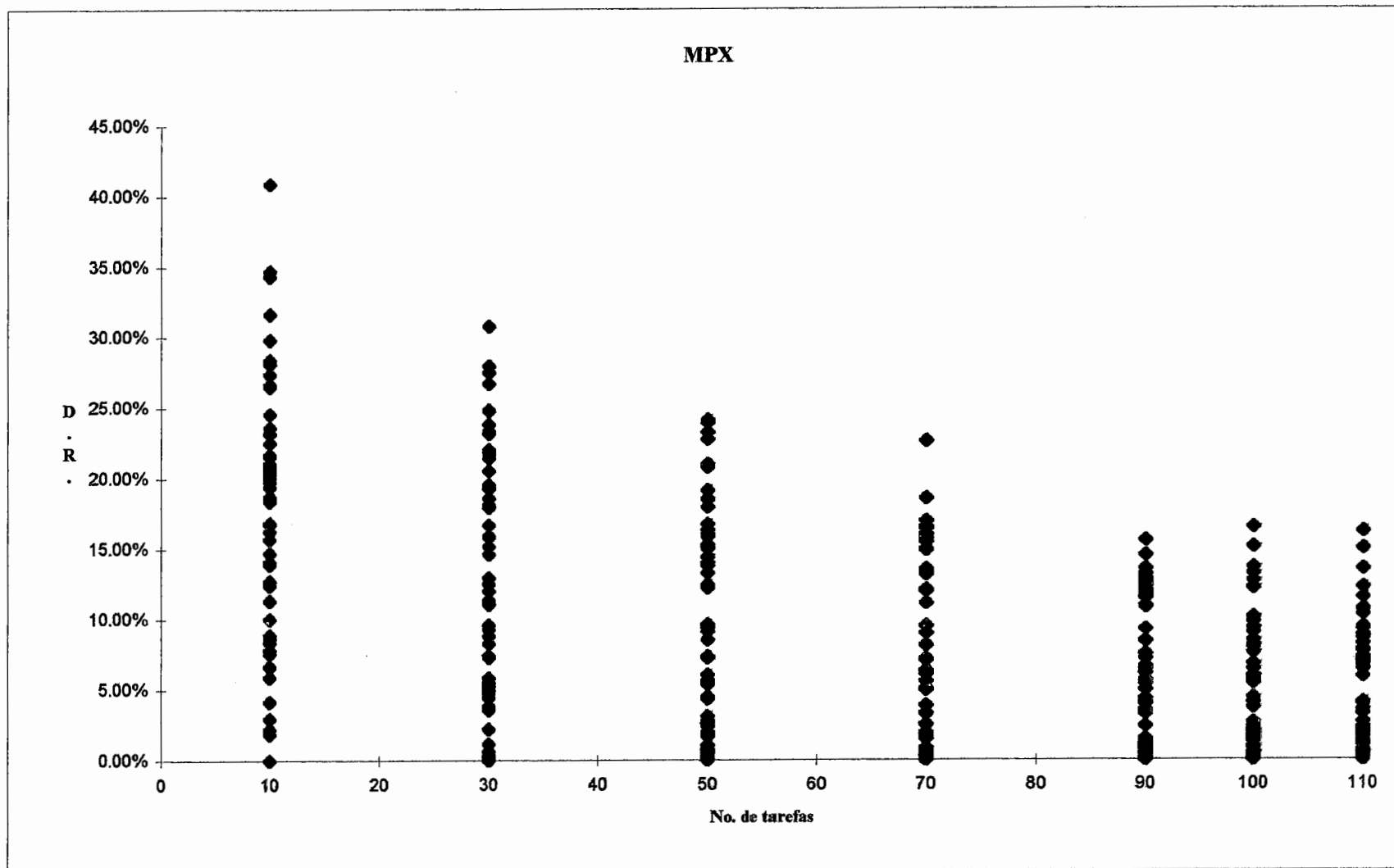


Gráfico 5.3.16: Dispersão dos desvios em relação ao número de tarefas para o operador MPX.

SEMENTES NÃO-ALEATÓRIAS

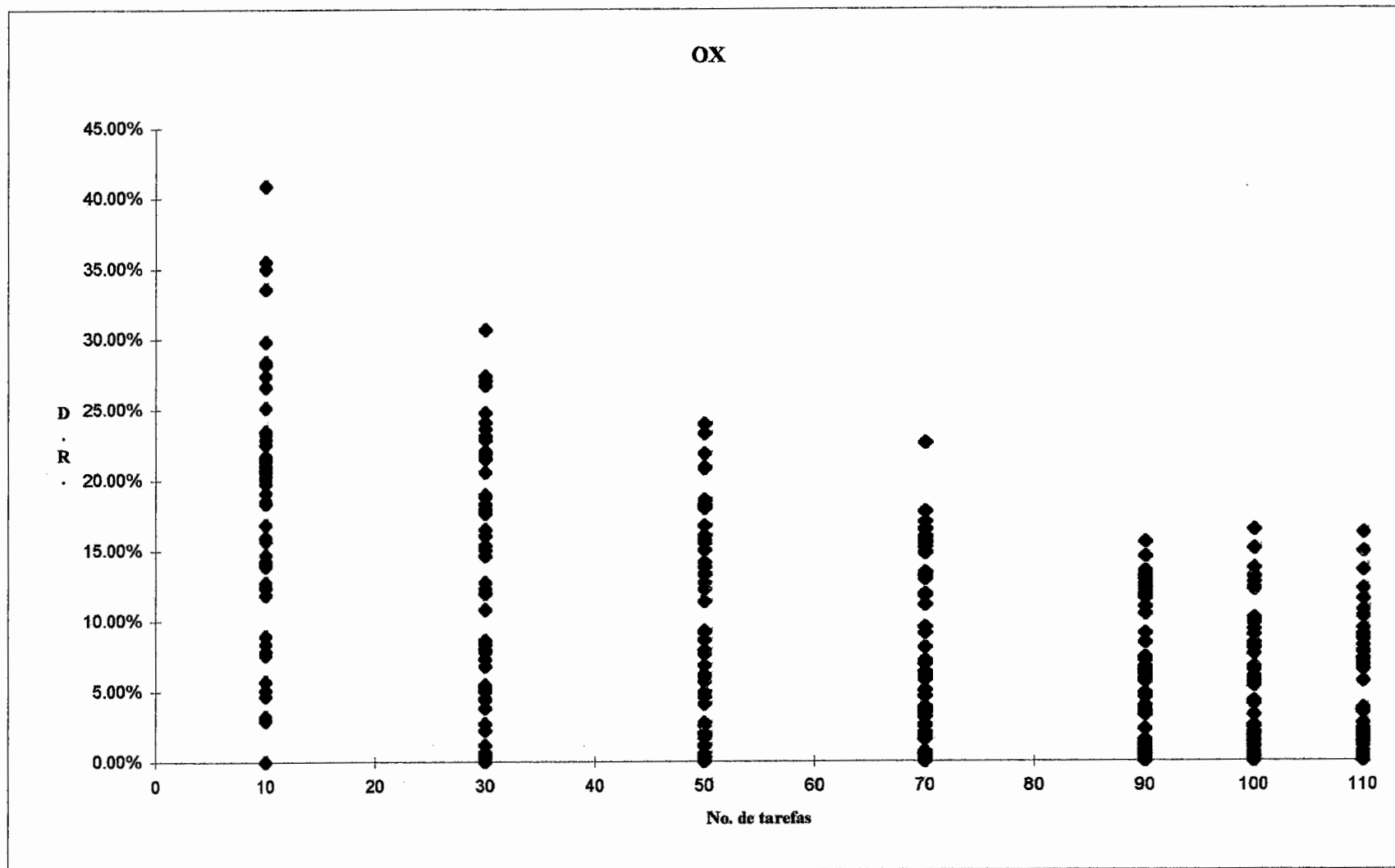


Gráfico 5.3.17: Dispersão dos desvios em relação ao número de tarefas para o operador OX.

SEMENTES NÃO-ALEATÓRIAS

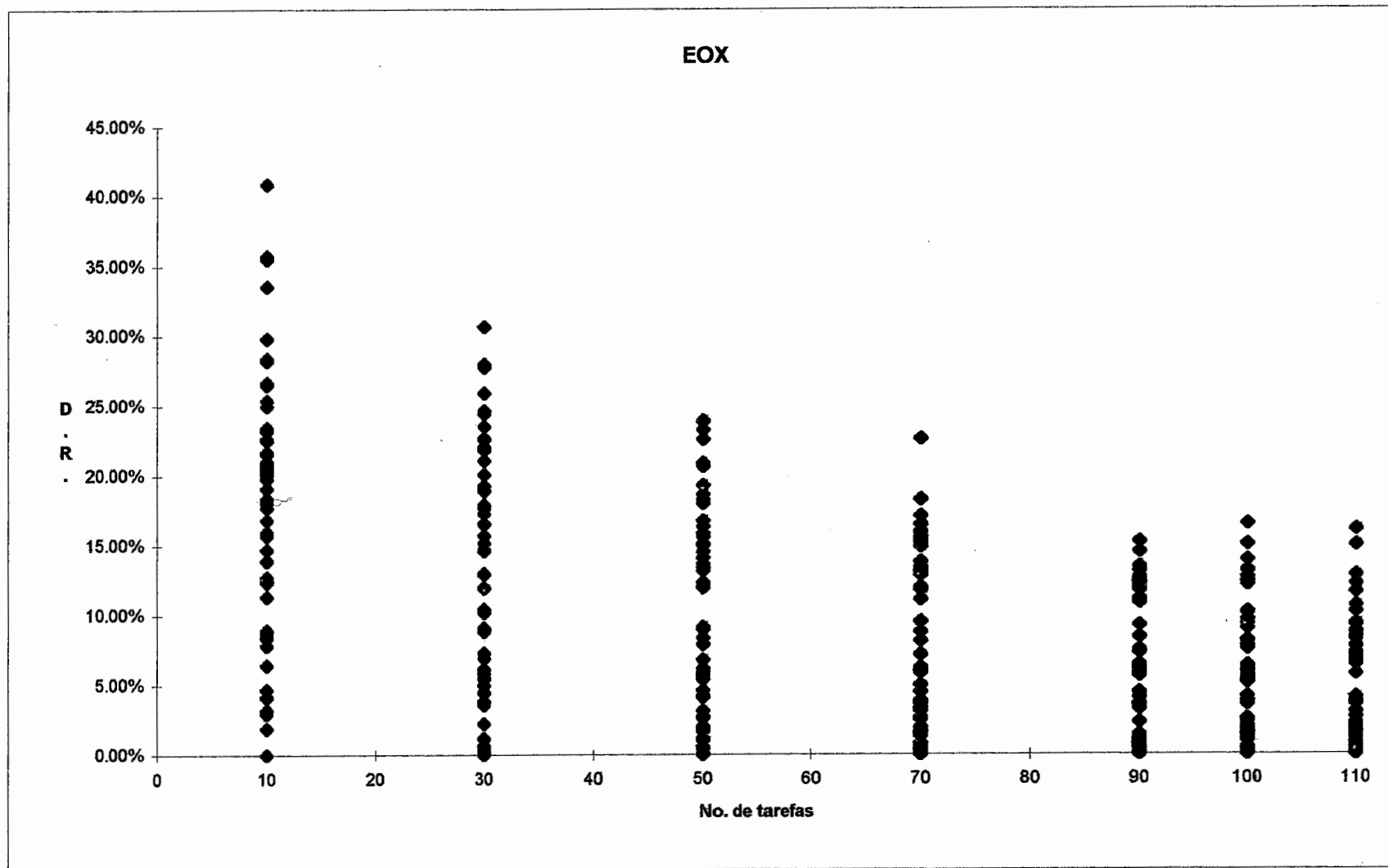


Gráfico 5.3.18: Dispersão dos desvios em relação ao número de tarefas para o operador EOX.

SEMENTES NÃO-ALEATÓRIAS

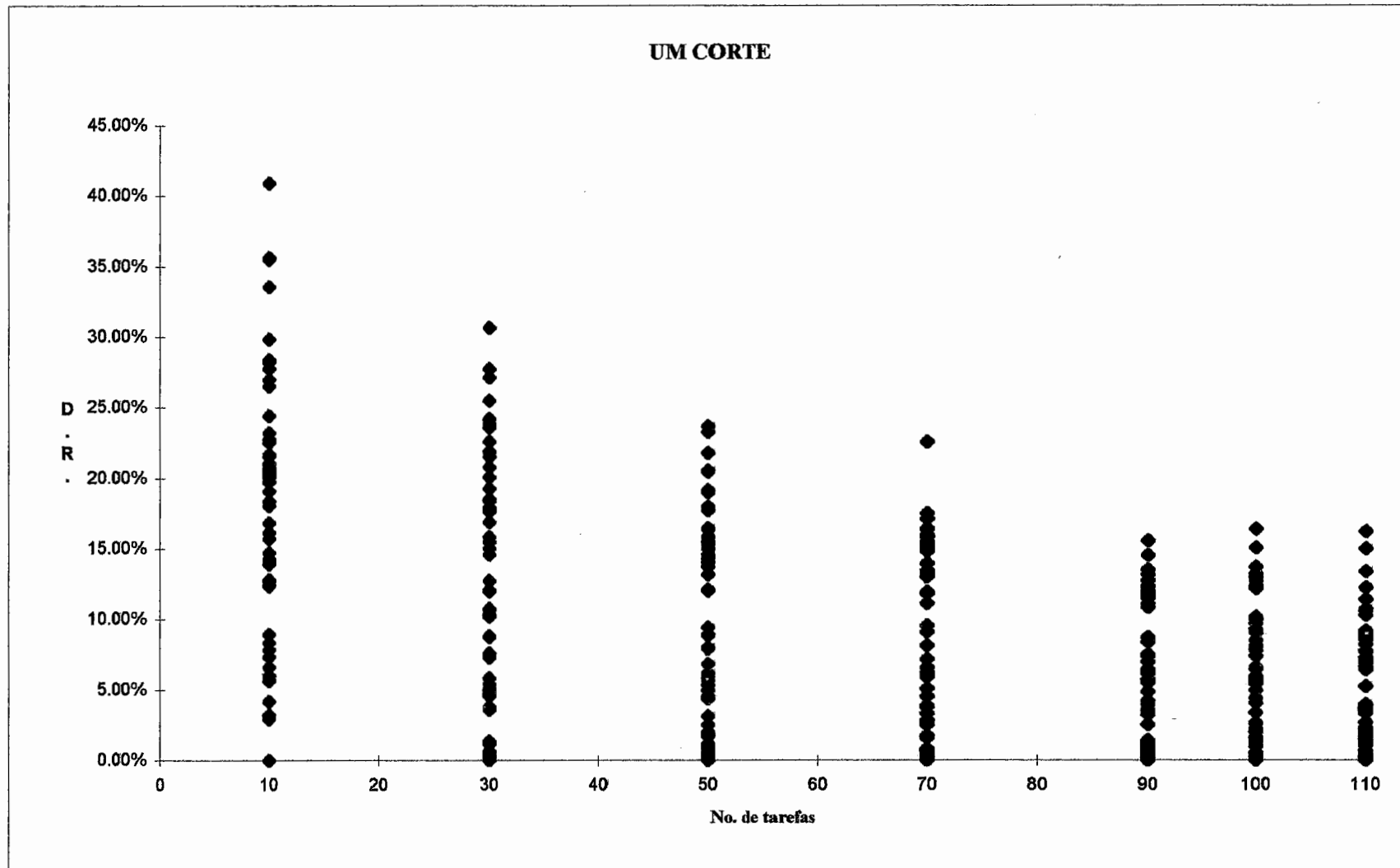


Gráfico 5.3.19: Dispersão dos desvios em relação ao número de tarefas para o operador UM CORTE.

SEMENTES NÃO-ALEATÓRIAS

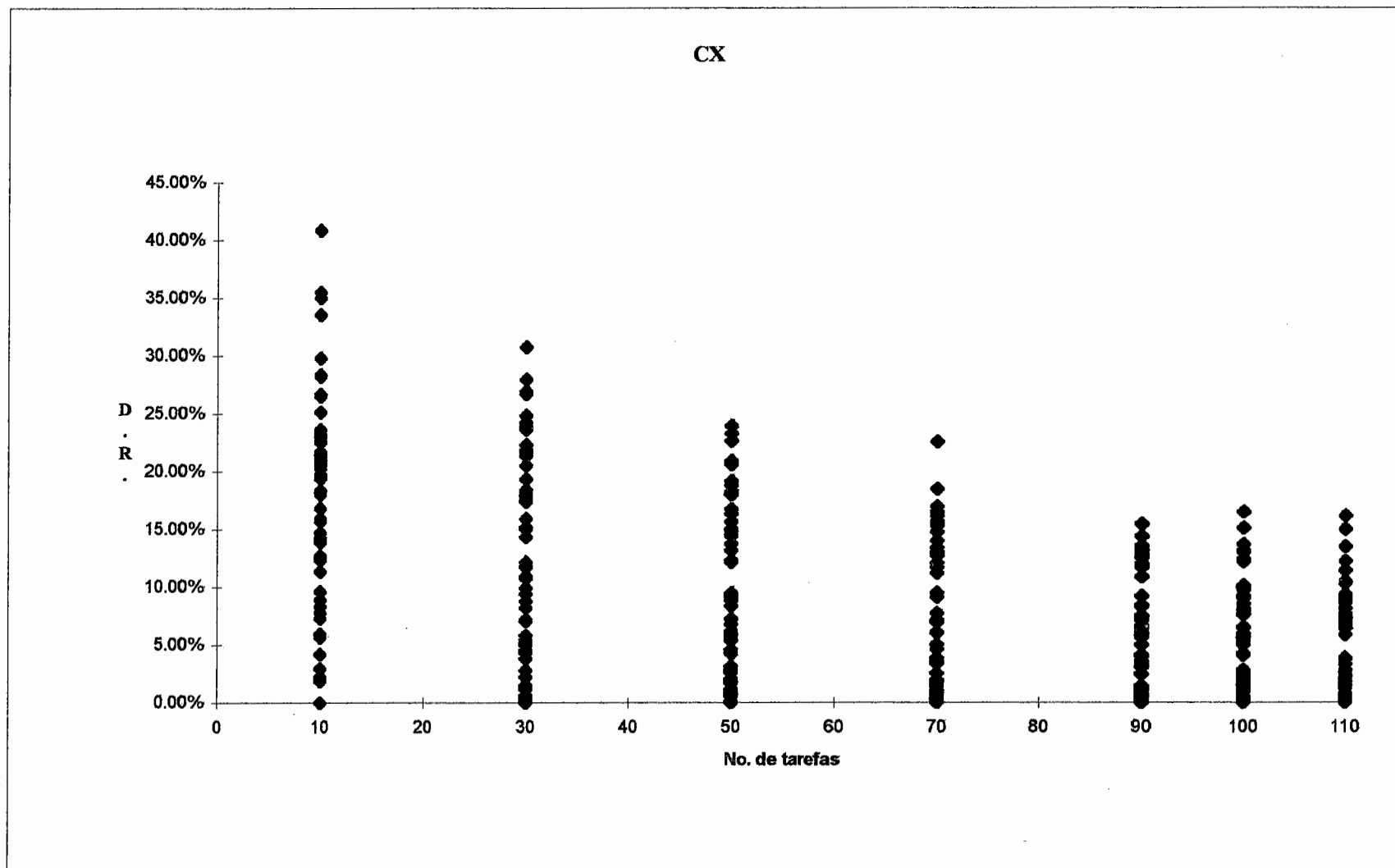


Gráfico 5.3.20: Dispersão dos desvios em relação ao número de tarefas para o operador CX.

SEMENTES NÃO-ALEATÓRIAS

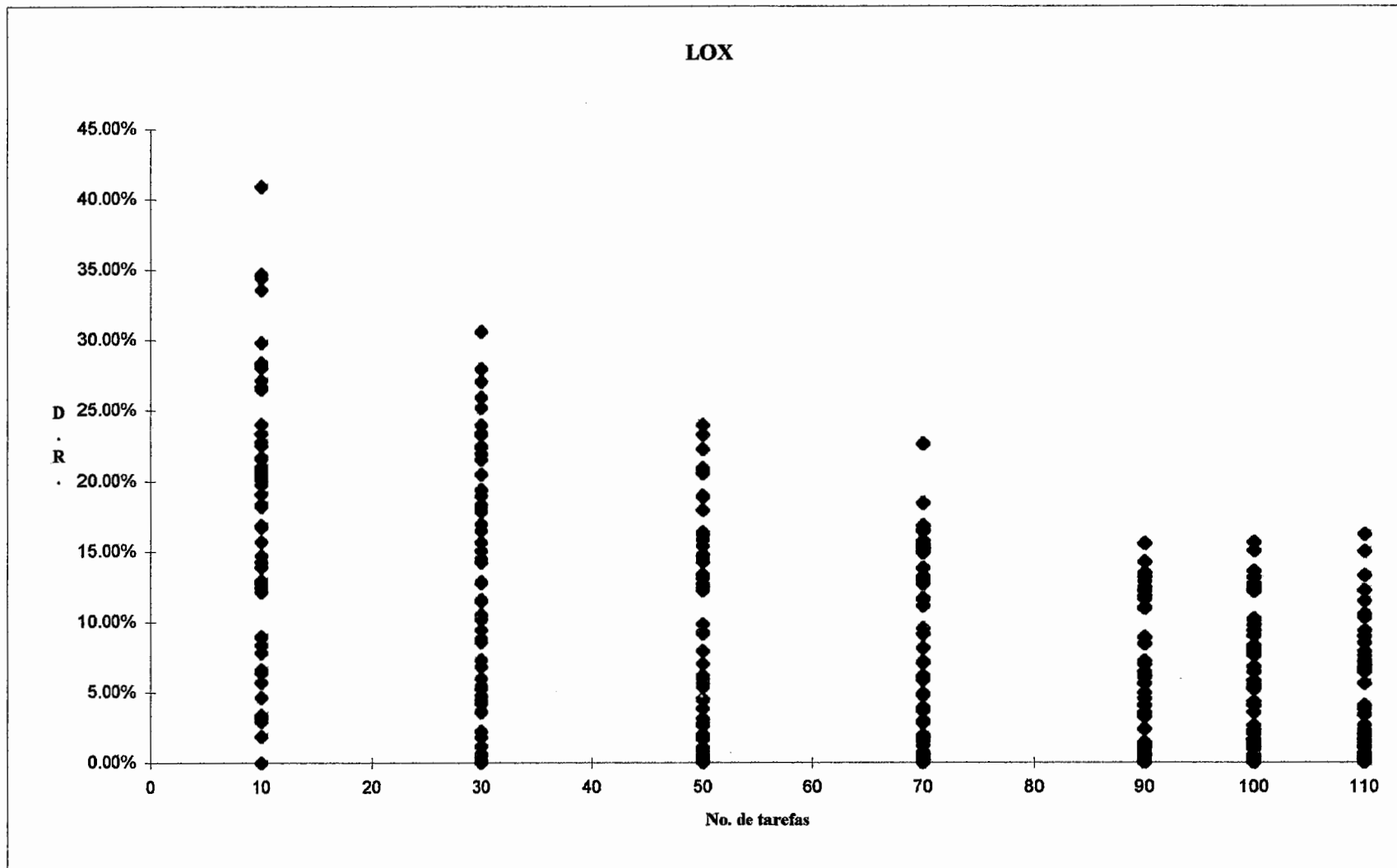


Gráfico 5.3.21: Dispersão dos desvios em relação ao número de tarefas para o operador LOX.

Pode-se observar que em todos os gráficos, os referentes as sementes não-aleatórias estão acima dos de sementes aleatórias e se parecem muito com relação a declividade.

Ainda para observar a eficiência dos algoritmos com sementes não-aleatórias, montou-se uma tabela com os problemas que não foram melhorados por nenhum dos sete algoritmos propostos (tabela 5.3.4).

Tabela No. 5.3.4: Número de problemas que nenhum algoritmo, com sementes não-aleatórias, conseguiu melhorar.

	n = 10	n = 30	n = 50	n = 70	n = 90	n = 100	n = 110
m = 4	5	6	8	6	9	10	8
m = 7	3	1	2	1	3	5	3
m = 10	8	0	1	1	2	3	5
m = 15	2	0	0	1	2	0	1
m = 20	5	0	0	0	0	0	0
m = 25	2	0	1	0	0	0	0
% parcial	36.7	11.7	20.0	16.7	30.0	30.0	30.0
PORCENTAGEM TOTAL = 24,8%							

6. CONCLUSÕES E CONSIDERAÇÕES FINAIS

Do presente trabalho várias conclusões importantes podem ser tiradas. Porém uma se destaca em relação ao uso de sementes iniciais (soluções iniciais) boas. Em sua quase totalidade os problemas resolvidos com boas sementes iniciais (não-aleatórias) tiveram um desempenho superior aos resolvidos com sementes iniciais aleatórias, reforçando a afirmação feita por JOHNSON et al (1989) para um outro algoritmo de pesquisa genérico já citado no capítulo 2, o Simulated Annealing, onde ele comprova o melhor desempenho com boas soluções iniciais. De forma global os gráficos 5.3.1 a 5.3.7 mostram bem esta afirmação.

O desempenho pior do algoritmo MPX é explicado através do operador de cruzamento utilizado. O MPX trabalha de forma a quebrar demais a seqüência (cromossomo), provocando um desarranjo muito grande e se distanciando de soluções boas.

Por outro lado os algoritmos UM CORTE e PMX apresentaram um bom desempenho pelo motivo oposto do acima. Os operadores UM CORTE e PMX trabalham de forma a não desarranjarem muito a seqüência, procurando boas soluções na vizinhança das soluções iniciais.

Quando da utilização de sementes não-aleatórias, observam-se nos problemas onde o número de máquinas é igual a $m = 4$ (Gráfico 5.3.4), que os algoritmos utilizados para encontrar as sementes iniciais (NEH e FSHOPH) na quase totalidade encontraram a solução ótima, não havendo possibilidade de melhoramento, ou seja, não havendo necessidade da presença dos algoritmos melhorativos.

Observando as médias dos desvios relativos percebe-se que com o aumento do número de tarefas esses desvios diminuem, levando-se a conclusão de que a

Dos gráficos de dispersão pode-se notar que quanto maior o número de tarefas, os algoritmos apresentam um desempenho mais estável, enquanto que para problemas de pequeno porte eles são mais sensíveis aos dados do problema, ou seja, os tempos de processamentos das operações.

Cabe observar que neste trabalho os parâmetros utilizados como número de gerações, tamanho da população, taxa de mutação e etc, tiveram como referência a utilização dessa técnica em outros tipos de problemas reportados na literatura. Caberia, em futuros trabalhos, uma análise mais apurada desses parâmetros, para o refinamento da técnica do Algoritmo Genético na resolução do problema de programação de operações flow shop permutacional. Além disso, na análise de resultados da experimentação computacional, poderiam ser utilizados outros elementos de comparação, além da porcentagem de sucesso e do desvio relativo, tal como a análise de postos ('ranking').

É apropriado também citar, que um trabalho futuro seria desenvolver novos operadores de cruzamento que levem em conta a estrutura do problema flow shop, ou seja, operadores que poderiam ser considerados específicos ou mais apropriados a esse problema. Pode-se pensar também em incorporar ao Algoritmo Genético elementos de outras técnicas como Simulated Annealing e Busca Tabu para a formação de sistemas híbridos que tenham desempenhos melhores àqueles obtidos neste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

BUFFA, E. S. (1975), Readings in Production and Questions Management. Huntington, Robert E. Krieger, 608 p.

BURBIDGE, J. L. (1983), Planejamento e Controle da Produção. Tradução: Luiz Henrique S. Cruz, São Paulo, Editora Atlas, 556p.

BIEGEL, J. E. and DAVERN, J. J. (1990), Genetic Algorithms and Job Shop Scheduling, Proceeding of the 12th Annual Conference on Computers and Industrial Engineering, Computers and industrial Engineering, Volume 19, Nos. 1 a 4, p. 81 a 91.

DANNENBRING, D. G. (1977), An Evaluation of Flow-Shop Sequencing Heutistics, Management Science, Volume 23, 1174 - 1182.

DAVIS, I. (1987), Genetic Algorithm and Simulated Annealing, London: Pitman.

CAMPBELL, H. G., DUDEK, R. A. and SMITH, M. L., A Heuristic Algorithm for n-job, m-machine Sequencing Problem, Management Science, Volume16/B, 630 - 937.

FALKENAUER, E. and BOUFFOUIX, S. (1991), A Genetic Algorithm for Job Shop, Proceedings of the 1991 IEEE, International Conference on Robotics and Automation, Sacramento, California, Abril, 824 - 829.

GOLDBERG D. E. (1989), Genetic Algorithms in Search, Optimization, and Machine Learning, Addison - Wesley, 412 p.

GORGES, S. M.(1989), ASPARGOS - An Asynchronous Parallel Genetic Optimization Strategy, Proceeding 3rd Conference on Genetic Algorithms, D. Schaffer (ed.), Morgan Kaufmann Publ., San Mateo 422 - 427.

GUPTA, J. N. D. A (1971), Functional Heuristic Algorithm for the Flow-Shop Scheduling Problem, Operational Research Quarterly, Volume 22, p.39 - 47.

HIGGINS, P., BROWNE, J. (1992), Master Production Scheduling: A Concurrent Planning Approach. Production and Planning Control, Volume 3, No. 1, p. 2 - 18.

HO, J. C. and CHANG, Y - L. (1991), A New Heuristic for the n-job, m-machine Flow-Shop Problem, European Journal of Operational Research, Volume 52, p. 194 - 202.

HOLLAND, J. H. (1975), Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, Mich.

HUNDAL, T. S. and RAJGOPAL, J. (1988), An Extension of Palmer's Heuristic for the Flow-Shop Scheduling Problem, International Journal of Production Research, Volume 26, p.1119 - 1124.

IGNALL, E. and SCHAGE, L. E. (1965), Application of Branch and Bound Technique to Some Flow-Shop Problem, Operations Research, Volume 13, p. 400 - 412.

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

JOHNSON, S. M. (1954), Optimal Two and Three Stage Production Schedules with Setup Times Included. Naval Research Logistic Quarterly, Volume 1, p. 61 - 68.

JOHNSON, D. S., ARAGON, C.R., MCGEOCH, L. A. and SCHEVON, C. (1989), Optimization by Simulated Annealing: An Experimental Evaluation: Part 1, Graph Partitioning, *Operation Research*, Volume 37, p. 865 - 891.

LAGEWEG, B. J. , LENSTRA, J. K. and RINNOOY KAN, A. H. G. (1978), A General Bounding Scheme for Permutation Flow-Shop Problem, *Operations Research*, Volume 26, p. 53 - 67.

MACCARTHY, B. L. and LIU, J. (1993), Addressing the Gap in Scheduling Research: A Review of Optimization and Heuristic in Production Scheduling. *International Journal of Production Research* , Volume 31, No. 1, p. 59 - 79.

MOCCELLIN J. V. (1993), Um Novo Método Heurístico para Solução do Problema de Programação de Operações Flow Shop Permutacional. 13o. Encontro Nacional de Engenharia de Produção, Anais, ABEPRO, p. 886 - 891.

MOCCELLIN, J. V. (1992), Uma contribuição à Programação de Operações em Sistemas de Produção Intermitente 'Flow-Shop', Tese de Livre-Docência USP- São Carlos, 126 p.

NAGANO, M. S. (1995), Novos Procedimentos de Busca Tabu para o Problema de Programação Flow-Shop Permutacional, Dissertação de Mestrado, Escola de Engenharia de São Carlos - U.S.P., 117p.

NAWAZ, M., ENSCORE Jr., E. E. Jr. and HAM (1983), A Heuristic Algorithm for the m-machine, n-job Sequencing Problem, *OMEGA*, Volume 11, p. 91-95.

OSMAN, I. H. and POTTS, C. N. (1989), Simulated Annealing for Permutation Flow-Shop Scheduling, *OMEGA*, Volume 17, 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*, Volume 16, p. 101 - 107.

PIRES, S. R. I. (1989), Planejamento e Controle da Produção em Indústrias que Utilizam Tecnologia de Grupo: Um Modelo de Sequenciamento da Produção Celular Dependente dos Tempos de Preparação de Máquinas. Tese de Doutorado pela Escola de Engenharia de São Carlos, U.S.P., São Carlos, 155p.

POTTS, C. N. (1980), An Adaptive Branching Rule for the Permutation Flow-Shop problem, European journal of operational research 5, 19 - 25.

REEVES, C. R. (1995), A Genetic Algorithm for FlowShop Sequencing, Computers Operational Research, Volume 22, No.1 , p. 5 - 13.

REEVES, C. R. (1993), Improving the Efficiency of Tabu Search for Machine Sequencing Problem, Journal of the Operational Research Society, Volume 44, p. 375 - 382.

SELEN, W. J. and HOTT, D. D. (1986), A Mixed Integer Goal Programming Formulation of the Standard Flow-Shop Scheduling Problem, Journal of Operational Research Society, Volume 37, p. 1121 - 1128.

SHANG, YI and LI, GUO-JIE (1991), New Crossover Operators in Genetic Algorithms, Proceedings of the 1991 IEEE, International Conference on Tools for AI, San Jose, Novembro, p. 150 - 153.

VASCONCELOS, M. A. (1979), Leituras Sobre Planejamento e Controle da Produção. São Paulo, EAE-FGV, (mimeo).

VENUGOPAL, V., NAREDRAN, T. T. (1992), A Genetic Algorithm Approach to the Machine-Component Grouping Problem with Multiple Objective, Computers and Industrial Engineering, Volume 22, No. 4, p. 469 a 480.

ZACCARELLI, S. B. (1986), Programação e Controle da Produção. São Paulo. Editora Pioneira.

WILSON, J. M. (1989), Alternative Formulations of Flow-Shop Scheduling Problem, Journal of Operational Research Society, Volume 40, p. 395 - 399.

WIDMER, M. and HERTZ, A. (1989), A New Heuristic Method for the Flow-Shop Sequencing Problem, European Journal of Operational Research, Volume 41, p. 186 - 193.

APÊNDICE I : PROGRAMAS COMPUTACIONAIS

UNIDADE CONTENDO OS ALGORITMOS GENÉTICOS PROPOSTOS

UNIT OPERGENE;

INTERFACE

USES WINCRT, WINDOS, WINCTV, STRINGS, WINPROCS, WINTYPES,
MAKE, SEMNEH, FSHOPH2, SEMEAL, POPMUT, OPERADOX,
VERIFICA, TIMER, APRESEN, ARQUITVAR, ABRIRARQ;

```
PROCEDURE ALGGENPMX(VAR ALEATORIA : BOOL;
                    VAR TAMPOP    : LONGINT;
                    VAR NUMGERA   : LONGINT;
                    VAR PROB      : STRING);
```

```
PROCEDURE ALGGENMPX(VAR ALEATORIA : BOOL;
                    VAR TAMPOP    : LONGINT;
                    VAR NUMGERA   : LONGINT;
                    VAR PROB      : STRING);
```

```
PROCEDURE ALGGENOX(VAR ALEATORIA : BOOL;
                   VAR TAMPOP    : LONGINT;
                   VAR NUMGERA   : LONGINT;
                   VAR PROB      : STRING);
```

```
PROCEDURE ALGGENEIX(VAR ALEATORIA : BOOL;
                    VAR TAMPOP    : LONGINT;
                    VAR NUMGERA   : LONGINT;
                    VAR PROB      : STRING);
```

```
PROCEDURE ALGGENLOX(VAR ALEATORIA : BOOL;
                    VAR TAMPOP    : LONGINT;
                    VAR NUMGERA   : LONGINT;
                    VAR PROB      : STRING);
```

```
PROCEDURE ALGGENCX(VAR ALEATORIA : BOOL;
                   VAR TAMPOP    : LONGINT;
                   VAR NUMGERA   : LONGINT;
                   VAR PROB      : STRING);
```

```
PROCEDURE ALGGENUMCORTE(VAR ALEATORIA:BOOL;
                        VAR TAMPOP :LONGINT;
                        VAR NUMGERA :LONGINT;
                        VAR PROB :STRING);
```

IMPLEMENTATION

```
VAR
  MAKEIN, MAKEFI, M1, M2, M3, M4, RESPOSTA,
  POSSOL, SEMENTE1, SEMENTE2, SEQUENCIA,
  CROMO1 : INTEGER;
  CROMO2, CROMOT1, CROMOT2, VALMAKE : ARRAN;
  TEMPOPRO : ARRMIN;
  TEMPODEC : REAL;
  VETOR : ARRAY[0..4] OF INTEGER;
```

(***** ALGORITMO USANDO PMX *****)

```
PROCEDURE ALGGENPMX(VAR ALEATORIA : BOOL;
                    VAR TAMPOP    : LONGINT;
                    VAR NUMGERA   : LONGINT;
                    VAR PROB      : STRING);
```

```
VAR
  TEXTBUFF : ARRAY[0..65] OF CHAR;
  GERACAO : INTEGER;
  OPGENE : STRING;
  RESULTADO : BOOL;
  PM : REAL;
```

```
BEGIN
  CLOCKON;
  IF ALEATORIA = TRUE THEN
    BEGIN
      SEMENTEAL(M,N,TEMPOPRO,CROMO1,CROMO2);
    END
  ELSE
    BEGIN
      FLOWSHOPH2(M,N,TEMPOPRO,CROMO1);
      NEH(M,N,TEMPOPRO,CROMO2);
    END;
    SEMENTE1 := CROMO1;
    SEMENTE2 := CROMO2;
    MAKEIN := MAKESPAN(M,N,SEMENTE1,TEMPOPRO);
    SEQUENCIA := SEMENTE1;
    IF MAKEIN > MAKESPAN(M,N,SEMENTE2,TEMPOPRO) THEN
      BEGIN
        MAKEIN := MAKESPAN(M,N,SEMENTE2,TEMPOPRO);
        SEQUENCIA := SEMENTE2;
      END;
    VALMAKE[0] := MAKEIN;
    GERACAO := 1;
    WHILE GERACAO <= NUMGERA DO
      BEGIN
        POPULACAO(M,N,TAMPOP,TEMPOPRO,CROMO1,CROMO2,M1,M2);
        OPERPMX(M,N,CROMO1,CROMO2,CROMOT1,CROMOT2);
        MUTACAO(M,N,CROMOT1,CROMOT2);
        REPLACE(M,N,TAMPOP,CROMOT1,CROMOT2,TEMPOPRO,M3,M4);
        VETOR[1]:=-M1;VETOR[2]:=-M2;VETOR[3]:=-M3;VETOR[4]:=-M4;
        FOR I := 1 TO 2 DO
          BEGIN
            FOR J := 3 TO 4 DO
              BEGIN
                IF VETOR[I] > VETOR[J] THEN
                  BEGIN
                    VETOR[0]:= VETOR[I];
                    VETOR[I]:= VETOR[J];
                    VETOR[J]:= VETOR[0];
                  END;
                END;
            END;
            IF VETOR[1] = M3 THEN CROMO1 := CROMOT1;
            IF VETOR[1] = M4 THEN CROMO1 := CROMOT2;
            IF VETOR[2] = M3 THEN CROMO2 := CROMOT1;
            IF VETOR[2] = M4 THEN CROMO2 := CROMOT2;
            IF VETOR[1] < VETOR[2] THEN
              BEGIN
                VALMAKE[GERACAO] := VETOR[1];
              END
            ELSE
              BEGIN
                VALMAKE[GERACAO] := VETOR[2];
              END;
            GERACAO := GERACAO + 1;
          END;
          M1 := MAKESPAN(M,N,CROMO1,TEMPOPRO);
          M2 := MAKESPAN(M,N,CROMO2,TEMPOPRO);
          IF M1 < M2 THEN
            BEGIN
              IF M1 < MAKEIN THEN
                BEGIN
                  SEQUENCIA := CROMO1;
                  MAKEFI := M1;
                END;
            END
          ELSE
            BEGIN
```

```

IF M2 < MAKEIN THEN
  BEGIN
    SEQUENCIA := CROMO2;
    MAKEFI := M2;
  END;
END;
CLOCKOFF(TEMPODEC);
END;
(***** ALGORITMO USANDO MPX *****)

PROCEDURE ALGGENMPX(
  VAR ALEATORIA : BOOL;
  VAR TAMPOP : LONGINT;
  VAR NUMGERA : LONGINT;
  VAR PROB : STRING);

VAR
  TEXTBUFF : ARRAY[0..65] OF CHAR;
  GERACAO : INTEGER;
  OPGENE : STRING;
  RESULTADO : BOOL;
  PM : REAL;

BEGIN
  CLOCKON;
  IF ALEATORIA = TRUE THEN
    BEGIN
      SEMENTEALE(M,N,TEMPOPRO,CROMO1,CROMO2);
    END
  ELSE
    BEGIN
      FLOWSHOPH2(M,N,TEMPOPRO,CROMO1);
      NEH(M,N,TEMPOPRO,CROMO2);
    END;
    SEMENTE1 := CROMO1;
    SEMENTE2 := CROMO2;
    MAKEIN := MAKESPAN(M,N,SEMENTE1,TEMPOPRO);
    SEQUENCIA := SEMENTE1;
    IF MAKEIN > MAKESPAN(M,N,SEMENTE2,TEMPOPRO) THEN
      BEGIN
        MAKEIN := MAKESPAN(M,N,SEMENTE2,TEMPOPRO);
        SEQUENCIA := SEMENTE2;
      END;
    VALMAKE[0] := MAKEIN;
    GERACAO := 1;
    WHILE GERACAO <= NUMGERA DO
      BEGIN
        POPULACAO(M,N,TAMPOP,TEMPOPRO,CROMO1,CROMO2,
          M1,M2);
        OPERMPX(M,N,CROMO1,CROMO2,CROMOT1,CROMOT2);
        MUTACAO(M,N,CROMOT1,CROMOT2);
        REPLACE(M,N,TAMPOP,CROMOT1,CROMOT2,TEMPOPRO,
          M3,M4);
        VETOR[1]:=M1;VETOR[2]:=M2;
        VETOR[3]:=M3;VETOR[4]:=M4;
        FOR I := 1 TO 2 DO
          BEGIN
            FOR J := 3 TO 4 DO
              BEGIN
                IF VETOR[I] > VETOR[J] THEN
                  BEGIN
                    VETOR[0]:= VETOR[I];
                    VETOR[I]:= VETOR[J];
                    VETOR[J]:= VETOR[0];
                  END;
                END;
              END;
            IF VETOR[1] = M3 THEN CROMO1 := CROMOT1;
            IF VETOR[1] = M4 THEN CROMO1 := CROMOT2;
            IF VETOR[2] = M3 THEN CROMO2 := CROMOT1;
            IF VETOR[2] = M4 THEN CROMO2 := CROMOT2;
            IF VETOR[1] < VETOR[2] THEN
              BEGIN
                VALMAKE[GERACAO] := VETOR[1];
              END
            ELSE
              BEGIN
                VALMAKE[GERACAO] := VETOR[2];
              END;
            GERACAO := GERACAO + 1;
          END;
        M1 := MAKESPAN(M,N,CROMO1,TEMPOPRO);
        M2 := MAKESPAN(M,N,CROMO2,TEMPOPRO);
        IF M1 < M2 THEN
          BEGIN
            IF M1 < MAKEIN THEN
              BEGIN
                SEQUENCIA := CROMO1;
                MAKEFI := M1;
              END;
            END
          ELSE
            BEGIN
              IF M2 < MAKEIN THEN
                BEGIN
                  SEQUENCIA := CROMO2;
                  MAKEFI := M2;
                END;
              END;
            END;
          END;
        END;
      END;
    END;
  END;

```

```

END;
CLOCKOFF(TEMPODEC);
END;
(***** ALGORITMO USANDO OX *****)

PROCEDURE ALGGENOX(VAR ALEATORIA : BOOL;
  VAR TAMPOP : LONGINT;
  VAR NUMGERA : LONGINT;
  VAR PROB : STRING);

VAR
  TEXTBUFF : ARRAY[0..65] OF CHAR;
  GERACAO : INTEGER;
  OPGENE : STRING;
  RESULTADO : BOOL;
  PM : REAL;

BEGIN
  CLOCKON;
  IF ALEATORIA = TRUE THEN
    BEGIN
      SEMENTEALE(M,N,TEMPOPRO,CROMO1,CROMO2);
    END
  ELSE
    BEGIN
      FLOWSHOPH2(M,N,TEMPOPRO,CROMO1);
      NEH(M,N,TEMPOPRO,CROMO2);
    END;
    SEMENTE1 := CROMO1;
    SEMENTE2 := CROMO2;
    MAKEIN := MAKESPAN(M,N,SEMENTE1,TEMPOPRO);
    SEQUENCIA := SEMENTE1;
    IF MAKEIN > MAKESPAN(M,N,SEMENTE2,TEMPOPRO) THEN
      BEGIN
        MAKEIN := MAKESPAN(M,N,SEMENTE2,TEMPOPRO);
        SEQUENCIA := SEMENTE2;
      END;
    VALMAKE[0] := MAKEIN;
    GERACAO := 1;
    WHILE GERACAO <= NUMGERA DO
      BEGIN
        POPULACAO(M,N,TAMPOP,TEMPOPRO,CROMO1,CROMO2,
          M1,M2);
        OPEROX(M,N,CROMO1,CROMO2,CROMOT1,CROMOT2);
        MUTACAO(M,N,CROMOT1,CROMOT2);
        REPLACE(M,N,TAMPOP,CROMOT1,CROMOT2,TEMPOPRO,
          M3,M4);
        VETOR[1]:=M1;VETOR[2]:=M2;VETOR[3]:=M3;
        VETOR[4]:=M4;
        FOR I := 1 TO 2 DO
          BEGIN
            FOR J := 3 TO 4 DO
              BEGIN
                IF VETOR[I] > VETOR[J] THEN
                  BEGIN
                    VETOR[0]:= VETOR[I];
                    VETOR[I]:= VETOR[J];
                    VETOR[J]:= VETOR[0];
                  END;
                END;
              END;
            IF VETOR[1] = M3 THEN CROMO1 := CROMOT1;
            IF VETOR[1] = M4 THEN CROMO1 := CROMOT2;
            IF VETOR[2] = M3 THEN CROMO2 := CROMOT1;
            IF VETOR[2] = M4 THEN CROMO2 := CROMOT2;
            IF VETOR[1] < VETOR[2] THEN
              BEGIN
                VALMAKE[GERACAO] := VETOR[1];
              END
            ELSE
              BEGIN
                VALMAKE[GERACAO] := VETOR[2];
              END;
            GERACAO := GERACAO + 1;
          END;
        M1 := MAKESPAN(M,N,CROMO1,TEMPOPRO);
        M2 := MAKESPAN(M,N,CROMO2,TEMPOPRO);
        IF M1 < M2 THEN
          BEGIN
            IF M1 < MAKEIN THEN
              BEGIN
                SEQUENCIA := CROMO1;
                MAKEFI := M1;
              END;
            END
          ELSE
            BEGIN
              IF M2 < MAKEIN THEN
                BEGIN
                  SEQUENCIA := CROMO2;
                  MAKEFI := M2;
                END;
              END;
            END;
          END;
        END;
      END;
    END;
  END;

```

CLOCKOFF(TEMPODEC);
END;

(* ***** ALGORITMO USANDO EOX ***** *)

PROCEDURE ALGGENEOX(VAR ALEATORIA : BOOL;
VAR TAMPOP : LONGINT;
VAR NUMGERA : LONGINT;
VAR PROB : STRING);

VAR

TEXTBUFF : ARRAY[0..65] OF CHAR;
GERACAO : INTEGER;
OPGENE : STRING;
RESULTADO : BOOL;
PM : REAL;

BEGIN

CLOCKON;

IF ALEATORIA = TRUE THEN

BEGIN

SEMENTEAL(M,N,TEMPOPRO,CROMO1,CROMO2);

END

ELSE

BEGIN

FLOWSHOP2(M,N,TEMPOPRO,CROMO1);

NEH(M,N,TEMPOPRO,CROMO2);

END;

SEMENTE1 := CROMO1;

SEMENTE2 := CROMO2;

MAKEIN := MAKESPAN(M,N,SEMENTE1,TEMPOPRO);

SEQUENCIA := SEMENTE1;

IF MAKEIN > MAKESPAN(M,N,SEMENTE2,TEMPOPRO) THEN

BEGIN

MAKEIN := MAKESPAN(M,N,SEMENTE2,TEMPOPRO);

SEQUENCIA := SEMENTE2;

END;

VALMAKE[0] := MAKEIN;

GERACAO := 1;

WHILE GERACAO <= NUMGERA DO

BEGIN

POPULACAO(M,N,TAMPOP,TEMPOPRO,CROMO1,CROMO2,
M1,M2);

OPEREOR(M,N,CROMO1,CROMO2,CROMOT1,CROMOT2);

MUTACAO(M,N,CROMOT1,CROMOT2);

REPLACE(M,N,TAMPOP,CROMOT1,CROMOT2,TEMPOPRO,
M3,M4);

VETOR[1]:=M1;VETOR[2]:=M2;

VETOR[3]:=M3;VETOR[4]:=M4;

FOR I := 1 TO 2 DO

BEGIN

FOR J := 3 TO 4 DO

BEGIN

IF VETOR[I] > VETOR[J] THEN

BEGIN

VETOR[0]:=VETOR[I];

VETOR[I]:=VETOR[J];

VETOR[J]:=VETOR[0];

END;

END;

END;

IF VETOR[1] = M3 THEN CROMO1 := CROMOT1;

IF VETOR[1] = M4 THEN CROMO1 := CROMOT2;

IF VETOR[2] = M3 THEN CROMO2 := CROMOT1;

IF VETOR[2] = M4 THEN CROMO2 := CROMOT2;

IF VETOR[1] < VETOR[2] THEN

BEGIN

VALMAKE[GERACAO] := VETOR[1];

END

ELSE

BEGIN

VALMAKE[GERACAO] := VETOR[2];

END;

GERACAO := GERACAO + 1;

END;

M1 := MAKESPAN(M,N,CROMO1,TEMPOPRO);

M2 := MAKESPAN(M,N,CROMO2,TEMPOPRO);

IF M1 < M2 THEN

BEGIN

IF M1 < MAKEIN THEN

BEGIN

SEQUENCIA := CROMO1;

MAKEFI := M1;

END;

END

ELSE

BEGIN

IF M2 < MAKEIN THEN

BEGIN

SEQUENCIA := CROMO2;

MAKEFI := M2;

END;

END;

CLOCKOFF(TEMPODEC);

END;

(* ***** ALGORITMO USANDO UMCORTE ***** *)

PROCEDURE ALGGENUMCORTE(VAR ALEATORIA:BOOL;
VAR TAMPOP :LONGINT;
VAR NUMGERA :LONGINT;
VAR PROB :STRING);

VAR

TEXTBUFF : ARRAY[0..65] OF CHAR;
GERACAO : INTEGER;
OPGENE : STRING;
RESULTADO : BOOL;
PM : REAL;

BEGIN

CLOCKON;

IF ALEATORIA = TRUE THEN

BEGIN

SEMENTEAL(M,N,TEMPOPRO,CROMO1,CROMO2);

END

ELSE

BEGIN

FLOWSHOP2(M,N,TEMPOPRO,CROMO1);

NEH(M,N,TEMPOPRO,CROMO2);

END;

SEMENTE1 := CROMO1;

SEMENTE2 := CROMO2;

MAKEIN := MAKESPAN(M,N,SEMENTE1,TEMPOPRO);

SEQUENCIA := SEMENTE1;

IF MAKEIN > MAKESPAN(M,N,SEMENTE2,TEMPOPRO) THEN

BEGIN

MAKEIN := MAKESPAN(M,N,SEMENTE2,TEMPOPRO);

SEQUENCIA := SEMENTE2;

END;

VALMAKE[0] := MAKEIN;

GERACAO := 1;

WHILE GERACAO <= NUMGERA DO

BEGIN

POPULACAO(M,N,TAMPOP,TEMPOPRO,CROMO1,CROMO2,
M1,M2);

OPERMUCORTE(M,N,CROMO1,CROMO2,CROMOT1,CROMOT2);

MUTACAO(M,N,CROMOT1,CROMOT2);

REPLACE(M,N,TAMPOP,CROMOT1,CROMOT2,TEMPOPRO,
M3,M4);

VETOR[1]:=M1;VETOR[2]:=M2;

VETOR[3]:=M3;VETOR[4]:=M4;

FOR I := 1 TO 2 DO

BEGIN

FOR J := 3 TO 4 DO

BEGIN

IF VETOR[I] > VETOR[J] THEN

BEGIN

VETOR[0]:=VETOR[I];

VETOR[I]:=VETOR[J];

VETOR[J]:=VETOR[0];

END;

END;

END;

IF VETOR[1] = M3 THEN CROMO1 := CROMOT1;

IF VETOR[1] = M4 THEN CROMO1 := CROMOT2;

IF VETOR[2] = M3 THEN CROMO2 := CROMOT1;

IF VETOR[2] = M4 THEN CROMO2 := CROMOT2;

IF VETOR[1] < VETOR[2] THEN

BEGIN

VALMAKE[GERACAO] := VETOR[1];

END

ELSE

BEGIN

VALMAKE[GERACAO] := VETOR[2];

END;

GERACAO := GERACAO + 1;

END;

M1 := MAKESPAN(M,N,CROMO1,TEMPOPRO);

M2 := MAKESPAN(M,N,CROMO2,TEMPOPRO);

IF M1 < M2 THEN

BEGIN

IF M1 < MAKEIN THEN

BEGIN

SEQUENCIA := CROMO1;

MAKEFI := M1;

END;

END

ELSE

BEGIN

IF M2 < MAKEIN THEN

BEGIN

SEQUENCIA := CROMO2;

MAKEFI := M2;

END;

END;

CLOCKOFF(TEMPODEC);

END;

(***** ALGORITMO USANDO CX *****)

```

PROCEDURE ALGGENCX(VAR ALEATORIA : BOOL;
                  VAR TAMPOP : LONGINT;
                  VAR NUMGERA : LONGINT;
                  VAR PROB : STRING);

VAR
  TEXTBUFF : ARRAY[0..65] OF CHAR;
  GERACAO : INTEGER;
  OPGENE : STRING;
  RESULTADO : BOOL;
  PM : REAL;

BEGIN
  CLOCKON;
  IF ALEATORIA = TRUE THEN
    BEGIN
      SEMENTEALE(M,N,TEMPOPRO,CROMO1,CROMO2);
    END
  ELSE
    BEGIN
      FLOWSHOPH2(M,N,TEMPOPRO,CROMO1);
      NEH(M,N,TEMPOPRO,CROMO2);
    END;
  SEMENTE1 := CROMO1;
  SEMENTE2 := CROMO2;
  MAKEIN := MAKESPAN(M,N,SEMENTE1,TEMPOPRO);
  SEQUENCIA := SEMENTE1;
  IF MAKEIN > MAKESPAN(M,N,SEMENTE2,TEMPOPRO) THEN
    BEGIN
      MAKEIN := MAKESPAN(M,N,SEMENTE2,TEMPOPRO);
      SEQUENCIA := SEMENTE2;
    END;
  VALMAKE[0] := MAKEIN;
  GERACAO := 1;
  WHILE GERACAO <= NUMGERA DO
    BEGIN
      POPULACAO(M,N,TAMPOP,TEMPOPRO,CROMO1,CROMO2,
        M1,M2);
      OPERCX(M,N,CROMO1,CROMO2,CROMOT1,CROMOT2);
      MUTACAO(M,N,CROMOT1,CROMOT2);
      REPLACE(M,N,TAMPOP,CROMOT1,CROMOT2,TEMPOPRO,
        M3,M4);
      VETOR[1]:=M1;VETOR[2]:=M2;
      VETOR[3]:=M3;VETOR[4]:=M4;
      FOR I := 1 TO 2 DO
        BEGIN
          FOR J := 3 TO 4 DO
            BEGIN
              IF VETOR[I] > VETOR[J] THEN
                BEGIN
                  VETOR[0]:= VETOR[I];
                  VETOR[I]:= VETOR[J];
                  VETOR[J]:= VETOR[0];
                END;
            END;
          IF VETOR[1] = M3 THEN CROMO1 := CROMOT1;
          IF VETOR[1] = M4 THEN CROMO1 := CROMOT2;
          IF VETOR[2] = M3 THEN CROMO2 := CROMOT1;
          IF VETOR[2] = M4 THEN CROMO2 := CROMOT2;
          IF VETOR[1] < VETOR[2] THEN
            BEGIN
              VALMAKE[GERACAO] := VETOR[1];
            END
          ELSE
            BEGIN
              VALMAKE[GERACAO] := VETOR[2];
            END;
          GERACAO := GERACAO + 1;
        END;
      M1 := MAKESPAN(M,N,CROMO1,TEMPOPRO);
      M2 := MAKESPAN(M,N,CROMO2,TEMPOPRO);
      IF M1 < M2 THEN
        BEGIN
          IF M1 < MAKEIN THEN
            BEGIN
              SEQUENCIA = CROMO1;
              MAKEFI := M1;
            END;
          END
        ELSE
          BEGIN
            IF M2 < MAKEIN THEN
              BEGIN
                SEQUENCIA = CROMO2;
                MAKEFI := M2;
              END;
            END;
          END;
      CLOCKOFF(TEMPODEC);
    END;
  END;

```

(***** ALGORITMO USANDO LOX *****)

```

PROCEDURE ALGGENLOX(VAR ALEATORIA : BOOL;
                   VAR TAMPOP : LONGINT;
                   VAR NUMGERA : LONGINT;
                   VAR PROB : STRING);

VAR
  TEXTBUFF : ARRAY[0..65] OF CHAR;
  GERACAO : INTEGER;
  OPGENE : STRING;
  RESULTADO : BOOL;
  PM : REAL;

BEGIN
  CLOCKON;
  IF ALEATORIA = TRUE THEN
    BEGIN
      SEMENTEALE(M,N,TEMPOPRO,CROMO1,CROMO2);
    END
  ELSE
    BEGIN
      FLOWSHOPH2(M,N,TEMPOPRO,CROMO1);
      NEH(M,N,TEMPOPRO,CROMO2);
    END;
  SEMENTE1 := CROMO1;
  SEMENTE2 := CROMO2;
  MAKEIN := MAKESPAN(M,N,SEMENTE1,TEMPOPRO);
  SEQUENCIA := SEMENTE1;
  IF MAKEIN > MAKESPAN(M,N,SEMENTE2,TEMPOPRO) THEN
    BEGIN
      MAKEIN := MAKESPAN(M,N,SEMENTE2,TEMPOPRO);
      SEQUENCIA := SEMENTE2;
    END;
  VALMAKE[0] := MAKEIN;
  GERACAO := 1;
  WHILE GERACAO <= NUMGERA DO
    BEGIN
      POPULACAO(M,N,TAMPOP,TEMPOPRO,CROMO1,CROMO2,
        M1,M2);
      OPERLOX(M,N,CROMO1,CROMO2,CROMOT1,CROMOT2);
      MUTACAO(M,N,CROMOT1,CROMOT2);
      REPLACE(M,N,TAMPOP,CROMOT1,CROMOT2,TEMPOPRO,
        M3,M4);
      VETOR[1]:=M1;VETOR[2]:=M2;
      VETOR[3]:=M3;VETOR[4]:=M4;
      FOR I := 1 TO 2 DO
        BEGIN
          FOR J := 3 TO 4 DO
            BEGIN
              IF VETOR[I] > VETOR[J] THEN
                BEGIN
                  VETOR[0]:= VETOR[I];
                  VETOR[I]:= VETOR[J];
                  VETOR[J]:= VETOR[0];
                END;
            END;
          IF VETOR[1] = M3 THEN CROMO1 := CROMOT1;
          IF VETOR[1] = M4 THEN CROMO1 := CROMOT2;
          IF VETOR[2] = M3 THEN CROMO2 := CROMOT1;
          IF VETOR[2] = M4 THEN CROMO2 := CROMOT2;
          IF VETOR[1] < VETOR[2] THEN
            BEGIN
              VALMAKE[GERACAO] := VETOR[1];
            END
          ELSE
            BEGIN
              VALMAKE[GERACAO] := VETOR[2];
            END;
          GERACAO := GERACAO + 1;
        END;
      M1 := MAKESPAN(M,N,CROMO1,TEMPOPRO);
      M2 := MAKESPAN(M,N,CROMO2,TEMPOPRO);
      IF M1 < M2 THEN
        BEGIN
          IF M1 < MAKEIN THEN
            BEGIN
              SEQUENCIA = CROMO1;
              MAKEFI := M1;
            END;
          END
        ELSE
          BEGIN
            IF M2 < MAKEIN THEN
              BEGIN
                SEQUENCIA = CROMO2;
                MAKEFI := M2;
              END;
            END;
          END;
      CLOCKOFF(TEMPODEC);
    END;
  END.

```

UNIDADE
CONTENDO
O PROGRAMA
FSHOPH2(MOCCELLIN, 1992)

```
UNIT FSHOPH2;
INTERFACE
USES WINCRT, WINDOS, STRINGS, WINPROCS, WINTYPES, WINCTV;
```

```
PROCEDURE FLOWSHOPH2(
VAR M,N : INTEGER;
VAR P : ARRMM;
VAR SEQUENCE : ARRNN);
```

IMPLEMENTATION

```
PROCEDURE FLOWSHOPH2(
VAR M,N : INTEGER;
VAR P : ARRMM;
VAR SEQUENCE : ARRNN);
```

```
TYPE
PARRNN = ^ARRNN;
```

```
VAR
LK,X : INTEGER;
UBG : PARRNN;
```

```
PROCEDURE UBGU( VAR M,N : INTEGER;
VAR P : ARRMM;
VAR UBG : PARRNN);
```

```
VAR
K,L,J : INTEGER;
DK : PARRNN;
```

```
PROCEDURE DKU( VAR K : INTEGER;
VAR P : ARRMM;
VAR DK : PARRNN);
```

```
VAR
L,J : INTEGER;
```

```
BEGIN
FOR I:=0 TO N DO
BEGIN
FOR J:=0 TO N DO
BEGIN
IF I=J THEN
BEGIN
DK[L,J] := INF;
END
ELSE
BEGIN
DK[L,J] := P[K,J] - P[K+1,J];
END;
END;
END;
END;
```

```
BEGIN (* UBGU *)
```

```
NEW(DK);
K := 1;
DKU(K,P,DK);
FOR I:=0 TO N DO
BEGIN
FOR J:=0 TO N DO
BEGIN
UBG[L,J] := DK[L,J];
IF UBG[L,J] < 0 THEN
BEGIN
UBG[L,J] := 0;
END;
END;
END;
END;
IF M > 2 THEN
BEGIN
FOR K:=2 TO (M-1) DO
BEGIN
DKU(K,P,DK);
FOR I:=0 TO N DO
BEGIN
```

```
FOR J:=0 TO N DO
BEGIN
IF I=J THEN
BEGIN
UBG[L,J] := INF;
END
ELSE
BEGIN
UBG[L,J] := UBG[L,J] + DK[L,J];
IF UBG[L,J] < 0 THEN
BEGIN
UBG[L,J] := 0;
END;
END;
END;
END;
END;
DISPOSE(DK);
END; (* UBGU *)
```

```
PROCEDURE FITSP( S : INTEGER;
VAR N : INTEGER;
VAR ROUTE : ARRNN;
VAR W : PARRNN);
```

```
VAR
END1, END2, FARTHEST, I, INDEX, INSCOST, J,
MAXDIST, NEWCOST, NEXTINDEX : INTEGER;
CYCLE, DIST : ARRNN;
```

```
BEGIN (* FITSP *)
FOR I := 0 TO N DO CYCLE[I] := -1;
CYCLE[S] := S;
W[S,S] := 0;
FOR I := 0 TO N DO
BEGIN
DIST[I] := W[S,I];
END;
```

```
I := 1;
REPEAT
MAXDIST := -INF;
FOR J := 0 TO N DO
BEGIN
IF CYCLE[J] = -1 THEN
BEGIN
IF DIST[J] > MAXDIST THEN
BEGIN
MAXDIST := DIST[J];
FARTHEST := J;
END;
END;
END;
INSCOST := INF;
INDEX := S;
FOR J := 1 TO I DO
BEGIN
NEXTINDEX := CYCLE[INDEX];
NEWCOST := W[INDEX,FARTHEST] +
W[FARTHEST,NEXTINDEX] -
W[INDEX,NEXTINDEX];
IF NEWCOST < INSCOST THEN
BEGIN
INSCOST := NEWCOST;
END1 := INDEX;
END2 := NEXTINDEX;
END;
INDEX := NEXTINDEX;
END;
CYCLE[FARTHEST] := END2;
CYCLE[END1] := FARTHEST;
```

```
IF I < N THEN
BEGIN
FOR J := 0 TO N DO
BEGIN
IF CYCLE[J] = -1 THEN
BEGIN
IF W[FARTHEST,J] < DIST[J] THEN
BEGIN
DIST[J] := W[FARTHEST,J];
END;
END;
END;
END;
I := I + 1;
UNTIL (I = N + 1);
```

```

INDEX := S ;
FOR I := 0 TO N DO
  BEGIN
    ROUTE[I] := INDEX ;
    INDEX := CYCLE[INDEX]
  END;
END; (* FITSP *)

FUNCTION MAKESPAN2(VAR M,N : INTEGER;
                  VAR S : ARR;
                  VAR P : ARR;): INTEGER;

VAR
  U,V : INTEGER;
  C : ARR;

BEGIN
  C[1,1] := P[1,S[1]];
  FOR V:=2 TO N DO
    BEGIN
      C[1,V] := C[1,V-1] + P[1,S[V]];
    END;
  FOR U:=2 TO M DO
    BEGIN
      C[U,2] := C[U-1,2] + P[U,S[2]];
      FOR V:=2 TO N DO
        BEGIN
          IF C[U-1,V] >= C[U,V-1] THEN
            BEGIN
              C[U,V] := C[U-1,V] + P[U,S[V]];
            END
          ELSE
            BEGIN
              C[U,V] := C[U,V-1] + P[U,S[V]];
            END;
          END;
        END;
      END;
    MAKESPAN2 := C[M,N]
  END;

BEGIN (* FLOWSHOP2 *)
  NEW(UBG);
  FOR I := 0 TO N DO SEQUENCE[I] := 0;
  UBGU(M,N,P,UBG);
  FITSP(0,N,SEQUENCE,UBG);
  DISPOSE(UBG);
END; (* FLOWSHOP2 *)
END.

```

UNIDADE
CONTENDO
O PROGRAMA
NEH(MOCCELLIN,1992)

```

UNIT SEMNEH;

INTERFACE

USES
WINCRT, WINDOS, WINCTV, WINTYPES, STRINGS, WINPROCS, TIMER;

PROCEDURE NEH(
    VAR M,N      : INTEGER;
    VAR P        : ARRNM;
    VAR SEQUENCE : ARRNM);

IMPLEMENTATION

PROCEDURE NEH(
    VAR M,N      : INTEGER;
    VAR P        : ARRNM;
    VAR SEQUENCE : ARRNM);

VAR
    I, K      : INTEGER;
    LPTSEQ    : ARRNM;

FUNCTION MAKESPAN1(
    VAR M,NT : INTEGER;
    VAR S    : ARRNM;
    VAR P    : ARRNM) : INTEGER;

VAR
    U,V : INTEGER;
    C    : ARRNM;

BEGIN
    C[1,1] := P[1,S[1]];
    FOR V:=2 TO NT DO
        BEGIN
            C[1,V] := C[1,V-1] + P[1,S[V]];
        END;
    FOR U:=2 TO M DO
        BEGIN
            C[U,1] := C[U-1,1] + P[U,S[1]];
            FOR V:=2 TO NT DO
                BEGIN
                    IF C[U-1,V] >= C[U,V-1] THEN
                        BEGIN
                            C[U,V] := C[U-1,V] + P[U,S[V]];
                        END
                    ELSE
                        BEGIN
                            C[U,V] := C[U,V-1] + P[U,S[V]];
                        END;
                END;
            END;
        END;
    MAKESPAN1 := C[M,NT];
END;

PROCEDURE LPTJOBSORT(VAR LPTSEQ : ARRNM);

TYPE
    TPJOB = RECORD
        INDEX, TP : INTEGER;
    END;
    TPLIST = ARRAY[0..140] OF TPJOB;

VAR
    I, U, V : INTEGER;
    TPL     : TPLIST;

PROCEDURE QUICKSORT(START, FINISH : INTEGER);

VAR
    BEFORE, AFTER, MIDPOINT : INTEGER;
    AUX                    : TPJOB;

BEGIN
    (* QUICKSORT *)
    BEFORE := START;
    AFTER  := FINISH;
    MIDPOINT := TPL[ROUND((START+FINISH) DIV 2)].TP;

```

```

REPEAT
    WHILE TPL[BEFORE].TP > MIDPOINT DO
        BEGIN
            BEFORE := BEFORE+1;
        END;
    WHILE MIDPOINT > TPL[AFTER].TP DO
        BEGIN
            AFTER := AFTER-1;
        END;
    IF BEFORE <= AFTER THEN
        BEGIN
            AUX := TPL[BEFORE];
            TPL[BEFORE] := TPL[AFTER];
            TPL[AFTER] := AUX;
            BEFORE := BEFORE+1;
            AFTER := AFTER-1;
        END;
    UNTIL (BEFORE > AFTER);
    IF START < AFTER THEN
        BEGIN
            QUICKSORT(START, AFTER);
        END;
    IF BEFORE < FINISH THEN
        BEGIN
            QUICKSORT(BEFORE, FINISH);
        END;
    END;
    (* QUICKSORT *)

BEGIN
    (* LPTJOBSORT *)
    FOR V := 0 TO N DO
        BEGIN
            TPL[V].INDEX := V; TPL[V].TP := 0;
            FOR U := 1 TO M DO
                BEGIN
                    TPL[V].TP := TPL[V].TP + P[U,V];
                END;
            END;
            QUICKSORT(1,N);
        END;
    FOR I := 0 TO N DO
        BEGIN
            LPTSEQ[I] := TPL[I].INDEX;
        END;
    END;
    (* LPTJOBSORT *)

PROCEDURE NEHJOBINSERTIONSEQUENCE(
    VAR N      : INTEGER;
    VAR S, SEQUENCE : ARRNM);

VAR
    I, J, K, Z, STARTN : INTEGER;
    STARTD, INSD, NEWD : INTEGER;
    INDEX, NEXTINDEX, INSJOB : INTEGER;
    END1, END2, NBJOB : INTEGER;
    STARTSEQ, PARTIALSEQ : ARRNM;
    PARTIALCYCLE, CYCLE : ARRNM;

BEGIN
    (* NEHJOBINSERTIONSEQUENCE *)
    STARTN := 2;
    STARTSEQ[0] := S[0];
    STARTSEQ[1] := S[1]; STARTSEQ[2] := S[2];
    STARTD := MAKESPAN1(M, STARTN, STARTSEQ, P);
    STARTSEQ[1] := S[2]; STARTSEQ[2] := S[1];
    IF STARTD < MAKESPAN1(M, STARTN, STARTSEQ, P) THEN
        BEGIN
            STARTSEQ[1] := S[1];
            STARTSEQ[2] := S[2];
        END;
    FOR I := 0 TO N DO CYCLE[I] := 0;
    CYCLE[0] := STARTSEQ[1];
    CYCLE[STARTSEQ[1]] := STARTSEQ[2];
    CYCLE[STARTSEQ[2]] := 0;
    FOR K := 3 TO N DO
        BEGIN
            INSJOB := S[K];
            NBJOB := K;
            INSD := INF;
            INDEX := 0;
            FOR I := 0 TO N DO
                BEGIN
                    PARTIALCYCLE[I] := CYCLE[I];
                END;
            FOR J := 1 TO K DO
                BEGIN

```

```
NEXTINDEX := PARTIALCYCLE[INDEX];
PARTIALCYCLE[INDEX] := INSJOB;
PARTIALCYCLE[INSJOB] := NEXTINDEX;
Z := 0;
FOR I := 0 TO NBJOB DO
  BEGIN
    PARTIALSEQ[I] := Z;
    Z := PARTIALCYCLE[Z];
  END;

NEWD := MAKESPAN1(M,NBJOB,PARTIALSEQ,P);
IF NEWD < INSD THEN
  BEGIN
    INSD := NEWD;
    END1 := INDEX;
    END2 := NEXTINDEX;
  END;
INDEX := NEXTINDEX;
FOR I := 0 TO N DO
  BEGIN
    PARTIALCYCLE[I] := CYCLE[I];
  END;
END;
CYCLE[INSJOB] := END2;
CYCLE[END1] := INSJOB;
END;
INDEX := 0;
FOR I := 0 TO N DO
  BEGIN
    SEQUENCE[I] := INDEX;
    INDEX := CYCLE[INDEX];
  END;
END;
(* NEHJOBINSERTIONSEQUENCE *)

BEGIN
  (* NEH *)
  FOR I := 0 TO N DO SEQUENCE[I] := 0;
  LPTJOB SORT(LPTSEQ);
  NEHJOBINSERTIONSEQUENCE(N,LPTSEQ,SEQUENCE);
END;
(* NEH *)
END.
```


UNIDADE CONTENDO
OS PROGRAMAS
DE SELEÇÃO DOS PAIS, TROCA DOS PAIS E MUTAÇÃO
(WAGNER LISBOA MOTA)

```
UNIT POPMUT;
```

```
INTERFACE
```

```
USES WINCRT, WINDOS, WINCTV, MAKE;
```

```
PROCEDURE POPULACAO(
  VAR M,N           : INTEGER;
  VAR TAMPOP        : LONGINT;
  VAR TEMPOPRO      : ARRNM;
  VAR CROMO1,CROMO2 : ARRNM;
  VAR M1,M2         : INTEGER);
```

```
PROCEDURE MUTACAO(
  VAR M,N           : INTEGER;
  VAR CROMOT1,CROMOT2 : ARRNM);
```

```
PROCEDURE REPLACE(
  VAR M,N           : INTEGER;
  VAR TAMPOP        : LONGINT;
  VAR CROMOT1,CROMOT2 : ARRNM;
  VAR TEMPOPRO      : ARRNM;
  VAR M3,M4         : INTEGER);
```

```
IMPLEMENTATION
```

```
PROCEDURE POPULACAO(
  VAR M,N           : INTEGER;
  VAR TAMPOP        : LONGINT;
  VAR TEMPOPRO      : ARRNM;
  VAR CROMO1,CROMO2 : ARRNM;
  VAR M1,M2         : INTEGER);
```

```
VAR
  M3,M4,TEMPORARIO,N1,N2 : INTEGER;
  CROMT1, CROMT2         : ARRNM;
```

```
BEGIN
```

```
M1 := MAKESPAN(M,N,CROMO1,TEMPOPRO);
M2 := MAKESPAN(M,N,CROMO2,TEMPOPRO);
```

```
CROMT1 := CROMO1;
```

```
CROMT2 := CROMO2;
```

```
RANDOMIZE;
```

```
REPEAT
```

```
  N1 := RANDOM(N) + 1;
```

```
  N2 := RANDOM(N) + 1;
```

```
UNTIL( N1 <> N2 );
```

```
TEMPORARIO := CROMT1[N1];
```

```
CROMT1[N1] := CROMT1[N2];
```

```
CROMT1[N2] := TEMPORARIO;
```

```
REPEAT
```

```
  N1 := RANDOM(N) + 1;
```

```
  N2 := RANDOM(N) + 1;
```

```
UNTIL( N1 <> N2 );
```

```
TEMPORARIO := CROMT2[N1];
```

```
CROMT2[N1] := CROMT2[N2];
```

```
CROMT2[N2] := TEMPORARIO;
```

```
FOR I := 1 TO TAMPOP DO
```

```
  BEGIN
```

```
    TEMPORARIO := CROMT1[I];
```

```
    FOR J := 1 TO N-1 DO
```

```
      BEGIN
```

```
        CROMT1[J] := CROMT1[J+1];
```

```
      END;
```

```
    CROMT1[N] := TEMPORARIO;
```

```
    M3 := MAKESPAN(M,N,CROMT1,TEMPOPRO);
```

```
    IF M3 < M1 THEN
```

```
      BEGIN
```

```
        CROMO1 := CROMT1;
```

```
        M1 := M3;
```

```
      END;
```

```
    TEMPORARIO := CROMT2[I];
```

```
    FOR J := 1 TO N-1 DO
```

```
      BEGIN
```

```
        CROMT2[J] := CROMT2[J+1];
```

```
      END;
```

```
    CROMT2[N] := TEMPORARIO;
```

```
    M4 := MAKESPAN(M,N,CROMT2,TEMPOPRO);
```

```
    IF M4 < M2 THEN
```

```
      BEGIN
```

```
        CROMO2 := CROMT2;
```

```
        M2 := M4;
```

```
      END;
```

```
END;
```

```
END;
```

```
PROCEDURE MUTACAO(
  VAR M,N           : INTEGER;
  VAR CROMOT1,CROMOT2 : ARRNM);
```

```
VAR
```

```
  N1,N2,X : INTEGER;
```

```
BEGIN
```

```
  RANDOMIZE;
```

```
  REPEAT
```

```
    N1 := RANDOM(N) + 1;
```

```
    N2 := RANDOM(N) + 1;
```

```
  UNTIL( N1 <> N2 );
```

```
  X := CROMOT1[N1];
```

```
  CROMOT1[N1] := CROMOT1[N2];
```

```
  CROMOT1[N2] := X;
```

```
  REPEAT
```

```
    N1 := RANDOM(N) + 1;
```

```
    N2 := RANDOM(N) + 1;
```

```
  UNTIL( N1 <> N2 );
```

```
  X := CROMOT2[N1];
```

```
  CROMOT2[N1] := CROMOT2[N2];
```

```
  CROMOT2[N2] := X;
```

```
END;
```

```
PROCEDURE REPLACE(
```

```
  VAR M,N           : INTEGER;
```

```
  VAR TAMPOP        : LONGINT;
```

```
  VAR CROMOT1,CROMOT2 : ARRNM;
```

```
  VAR TEMPOPRO      : ARRNM;
```

```
  VAR M3,M4         : INTEGER);
```

```
VAR
```

```
  TEMPORARIO,M5,M6 : INTEGER;
```

```
  CROM1,CROM2      : ARRNM;
```

```
BEGIN
```

```
M3 := MAKESPAN(M,N,CROMOT1,TEMPOPRO);
```

```
M4 := MAKESPAN(M,N,CROMOT2,TEMPOPRO);
```

```
CROM1 := CROMOT1;
```

```
CROM2 := CROMOT2;
```

```
FOR I := 1 TO TAMPOP-1 DO
```

```
  BEGIN
```

```
    TEMPORARIO := CROM1[I];
```

```
    FOR J := 1 TO N-1 DO
```

```
      BEGIN
```

```
        CROM1[J] := CROM1[J+1];
```

```
      END;
```

```
    CROM1[N] := TEMPORARIO;
```

```
    M5 := MAKESPAN(M,N,CROM1,TEMPOPRO);
```

```
    IF M5 < M3 THEN
```

```
      BEGIN
```

```
        CROMOT1 := CROM1;
```

```
        M3 := M5;
```

```
      END;
```

```
    TEMPORARIO := CROM2[I];
```

```
    FOR J := 1 TO N-1 DO
```

```
      BEGIN
```

```
        CROM2[J] := CROM2[J+1];
```

```
      END;
```

```
    CROM2[N] := TEMPORARIO;
```

```
    M6 := MAKESPAN(M,N,CROM2,TEMPOPRO);
```

```
    IF M6 < M4 THEN
```

```
      BEGIN
```

```
        CROMOT2 := CROM2;
```

```
        M4 := M6;
```

```
      END;
```

```
    END;
```

```
  END;
```

```
END;
```

```
END.
```

**APÊNDICE II : SOLUÇÕES DOS PROBLEMAS DO
EXPERIMENTO**

SEMENTES NÃO-ALEATÓRIAS

PROBLEMAS DE PEQUENO PORTE

Table with 28 columns: No. DO PROBLEMA, PMX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), MPX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), OX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), EOX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), UMCORTE (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), CX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), LOX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), and summary columns PMX, MPX, OX, EOX, UM CORTE, CX, LOX.

Table with 28 columns: No. DO PROBLEMA, PMX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), MPX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), OX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), EOX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), UMCORTE (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), CX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), LOX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), and summary columns PMX, MPX, OX, EOX, UM CORTE, CX, LOX.

Table with 28 columns: No. DO PROBLEMA, PMX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), MPX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), OX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), EOX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), UMCORTE (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), CX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), LOX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), and summary columns PMX, MPX, OX, EOX, UM CORTE, CX, LOX.

Table with 28 columns: No. DO PROBLEMA, PMX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), MPX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), OX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), EOX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), UMCORTE (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), CX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), LOX (Dur. Inicial, Dur. Final, Desvio Relat., Tempo), and summary columns PMX, MPX, OX, EOX, UM CORTE, CX, LOX.

SEMENTES NÃO-ALÉATORIAS

PROBLEMAS DE GRANDE PORTE

Table with 100 columns (No. DO, PMX, MPX, OX, BOX, UMCORTE, CX, LOX) and 11 rows (04X070P01 to MEDIA). Columns include Dur. Inicial, Dur. Final, Desvio Relat., Tempo for each category.

Summary row for the first table: 7 8 6 8 7 6 7

Table with 100 columns (No. DO, PMX, MPX, OX, BOX, UMCORTE, CX, LOX) and 11 rows (07X070P01 to MEDIA). Columns include Dur. Inicial, Dur. Final, Desvio Relat., Tempo for each category.

Summary row for the second table: 2 3 3 3 2 5 4

Table with 100 columns (No. DO, PMX, MPX, OX, BOX, UMCORTE, CX, LOX) and 11 rows (10X070P01 to MEDIA). Columns include Dur. Inicial, Dur. Final, Desvio Relat., Tempo for each category.

Summary row for the third table: 4 2 2 5 5 4 4

Table with 100 columns (No. DO, PMX, MPX, OX, BOX, UMCORTE, CX, LOX) and 11 rows (15X070P01 to MEDIA). Columns include Dur. Inicial, Dur. Final, Desvio Relat., Tempo for each category.

Summary row for the fourth table: 4 2 2 3 2 2 1

SEMENTES NÃO-ALBATÓRIAS

Table with 35 columns (No. DO, PMX, MPX, OX, EOX, UMCORTE, CX, LOX) and 11 rows of data including a summary row 'MÉDIA'.

Table with 35 columns (No. DO, PMX, MPX, OX, EOX, UMCORTE, CX, LOX) and 11 rows of data including a summary row 'MÉDIA'.

Table with 35 columns (No. DO, PMX, MPX, OX, EOX, UMCORTE, CX, LOX) and 11 rows of data including a summary row 'MÉDIA'.

Table with 35 columns (No. DO, PMX, MPX, OX, EOX, UMCORTE, CX, LOX) and 11 rows of data including a summary row 'MÉDIA'.

TOTAL DE SUCESSOS 32 21 28 29 25 28 28

SEMENTES ALEATÓRIAS

No. DO PROBLEMA	PMX				MPX				OX				EOX				UM CORTE				CX				LOX				PMX	MPX	OX	EOX	UM CORTE	CX	LOX																			
	Dur. Inicial	Dur. Final	Desvio Relat.	Tempo	Dur. Inicial	Dur. Final	Desvio Relat.	Tempo	Dur. Inicial	Dur. Final	Desvio Relat.	Tempo	Dur. Inicial	Dur. Final	Desvio Relat.	Tempo	Dur. Inicial	Dur. Final	Desvio Relat.	Tempo	Dur. Inicial	Dur. Final	Desvio Relat.	Tempo	Dur. Inicial	Dur. Final	Desvio Relat.	Tempo								Dur. Inicial	Dur. Final	Desvio Relat.	Tempo															
20X010P01	1927	1682	21.44%	3.57	2005	1693	22.24%	3.57	1957	1682	21.44%	3.57	1886	1683	21.52%	3.62	1934	1683	21.52%	3.57	1919	1747	26.14%	3.51	1918	1683	21.52%	3.57	1	0	1	0	0	0	0																			
20X010P02	1887	1733	40.89%	3.57	2072	1733	40.89%	3.63	1951	1745	41.87%	3.57	1984	1749	42.20%	3.57	1972	1749	42.20%	3.57	1949	1765	43.50%	3.57	1925	1745	41.95%	3.57	1	1	0	0	0	0	0																			
20X010P03	1872	1724	29.24%	3.57	1925	1719	28.86%	3.63	1907	1714	28.49%	3.57	1881	1713	28.41%	3.62	2022	1709	28.11%	3.57	1912	1734	29.99%	3.51	1833	1715	28.36%	3.57	0	0	0	0	1	0	0																			
20X010P04	1933	1648	20.73%	3.57	1891	1636	19.85%	3.57	1904	1670	22.34%	3.62	1920	1649	20.81%	3.57	1880	1675	22.71%	3.57	1874	1662	21.76%	3.51	1928	1632	19.56%	3.57	0	0	0	0	0	1	0																			
20X010P05	1936	1752	21.41%	3.57	1904	1753	21.48%	3.62	2074	1765	22.31%	3.57	2012	1753	21.48%	3.63	1973	1747	21.07%	3.57	1972	1767	22.45%	3.57	1972	1761	22.04%	3.57	0	0	0	0	1	0	0																			
20X010P06	1845	1622	13.43%	3.57	1888	1630	15.38%	3.63	1961	1655	15.73%	3.63	1844	1615	12.94%	3.63	1825	1661	16.15%	3.57	1984	1648	15.24%	3.57	1879	1615	12.94%	3.52	0	0	0	1	0	0	1																			
20X010P07	2212	1896	19.10%	3.62	2096	1880	18.09%	3.57	2062	1892	18.84%	3.63	2044	1903	19.54%	3.62	2009	1915	20.29%	3.57	2063	1883	18.28%	3.57	2101	1891	18.78%	3.57	0	1	0	0	0	0	0																			
20X010P08	1916	1699	39.72%	3.57	1834	1670	37.34%	3.63	1762	1675	37.75%	3.57	1889	1718	41.28%	3.63	1862	1667	37.09%	3.57	1878	1691	39.06%	3.57	1861	1689	38.90%	3.57	0	0	0	0	1	0	0																			
20X010P09	1845	1630	28.65%	3.57	1816	1611	27.15%	3.63	1761	1624	28.18%	3.63	1824	1620	27.86%	3.57	1852	1629	28.57%	3.57	1668	1620	27.86%	3.51	1764	1641	29.52%	3.62	0	1	0	0	0	0	0																			
20X010P10	1985	1734	29.11%	3.57	1942	1752	30.45%	3.68	1937	1719	28.00%	3.51	1909	1717	27.85%	3.62	1904	1723	28.29%	3.57	1765	1721	28.15%	3.51	1899	1723	28.29%	3.57	0	0	0	1	0	0	0																			
MÉDIA	1935.8	1712	26.37%	3.575	1937.3	1709.7	26.17%	3.616	1927.6	1714.1	26.50%	3.587	1919.3	1712	26.39%	3.608	1923.3	1715.8	26.60%	3.57	1898.4	1723.8	27.24%	3.54	1908	1709.6	26.21%	3.57																										
																												2	3	1	2	3	0	2																				

No. DO PROBLEMA	PMX				MPX				OX				EOX				UM CORTE				CX				LOX				PMX	MPX	OX	EOX	UM CORTE	CX	LOX																																															
	Dur. Inicial	Dur. Final	Desvio Relat.	Tempo	Dur. Inicial	Dur. Final	Desvio Relat.	Tempo	Dur. Inicial	Dur. Final	Desvio Relat.	Tempo	Dur. Inicial	Dur. Final	Desvio Relat.	Tempo	Dur. Inicial	Dur. Final	Desvio Relat.	Tempo	Dur. Inicial	Dur. Final	Desvio Relat.	Tempo	Dur. Inicial	Dur. Final	Desvio Relat.	Tempo								Dur. Inicial	Dur. Final	Desvio Relat.	Tempo																																											
25X010P01	2277	2151	20.98%	4.45	2413	2110	18.67%	4.51	2337	2124	19.46%	4.45	2296	2132	19.91%	4.44	2491	2121	19.29%	4.45	2216	2154	21.15%	4.39	2296	2106	18.45%	4.45	0	0	0	0	0	0	1																																															
25X010P02	2234	2040	31.70%	4.45	2274	2081	34.34%	4.5	2309	2065	33.31%	4.5	2231	2062	33.12%	4.44	2191	2061	33.05%	4.5	2277	2052	32.47%	4.45	2301	2061	33.05%	4.45	1	0	0	0	0	0	0																																															
25X010P03	2280	2054	23.22%	4.51	2272	2027	21.60%	4.45	2205	2045	22.68%	4.5	2260	2029	21.72%	4.5	2276	2025	21.48%	4.45	2338	2049	22.92%	4.45	2235	2055	23.28%	4.51	0	0	0	0	1	0	0																																															
25X010P04	2243	2096	28.35%	4.45	2184	2105	28.90%	4.5	2261	2073	26.94%	4.5	2273	2093	28.17%	4.5	2206	2112	29.33%	4.45	2319	2098	28.48%	4.45	2231	2090	27.99%	4.45	0	0	1	0	0	0	0																																															
25X010P05	2072	1967	20.01%	4.4	2152	1973	20.38%	4.51	2207	1971	20.26%	4.5	2284	2006	22.39%	4.5	2241	1980	20.81%	4.45	2224	1983	20.99%	4.39	2290	1987	21.23%	4.45	1	0	0	0	0	0	0																																															
25X010P06	2296	2125	28.79%	4.51	2304	2125	28.79%	4.51	2342	2145	30.00%	4.51	2301	2102	27.39%	4.45	2358	2147	30.12%	4.51	2313	2127	28.91%	4.45	2244	2150	30.30%	4.51	0	0	0	1	0	0	0																																															
25X010P07	2191	1899	16.93%	4.45	2054	1896	16.75%	4.45	2177	1901	17.06%	4.45	2194	1890	16.38%	4.5	2045	1863	14.72%	4.45	2035	1862	14.66%	4.45	2299	1990	22.54%	4.5	0	0	0	0	1	0	0																																															
25X010P08	2340	1974	22.99%	4.45	2237	1997	24.42%	4.5	2394	1976	23.12%	4.5	2249	1953	21.68%	4.51	2356	1953	21.68%	4.39	2330	1997	24.42%	4.44	2298	1978	23.24%	4.44	0	0	0	1	1	0	0																																															
25X010P09	2055	1917	21.71%	4.5	2331	1917	21.71%	4.45	2136	1909	21.21%	4.5	2138	1935	22.86%	4.5	2128	1914	21.52%	4.45	2104	1930	22.54%	4.45	2077	1909	21.21%	4.5	0	0	1	0	0	0	1																																															
25X010P10	2349	2129	22.64%	4.45	2324	2102	21.08%	4.44	2453	2121	22.18%	4.45	2410	2085	20.10%	4.51	2261	2151	23.91%	4.45	2263	2141	23.33%	4.4	2298	2131	22.75%	4.51	0	0	0	1	0	0	0																																															
MÉDIA	2233.7	2035.2	23.73%	4.462	2254.5	2033.3	23.67%	4.482	2282.1	2033	23.62%	4.486	2263.6	2028.7	23.37%	4.485	2255.3	2032.7	23.59%	4.455	2241.9	2039.3	23.99%	4.432	2229.9	2045.7	24.40%	4.477																																																						
																												2	0	2	3	2	1	2																																																
																												TOTAL DE SUCESSOS																												12	10	13	12	17	8	11																				

PROBLEMAS DE MÉDIO PORTE

Table with 29 columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo (repeated for PMX, MPX, OX, EOX, UM CORTE, CX, LOX), PMX, MPX, OX, EOX, UM CORTE, CX, LOX. Rows include 04X030P01 to 04X030P10 and a MÉDIA row.

Table with 29 columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo (repeated for PMX, MPX, OX, EOX, UM CORTE, CX, LOX), PMX, MPX, OX, EOX, UM CORTE, CX, LOX. Rows include 07X030P01 to 07X030P10 and a MÉDIA row.

Table with 29 columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo (repeated for PMX, MPX, OX, EOX, UM CORTE, CX, LOX), PMX, MPX, OX, EOX, UM CORTE, CX, LOX. Rows include 10X030P01 to 10X030P10 and a MÉDIA row.

Table with 29 columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo (repeated for PMX, MPX, OX, EOX, UM CORTE, CX, LOX), PMX, MPX, OX, EOX, UM CORTE, CX, LOX. Rows include 15X030P01 to 15X030P10 and a MÉDIA row.

SEMENTES ALBATÓRIAS

Table with 28 columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo (repeated for FMX, MPX, OX, EOX, UM CORTE, CX, LOX), and success indicators (PMX, MPX, OX, EOX, UM CORTE, CX, LOX). Rows include individual problem data and a final 'MÉDIA' row.

Table with 28 columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo (repeated for FMX, MPX, OX, EOX, UM CORTE, CX, LOX), and success indicators (PMX, MPX, OX, EOX, UM CORTE, CX, LOX). Rows include individual problem data and a final 'MÉDIA' row.

TOTAL DE SUCESSOS 8 2 12 13 11 8 14

Table with 28 columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo (repeated for FMX, MPX, OX, EOX, UM CORTE, CX, LOX), and success indicators (PMX, MPX, OX, EOX, UM CORTE, CX, LOX). Rows include individual problem data and a final 'MÉDIA' row.

Table with 28 columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo (repeated for FMX, MPX, OX, EOX, UM CORTE, CX, LOX), and success indicators (PMX, MPX, OX, EOX, UM CORTE, CX, LOX). Rows include individual problem data and a final 'MÉDIA' row.

SEMENTES ALLATÓRIAS

PROBLEMAS DE GRANDE PORTE

Table with columns for PMX, MPX, OX, EOX, UM CORTE, CX, LOX and rows for No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo, etc.

Table with columns for PMX, MPX, OX, EOX, UM CORTE, CX, LOX and rows for No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo, etc.

Table with columns for PMX, MPX, OX, EOX, UM CORTE, CX, LOX and rows for No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo, etc.

Table with columns for PMX, MPX, OX, EOX, UM CORTE, CX, LOX and rows for No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo, etc.

SEMENTES ALEATÓRIAS

Table with columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo (PMX, MPX, OX, EOX, UM CORTE, CX, LOX), and success indicators (PMX, MPX, OX, EOX, UM CORTE, CX, LOX).

Table with columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo (PMX, MPX, OX, EOX, UM CORTE, CX, LOX), and success indicators (PMX, MPX, OX, EOX, UM CORTE, CX, LOX).

TOTAL DE SUCESSOS 14 8 6 7 16 11 4

Table with columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo (PMX, MPX, OX, EOX, UM CORTE, CX, LOX), and success indicators (PMX, MPX, OX, EOX, UM CORTE, CX, LOX).

Table with columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo (PMX, MPX, OX, EOX, UM CORTE, CX, LOX), and success indicators (PMX, MPX, OX, EOX, UM CORTE, CX, LOX).

SEMENTES ALEATÓRIAS

Table with columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo, PMX, MPX, OX, EOX, UM CORTE, CX, LOX. Contains 10 rows of data and a final 'MÉDIA' row.

Small table with columns: PMX, MPX, OX, EOX, UM CORTE, CX, LOX. Contains 10 rows of data and a final 'MÉDIA' row.

Table with columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo, PMX, MPX, OX, EOX, UM CORTE, CX, LOX. Contains 10 rows of data and a final 'MÉDIA' row.

Small table with columns: PMX, MPX, OX, EOX, UM CORTE, CX, LOX. Contains 10 rows of data and a final 'MÉDIA' row.

Table with columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo, PMX, MPX, OX, EOX, UM CORTE, CX, LOX. Contains 10 rows of data and a final 'MÉDIA' row.

Small table with columns: PMX, MPX, OX, EOX, UM CORTE, CX, LOX. Contains 10 rows of data and a final 'MÉDIA' row.

Table with columns: No. DO PROBLEMA, Dur. Inicial, Dur. Final, Desvio Relat., Tempo, PMX, MPX, OX, EOX, UM CORTE, CX, LOX. Contains 10 rows of data and a final 'MÉDIA' row.

Small table with columns: PMX, MPX, OX, EOX, UM CORTE, CX, LOX. Contains 10 rows of data and a final 'MÉDIA' row.

TOTAL DE SUCESSOS: 10 5 8 9 11 12 8