EDSON ROVINA

SEQUENCIAMENTO DE EMBARQUE DE GRANÉIS SÓLIDOS POR MEIO DE TERMINAIS PORTUÁRIOS UTILIZANDO ALGORITMOS GENÉTICOS

Dissertação apresentada ao Curso de Pós-Graduação em Métodos Numéricos de Engenharia, concentração em Programação Matemática, Setor de Tecnologia, Departamento de Construção Civil e Setor de Ciências Exatas, Departamento de Ciências Exatas, Departamento de Matemática da Universidade Federal do Paraná.

Orientador: Prof. Dr. Celso Carnieri
Co-orientador: MSc. Marcos Masnik

CURITIBA

2008

TERMO DE APROVAÇÃO

EDSON ROVINA

SEQUENCIAMENTO DE EMBARQUE DE GRANÉIS SÓLIDOS POR MEIO DE TERMINAIS PORTUÁRIOS UTILIZANDO ALGORITMOS GENÉTICOS

Dissertação aprovada como requisito parcial à obtenção de grau de Mestre em Ciências, Curso de Pós-Graduação em Métodos Numéricos de Engenharia, concentração em Programação Matemática, Setor de Tecnologia, Departamento de Construção Civil e Setor de Ciências Exatas, Departamento de Ciências Exatas, Departamento de Matemática da Universidade Federal do Paraná, pela seguinte banca examinadora:

Orientador Dr. Eng. Celso Carnieri
Programa de Pós-Graduação em Métodos Numéricos em Engenharia - UFPR

Dr. Eng. Leandro Coelho
Depto. de Engenharia de Produção – PUCPR

Dr. Eng. Luis Fernando Nunes
Depto. de Matemática - UTFPR

Co-orientador M.Cs. Marcos Antonio Masnik Ferreira
Analista do Banco Central do Brasil

Curitiba
2008

AGRADECIMENTOS

Ao longo do período em que dedicamos à essa jornada, muitas pessoas estenderam a mão do apoio e utilizaram a voz do reconforto. A todos vocês que, de forma explicita ou implícita contribuíram, minha eterna gratidão. Entretanto alguns merecem um especial destaque.

Ao Prof. Dr. Celso Carnieri e ao MCs. Marcos Masnik pela acolhida, pela paciência, pela compreensão e pela indispensável ajuda.

A administração do porto de Paranguá e em especial, aos senhores Gilmar e Jocilei pela oportunidade fornecida e principalmente por compartilharem seus valiosos conhecimentos sobre sequenciamento de navios graneleiros.

A todos os professores do programa pós graduação em métodos numéricos de engenharia que ensinaram lições que vão além da técnica e da ciência.

As três gerações de mulheres da minha vida, minha filha, Tassiana pela compreensão da privação do convívio de muitas horas, a minha esposa Linda pelo incondicional apoio em todos os momentos e a minha mãe, Nilde Galasse pelo exemplo de determinação.

Aos amigos e companheiros do programa pós graduação em métodos numéricos de engenharia e em especial, a Carlos Smaka, Clodoaldo José Figueiredo, Maickel Hubner e Thober Detofeno pelo incentivo, apoio, ajuda e carinho que me dedicaram.

A Maristela Bandil, secretária do PPGMN, que nunca poupou esforços para nos ajudar nas mais diversas situações.

Pois é indigno destes doutos homens perder horas como escravos, em trabalhos de cálculos que poderiam, com segurança, ficar a encargo de qualquer pessoa, caso se utilizassem máquinas

GOTTFRIED WILHELM VON LEIBNITZ

(21/06/1646 - 14/11/1716)

RESUMO

No embarque de granéis em terminais portuários (TP), os conflitos ocorrem quando há a necessidade de embarque simultâneo de um TP em mais de um navio. Através de uma seqüência otimizada pode-se eliminar ou minimizar os conflitos de embarque. O problema é classificado dentro da literatura como NP - difícil, ou seja, problemas onde o número de operações do melhor algoritmo conhecido cresce exponencialmente. É um problema *job-shop*, por ser caracterizado por programar o embarque de granéis de forma intermitente e diversificada e, na solução apresentada foi considerada a programação para frente e finita. Os algoritmos genéticos (AG) tem sido alvo de pesquisa em diversas áreas e apresentam bons resultados, com tempo computacional aceitável, para a obtenção do sequencimanto. O funcionamento do AG é simples sendo uma meta-heurística, baseada em processo probabilístico e, o ponto chave da sua concepção é a definição da representação, dos operadores e da função objetivo. O AG utilizado é baseado em ordem, e apresentou resultados satisfatórios para o sequenciamento de TPs.

Palavras chaves: sequenciamento, embarque, programação para frente, programação finita, algoritmos genéticos baseados em ordem.

ABSTRACT

In the load of granary in terminal ports (PT) the conflicts happen when there is the necessity of simultaneous load of a terminal port (PT) in more than one ship. Over a determinate optimized schedule, conflicts of shipment can be minimized or eliminated. The problem is classified in literature as NP – hard, when the number of operations of the best known algorithm grows exponentially. It's a job-stop problem for being characterized for programming the granary load in an intermittently and diversified way and in the presented solution it was considered the forward and finite programming. The genetic algorithms (GAs) have been in focus of researches in different areas and present good results with reasonable computational time for obtaining the schedule loading. The operation of GAs is simple for being a meta-heuristics based in probabilistic process and the primal key of its conception is the definition of the operators' representation and the objective function. The GAs used is based in order and presented satisfactory results for the schedule load of PTs.

Keywords: schedule, loading, forward programming, finite programming, genetic algorithms based in order.

ABREVIATURAS

APPA – Administração dos Portos de Paranguá e Antonina

NP – não polinomial

TP – Terminal portuário

SL – *Shiploader*

LC – Linha de carregamento

SV - Terminal APPA - farelo

SH - Terminal APPA

CB - Terminal Coimbra

CG - Terminal Cargill

CL - Terminal CBL

CO - Terminal Coamo

CS - Terminal Centro Sul

CT - Terminal Cotriguaçú

PA - Terminal AGTL

SL – Silo

p – Página

L – Número de variáveis binárias

S – String

t – Tamanho da *string*

n – quantidade de símbolos da *string*

Γ - conjunto

$*$ - *Wildcard*

k – Posição na *string*

O(H) – Ordem do esquema

δ(H) – Tamanho do esquema

SOT – *Shortest Operation Time*

N(s) - Vizinhança

s - Solução

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Como efeito da globalização a competição tornou-se acirrada nos mais diversos setores e, nesse sentido, não é suficiente que a produção ocorra com os menores custos e prazos, mas também necessário que os custos e os tempos das demais atividades agregadas, como as de logísticas sejam minimizados. O Brasil, hoje, é um dos principais produtores mundiais de grão, e desde a fonte produtora até o comprador, existe um longo percurso. Neste contexto, tanto para a exportação quanto para a importação, os portos assumem uma função vital.

O principal objetivo dessa dissertação é o de desenvolver um algoritmo que possa servir como base para apoio a decisão na atividade de programação de curto prazo (*scheduling*) de embarque de grãos em navios de diversos terminais portuários. No presente estudo, foram analisados os dados relativos ao embarque de grãos do Porto de Paranaguá, na cidade de Paranaguá no estado do Paraná.

O problema de sequenciamento de TPs é um problema de *job-shop*, com programação finita e para frente, e sendo utilizado algoritmo genético, que é uma das alternativas da computação evolutiva, que está inserida no contexto de inteligência computacional estudada pela computação natural.

Uma das maiores dificuldades encontradas por aqueles que trabalham nessa área é obter uma seqüência de embarque ótima, que evite eventuais paradas no processo de carregamento oriundo de conflitos gerados pela necessidade de embarque simultâneo de grãos do mesmo terminal em diferentes berços.

OBJETIVOS

Propor e desenvolver um algoritmo que efetue programação de curto prazo (*scheduling*) para o embarque de granéis sólidos (soja, farelo, trigo, etc) no porto de Paranaguá que minimize os tempos de conflitos produzidos pela necessidade de carregamentos simultâneos de um mesmo terminal portuário e, consequentemente reduzindo o tempo total de embarque.

Mais especificamente, os objetivos do trabalho são:

- Analisar o atual modelo de sequenciamento de embarques.

- Classificar o problema na administração de produção.
- Elaborar um algoritmo para o sequenciamento de embarque de granéis sólidos no porto de Paranaguá, utilizando algoritmos genéticos
- Implementar o algoritmo numa linguagem de programação.
- Testar e validar o algoritmo.

ESTRUTURA DE EMBARQUES DE GRANÉIS SÓLIDOS DO PORTO DE PARANAGUÁ

O complexo de embarque de granéis no porto de Paranaguá, conhecido como corredor de exportação, é composta de nove Terminais Portuários (TPs), sendo sete unidades privadas, e uma unidade pública composta de um silo vertical para grãos e quatro silos horizontais para farelo de soja. Existem três berços de atracação e a possibilidade de operar simultaneamente com duas linhas de carregamento em cada berço de atracação. Alguns dos TPs possuem somente uma linha de carregamento. No quadro 1.1 estão indicados os terminais portuários do porto de Paranaguá e as suas características operacionais (quantidade linhas de carregamento, capacidade por linha). As informações referentes às capacidades foram fornecidas pelos operadores do sistema.

Terminal	Abreviatura utilizada	Capacidade operacional por linha (t/h)	Quantidade de linhas de carregamento	Capacidade operacional total (t/h)
Terminal APPA - farelo	SV	500	1	500
Terminal APPA	SH	1000	2	2000
Terminal Coimbra	CB	1000	2	2000
Terminal Cargill	CG	1000	2	2000
Terminal CBL	CL	1000	2	2000
Terminal Coamo	CO	1000	2	2000
Terminal Centro Sul	CS	1000	2	2000
Terminal Cotriguaçú	CT	1000	2	2000
Terminal AGTL	PA	1000	1	1000

QUADRO 1.1 - Quantidade de linhas e capacidade por terminal portuário

A figura 1.1 mostra a estrutura de embarque do corredor de exportação do porto de Paranaguá. Cada um dos berços (212, 213 e 214) possui dois equipamentos de carregamento de navios, *shiploaders* (SL), que estão conectados

as linhas de carregamento (LC), que por sua vez têm a possibilidade de receber as mercadorias de cada um dos TPs.

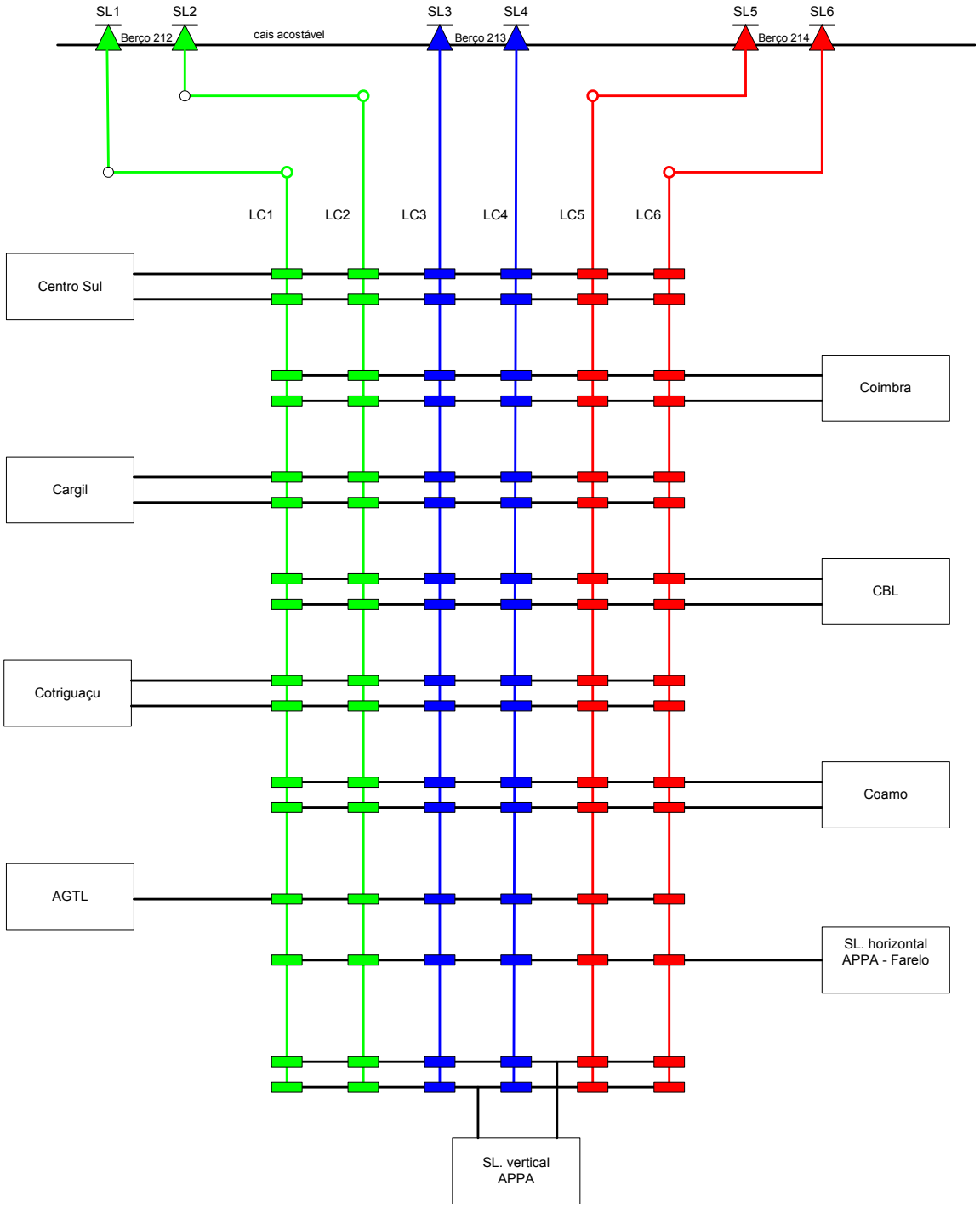

Figura 1.1 - Diagrama operacional do corredor de exportação

IMPORTÂNCIA DO TRABALHO

No ano de 2006, conforme se observa no quadro 1.2, o porto de Paranaguá movimentou um total de 20.316.486 toneladas, sendo que 12.655.448 toneladas representam a exportação de grãos (soja, farelo, milho, etc). Por estes dados verifica-se a importância da movimentação de grãos, visto que a exportação destes representa 62,29% do movimento total.

Curso Longo					
MERCADORIAS	SENTIDO	2003	2004	2005	2006
CEVADA	IMP	129.328	47.883	42.579	92.429
SOJA	EXP	5.931.950	5.084.975	5.227.856	4.046.803
FARELOS	EXP	5.962.041	5.282.377	5.501.985	5.058.780
FERTILIZANTES	IMP	5.708.009	5.559.444	4.541.763	4.826.763
MINÉRIOS	IMP	609	286	378	26
AÇUCAR	EXP	1.420.774	1.235.331	1.678.811	2.406.135
ARROZ	IMP	-	-	-	-
SAL	IMP	65.296	92.530	74.332	137.587
TRIGO	IMP	255.487	5.503	-	96.707
MILHO	EXP	2.765.671	3.541.294	620.836	3.347.487
DIVERSAS	EXP	-	-	-	-
DIVERSAS	IMP	42	61	142	-
S O M A CURSO LONGO		22.239.207	20.849.684	17.688.682	20.012.717
C A B O T A G E M					
MERCADORIAS	SENTIDO	2003	2004	2005	2006
SAL	IMP	160.400	122.081	111.874	135.730
MINÉRIOS	IMP	55.553	52.478	40.921	62.368
MILHO	EXP	120.714	60.839	797	105.671
FARELOS	EXP	-	-	-	-
S O M A CABOTAGEM		336.667	235.398	153.592	303.769
TOTAL GERAL		22.575.874	21.085.082	17.842.274	20.316.486

QUADRO 1.2 - Volume de Exportação de Granéis Sólidos no porto de Paranaguá em Toneladas
FONTE: APPA – Administração dos Portos de Paranaguá e Antonina

Em janeiro de 2007, o Governo Federal, através do PAC – Plano de Aceleração do Crescimento, divulgou investimento em diversas áreas e segundo o DNIT – Departamento Nacional de Infra-Estrutura em Transportes, até o ano de 2010 serão investidos R$ 55,2 bilhões para a infra-estrutura logística. Para a região sul do Brasil foram destinados investimentos em diversas localidades e o porto de Paranaguá será contemplado com a recuperação dos seus berços.

Os investimentos na área de infra-estrutura são de vital importância para a melhoria do complexo portuário. No entanto, o aumento do desempenho não

depende somente deste tipo de investimento. Segundo LANDMANN (2005), no adequado balanceamento dos recursos e na geração de um programa de produção que proporcione um atendimento satisfatório da clientela, com elevado índice de aproveitamento da capacidade disponível, reside o ponto-chave para a obtenção de níveis elevados de produção, produtividade e qualidade com custos adequados.

PEDROSO e CORRÊA (1996) afirmam que a busca da competitividade por parte das empresas, notadamente quando se objetiva reduzir os custos coloca o sistema de planejamento, programação e controle da produção (PPCP) como uma área de decisão prioritária para os executivos nos anos 90.

Conforme GOEBEL (1996, p. 1)

> À medida que a economia mundial vai se tornando cada vez mais globalizada e o Brasil vai incrementando gradativamente o seu comércio exterior, a logística passa a ter um papel acentuadamente mais importante, pois comércio e indústria consideram o mercado mundial como os seus fornecedores e clientes.

Segundo COELI (2004),

> A produção nacional de soja, por exemplo, apresenta custos extremamente baixos se comparados aos custos dos demais países produtores. Entretanto, o custo de logística para conduzir os grãos das áreas de origem aos portos aumenta excessivamente o custo total da soja brasileira.

O aumento da capacidade de operação pode ser obtido com um sequenciamento otimizado do embarque dos navios, por meio diversas técnicas de modelagem matemáticas, quer sejam a programação dinâmica, meta-heurística ou simulação. Desta forma, com um embarque otimizado de granéis, pode-se obter uma redução nos tempos de permanência dos navios no porto, o que consequentemente beneficiaria todos os usuários do complexo portuário.

DELIMITAÇÃO DO TRABALHO

O presente trabalho possui um escopo definido, visto que objetiva sequenciar o embarque dos navios graneleiros. Para tanto, somente é considerado o processo após a liberação do navio para a sua atracação. De acordo com as normas do porto de Paranaguá, um navio só adquire a condição de atracação quando toda mercadoria a ser embarcada já se encontra nos TPs. Portanto, a solução proposta parte do pressuposto de que toda a carga já está disponível para embarque.

Outro aspecto não considerado é o período de inatividade dos berços de atracação causado pelo efeito das marés, o que pode alterar o momento de início e ainda causar tempos de espera para a efetiva liberação do berço de atracação para o próximo navio.

ESTRUTURA DO TRABALHO

O presente trabalho está estruturado em seis capítulos além desta introdução.

No segundo capítulo, encontra-se a revisão da literatura nos aspectos da administração de produção e o terceiro capítulo aborda os fundamentos dos algoritmos genéticos.

O quarto capítulo dedica-se a descrever o problema real, detalhando suas peculiaridades e características. No quinto capítulo é apresentado o algoritmo proposto.

O sexto capítulo apresenta os resultados obtidos, as análises elaboradas e as conclusões. Neste capítulo, também, são abordados algumas sugestões para trabalhos futuros.

2 ADMINISTRAÇÃO DE PRODUÇÃO

Antes de iniciar a discussão do método proposto para a solução da programação de curto prazo (*scheduling*) de embarque de grãos em navios de diversos terminais portuários, é necessário situar o contexto de forma geral. Assim neste capítulo, verifica-se como é classificada a programação de curto prazo dentro da literatura e quais são as principais características que ela pode assumir em virtude das diferentes necessidades de prazo de entrega e demais fatores.

FUNÇÃO DA PRODUÇÃO

Conforme CORRÊA e CORRÊA (2007, p. 24), a gestão de operações ocupa-se da atividade de gerenciamento estratégico dos recursos escassos (humanos, tecnológicos, informacionais e outros), de sua interação e dos processos que produzem e entregam bens e serviços, visando atender a necessidade e/ou desejos de qualidade, tempo e custo de seus clientes. Muitos autores concordam com essa definição, que pode ser representada esquematicamente conforme verificada na figura 2.1.

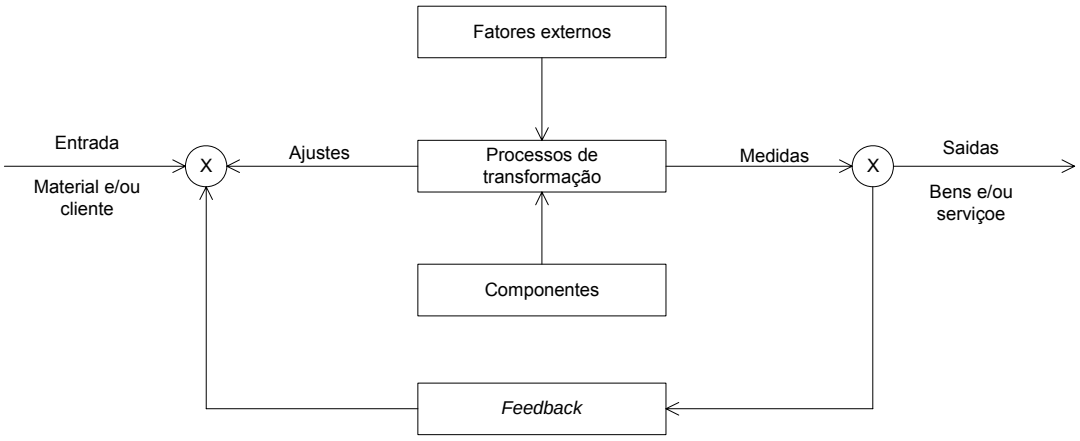

Figura 2.1 - Esquema Geral do Papel da Administração de Produção
FONTE: DARÚ (2005, p. 06)

Pelo esquema geral se identificam três componentes principais: as entradas, o processo de transformação e as saídas, e dois componentes complementares: os

ajustes das entradas e as medidas das saídas. A função Produção não compreende apenas a produção de bens, mas também a produção de serviços, tais como as atividades de armazenamento, movimentação, entretenimento, etc quando utilizadas como serviços (TUBINO,1997). No quadro 2.1 apresentam-se alguns exemplos das considerações que podem ser feitas em função dos tipos de operação.

Tipos de operação	Sistemas Produtivos
Produção de bens	Manufaturas, construção civil, estaleiros, agropecuários, etc
Movimentação e armazenagem	Correio, hotelaria, transportadoras, aerolineas, entrepostos
Entretenimento e comunicação	Estações de rádio e TV, clubes, estúdios de cinema, telecomunicações, jornais
Aluguel, permuta e empréstimos	Banco, operadoras de *leasing*, seguradoras, locadoras de bens

QUADRO 2.1 - Exemplos de operações produtivas
FONTE: TUBINO (1997, p. 19)

PLANEJAMENTO E CONTROLE DE PRODUÇÃO

Conforme SLACK *et al* (1997, p. 329), o planejamento e controle requerem a conciliação do fornecimento e da demanda em termos de volume, em termos de tempo e em termos de qualidade, e para conciliar o volume e o tempo, são desempenhadas três atividades distintas, embora integradas:

- Carregamento: determinação do volume com o qual uma operação produtiva pode lidar;
- Sequência: determinação da prioridade de tarefas a serem desempenhadas.
- Programação: decisão do tempo de início e fim para cada tarefa.

Assim o planejamento e controle de produção (PCP) é um sistema de informações que gerencia a produção, desde a obtenção e a concepção dos dados de planejamento até a sua utilização no dia-a-dia, mediante a adoção de regras para o seu funcionamento, visando comandar o processo produtivo (LANDAMMAN, 2005).

Nesse sentido há concordância dos demais autores ERDMANN (1998, *apud* CONSENTINO e ERDMANN, 1999, p. 25) de que o PCP é uma função administrativa que determina o que, quanto, quando, onde, como e quem vai

produzir. Porém para ABREU (2001, *apud* GADIOLI, 2003) é indispensável que a produção ocorra em bases organizadas e parametrizadas e sob o máximo controle. Para PEDROSO e CORREA (1996, p. 61-62), estas decisões definem quatro determinantes do desempenho destes sistemas:

- Os níveis, em volume e mix, de estoques de matérias-primas, produtos em processo e produtos acabados;
- Os níveis de utilização e de variação da capacidade produtiva (e, conseqüentemente, os custos financeiros e organizacionais decorrentes de ociosidade, hora extra, demissão, contratação, sub-contratação e outros);
- O nível de atendimento à demanda dos clientes, considerando a disponibilidade dos produtos em termos de quantidades e prazos de entrega;
- A competência quanto à reprogramação da produção, abordando a forma como a empresa reage às mudanças não previstas nos seus recursos de produção e na demanda dos clientes.

HIERARQUIA DO PLANEJAMENTO DE PRODUÇÃO

A necessidade do uso do conceito de hierarquia dos processos de planejamento que, segundo CORRÊA, GIANESI e CAON (2001), se dá em virtude das inércias diferentes das decisões que são tomadas em relação aos sistemas de produção. Decisões estratégicas são as que têm maior inércia e são mais difíceis de reverter e, uma vez tomadas, passam a representar restrições às alternativas de decisão de menor inércia. Portanto, as decisões maiores, de maior complexidade e abrangência, vão hierarquicamente restringindo as decisões menores e devem ser respeitadas para que haja coerência entre os níveis de planejamento.

O processo de tomada de decisão, no contexto do planejamento de produção, somente pode ser descrito por meio de uma estrutura hierárquica que segundo HAX e CANDEA (1984) consiste de três níveis distintos, porém altamente interativos: estratégico, tático e operacional.

Planejamento Estratégico da Produção

É um plano de produção de longo prazo baseado nas estimativas de vendas e disponibilidades de recursos financeiros e produtivos. É um plano pouco detalhado, sendo normalmente trabalhado com as famílias de produtos, e tem como finalidade adequar as capacidades dos recursos produtivos à demanda esperada dos mesmos.

Planejamento Mestre de Produção

É considerado de médio prazo e consiste em estabelecer um plano mestre de produção detalhando a médio prazo, período a período, os produtos finais, tomando como base o planejamento estratégico da produção, analisando as necessidades de recursos visando a identificar possíveis gargalos que possam inviabilizar a sua execução no curto prazo.

Programação de Produção

Com base nos estoques e no planejamento mestre de produção, a programação de produção estabelece, em curto prazo, quanto e quando comprar, montar, fabricar, cada um dos itens necessários a montagem final dos produtos.

O planejamento de curtíssimo prazo, conforme alguns autores, visa a atender os problemas de última hora tais como a quebra de máquinas, alteração de mix de produção, pedidos prioritários, atrasos de materiais e outros. A programação de curtíssimo prazo é muito utilizada na manufatura de serviços, no qual a velocidade de entrada de novos pedidos no mix de produção, normalmente não é medida em dias ou semanas, mas sim, em horas.

GRÁFICO DE GANTT

O gráfico de Gantt é um método de representação de programação comumente utilizado e, é uma ferramenta simples, descrito por H.L. Gantt em 1917 (*apud* DARU, 2005), que representa o tempo de operação de uma determinada

atividade como uma barra. Permite visualizar a relação entre máquinas e tarefas, exibindo os momentos de início, término e ociosidades.

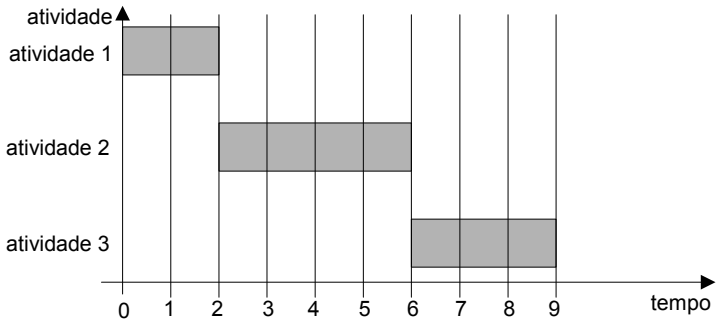

Figura 2.2 - Exemplo de gráfico de Gantt

As unidades de tempo podem ser expressas em horas, dias, semanas, etc. Cada uma das barras representa o tempo total de duração da atividade. Assim, por exemplo, a atividade um, tem duas unidades de tempo de duração, e inicia no tempo zero, e termina no tempo dois.

CLASSIFICAÇÃO DOS SISTEMAS DE PRODUÇÃO

A classificação dos sistemas de produção é realizada em função da abordagem dos recursos e em função das considerações ou não das restrições das capacidades e, são classificados em carregamento infinito e finito.

Os sistemas de produção também podem ser classificados em função dos momentos de entrega dos pedidos, sendo classificados como: programação para trás (*backward*) e programação para frente (*forward*).

Carregamento Infinito

O carregamento infinito, conforme CORRÊA e CORRÊA (2007, p. 581) ocorre quando se alocam tarefas a recursos simplesmente com base nas necessidades de atendimento de prazos. Chama-se infinito, pois programa as atividades, desconsiderando restrições das capacidades, ou seja, considera os recursos como sendo infinitos. Nesse mesmo sentido, SLACK *et al.* (1997) afirmam que o carregamento infinito é particularmente relevante para as operações em que:

- Não é possível limitar a carga.
- Não é necessário limitar a carga.
- O custo de limitação da carga é proibitivo.

A figura 2.3 apresenta uma situação hipotética de quatro atividades e com quatro datas distintas, considerando o carregamento infinito. A linha pontilhada representa o limite da capacidade de produção, que no exemplo é de duas unidades. Assim, na segunda semana, verifica-se que a exigência é de três unidades produtivas, havendo o carregamento em excesso de uma unidade.

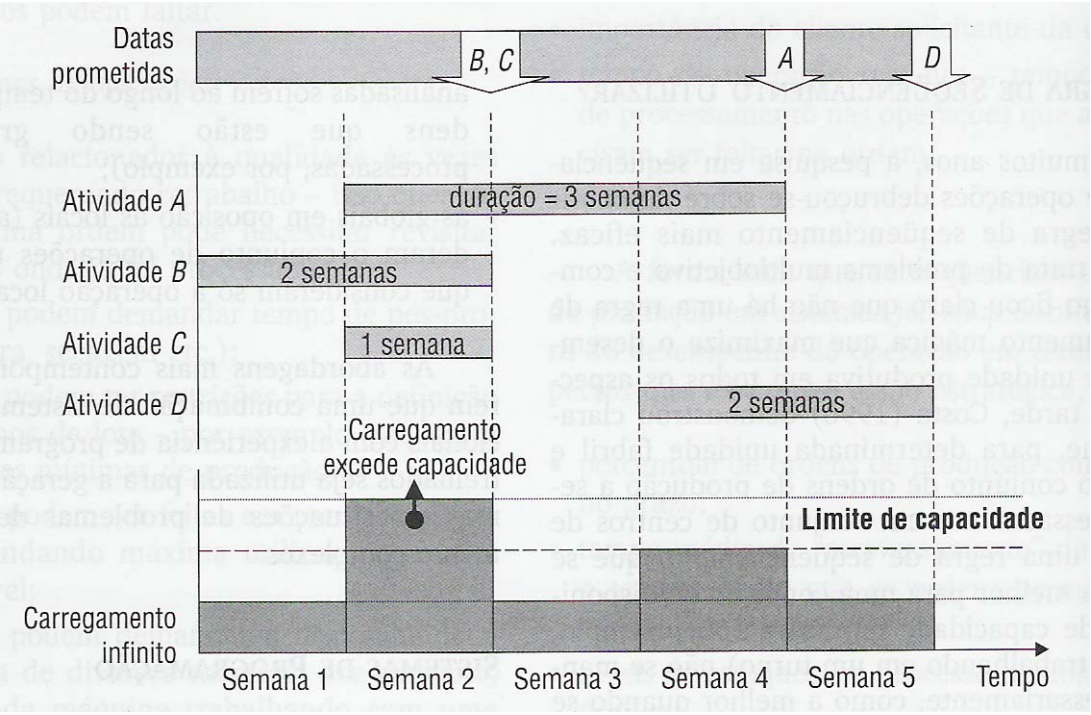

Figura 2.3 - Exemplo de carregamento infinito
FONTE: CORRÊA e CORRÊA (2007, p. 582)

Carregamento Finito

Segundo CORRÊA e CORRÊA (2007, p. 582), o carregamento finito ocorre quando a programação considera a utilização de recursos e sua disponibilidade detalhada no momento do carregamento e não programa uma ordem ou atividade para um período em que não haja disponibilidade de recursos. Nesse mesmo sentido, SLACK *et al.* (1997) descrevem que o carregamento finito é particularmente relevante para as operações em que:

- É possível limitar a carga.
- É necessário limitar a carga.
- O custo de limitação da carga não é proibitivo.

As mesmas condições, apresentadas no item 2.4.1, estão demonstradas figura 2.4, porém considerando o carregamento finito.

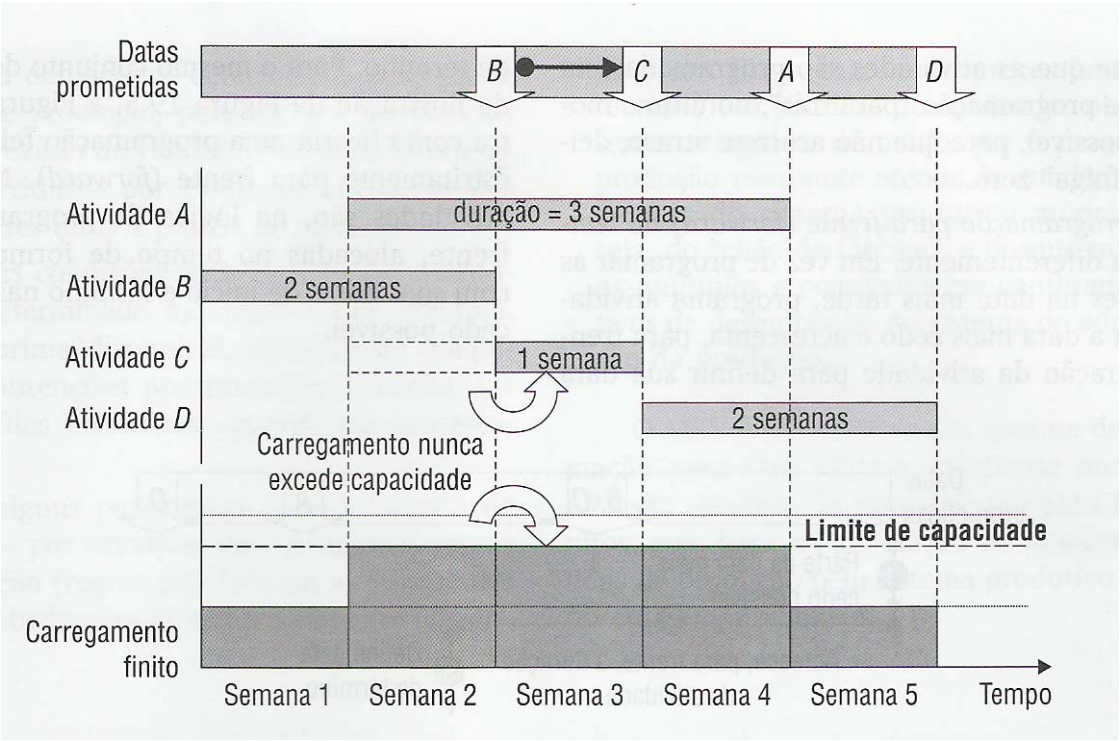

Figura 2.4 - Exemplo de carregamento finito
FONTE: CORRÊA & CORRÊA (2007, p. 583)

É importante observar a diferença existente entre o carregamento finito e infinito. Neste exemplo, a atividade “C” foi deslocada em uma semana para que o limite da capacidade pudesse ser respeitado, eliminando o excesso de uma unidade na segunda semana verificada no caso de carregamento infinito conforme se observa na figura 2.3.

Ainda, os sistemas de produção podem ser classificados em função dos momentos de entrega dos pedidos, sendo classificados como programação para trás (*backward*) e programação para frente (*forward*).

Programação para Trás (*backward*)

A programação para trás inicia as operações em algum momento futuro, normalmente na data prevista para a finalização das tarefas, não produzindo folgas no término das operações e sim no início e/ou no meio do processo. A figura 2.5 mostra um exemplo dessa situação, considerando quatro tarefas.

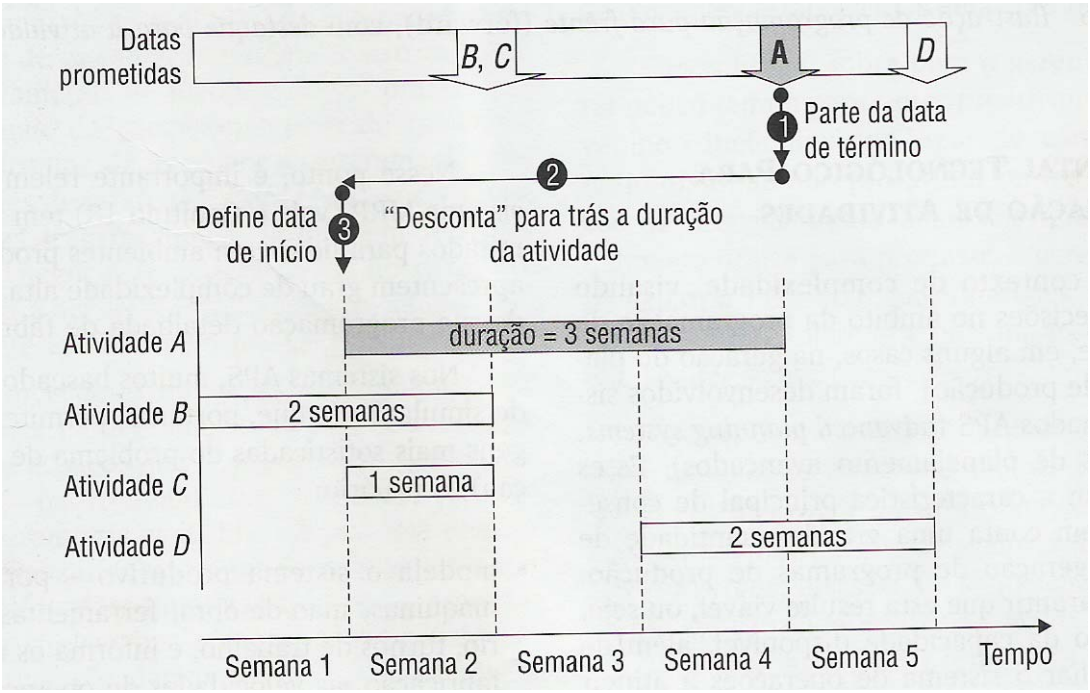

Figura 2.5 - Programação para trás
FONTE: CORRÊA & CORRÊA (2007, p. 583)

Programação para Frente (*forward*)

A programação para frente considera a alocação da tarefa a partir do momento de sua solicitação, partindo sempre da data mais cedo possível e dessa forma produz as folgas no final, possibilitando, em algumas situações, que a atividade seja concluída antes do prazo necessário. Na figura 2.6, o mesmo exemplo utilizado na programação para trás é mostrado agora na situação da programação para frente.

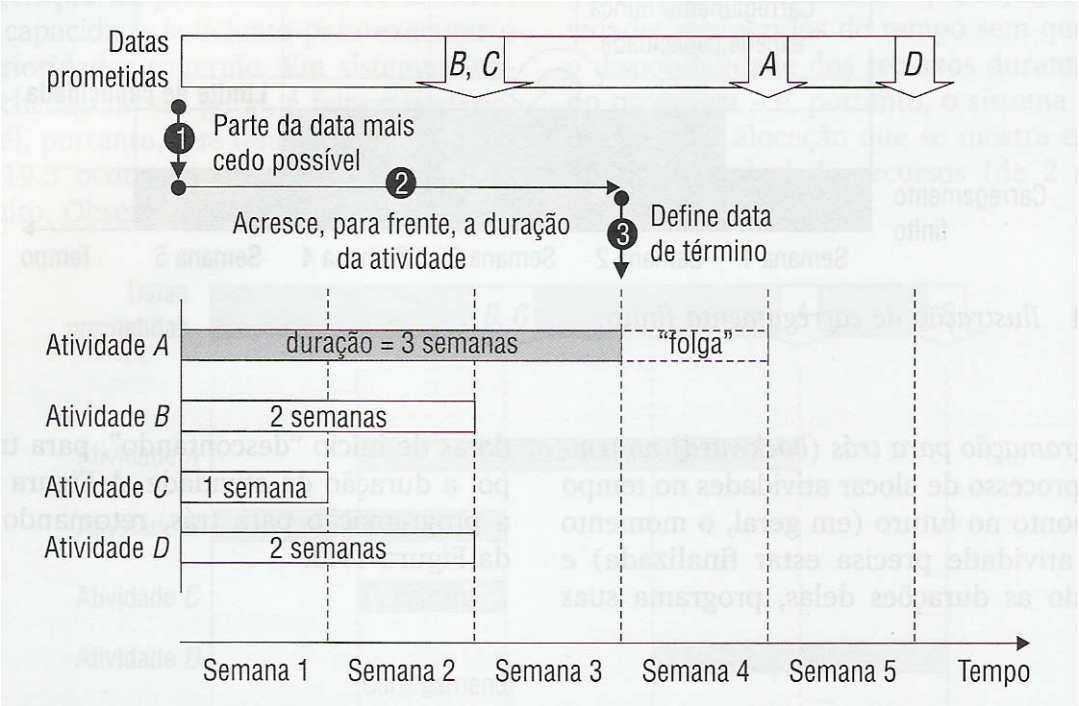

Figura 2.6 - Programação para frente
FONTE: CORRÊA & CORRÊA (2007, p. 584)

PROGRAMAÇÃO DE PRODUÇÃO

A programação detalhada da produção tem início após a tomada de decisão quanto à capacidade, níveis de estoques e pedidos a atender dentro de uma escala de tempo e, também, dependente do tipo de operação. Conforme MARTINS e LAUGENI (2003) existem quatro tipos de programação da produção:

- Programação de processos contínuos;
- Programação *Job-shop*;
- Programação de projetos;
- Programação de linhas de montagem.

Para o presente trabalho, somente é necessário a definição da programação *Job-shop* que é denominada por alguns autores por processos repetitivos em massa e tem por objetivo programar trabalhos ou ordens de produção, intermitentes e diversificadas, em um conjunto de máquinas.

> O *job shop* é caracterizado pela fabricação de produtos que devam passar por uma série de operações, que podem ter seqüências alternativas sem

> que isso comprometa o tempo de processo ou a qualidade do produto. (MARTINS e LAUGENI, 2003, p. 238)

Na programação *Job-shop* deve-se verificar a questão dos estoques em processo e estabelecer uma seqüência de trabalhos, detalhada em cada uma das máquinas ou dos processos de fabricação.

SEQUENCIAMENTO

Normalmente utiliza-se indistintamente a palavra sequenciar e programar com o mesmo sentido, porém segundo DARÚ (2005), existe uma diferença entre estas duas atividades.

> Sequenciar consiste em definir a ordem em que tarefas, operações ou atividades devem ser realizadas enquanto que programar consiste em alocar recursos ao longo do tempo para realizar um conjunto de tarefas (DARÚ, 2005, p. 34).

Segundo HEIZER e RENDER (2001, p. 432), a programação fornece uma base para alocação de tarefas a centros de trabalho. A alocação de carga é uma técnica de controle da capacidade que ressalta a sobrecarga e as ociosidades, e o seqüenciamento especifica a ordem em que as tarefas devem ser executadas em cada centro de trabalho.

CRITÉRIOS PARA AVALIAÇÃO DO DESENPENHO

O sequenciamento deve ter o seu desempenho avaliado frente a algumas premissas. Nesse sentido, MARTINS e LAUGENI (2003) definem que devem ser considerados os seguintes critérios:

- Porcentagem de ordens completadas na data certa;
- Distribuição estatística do tempo de atraso (média e desvio padrão dos atrasos);
- Porcentagem média de ordens aguardando processamento;
- Utilização da capacidade (máquinas e pessoas);
- Estoque médio em processo.

FATORES QUE AFETAM O SEQUENCIAMENTO

No caso da programação *job-shop*, múltiplas tarefas necessitam ser realizadas, passando por múltiplos centros de trabalho. Para tanto, elas têm de ser roteirizadas ao longo de seqüências de centros de trabalho para que possam ser completadas. Quando uma tarefa (uma ordem derivada de um pedido de um cliente, por exemplo) chega a um determinado centro de trabalho, ela entra numa fila, aguardando que algum centro de trabalho fique livre para que possa, então, ser preparado e executar a operação necessária. É preciso, assim, que a gestão da operação decida qual a posição na fila que a ordem merece e essa posição altera-se contínua e dinamicamente à medida que mais ordens cheguem ao centro do trabalho, levando em considerações várias variáveis, tais como:

Em termos de ordens:

- As ordens de produção apresentam datas de entrega diferentes, conforme o prometido pelos setores comerciais das organizações, buscando atender a solicitações dos clientes;
- Cada ordem, geralmente, está em um estado diferente de realização – para algumas, muitas operações ainda precisam ser feitas, para outras, poucas operações ainda faltam;
- As ordens podem apresentar *set-up* com tempos e atividades distintas. Em função da ordem anterior, às vezes, vale a pena colocar duas ordens em seqüência por terem a mesma preparação ou preparação similar;
- Cada ordem pode ter vários roteiros alternativos, dependendo das características tecnológicas dos equipamentos;
- Os roteiros alternativos podem ter produtividades diferentes;
- Cada ordem pode eventualmente ser feita em máquinas alternativas com eficiência diferentes;
- As ordens podem ser de clientes com importância relativa diferente.

Em termos de recursos:

- Máquinas quebram, bem como demandam manutenção, podendo não estar disponíveis em determinados momentos;
- Matérias-primas podem não estar sempre e confiavelmente disponíveis;
- Ferramentas podem não estar disponíveis;
- Funcionários podem faltar.

Em termos de operações:

- Problemas relacionados à quantidade às vezes ocorrem, requerendo retrabalho – isso quer dizer que uma ordem pode necessitar voltar a um centro onde já foi processada;
- Operações podem demandar tempo de pós-produção (cura, secagem, etc.);
- Operações podem ter restrições para a definição de tamanhos de lote – por exemplo, requerem quantidades mínimas de produção;
- Operações podem ser feitas em recursos gargalos, demandando máxima utilização, sempre que possível;
- Operações podem demandar a disponibilidade simultânea de diversos recursos, por exemplo, determinada máquina trabalhando com uma ferramenta ou operador especializado, sendo que essas disponibilidades devem ocorrer de forma simultânea.

TÉCNICAS PARA O SEQUENCIAMENTO

A solução dos problemas de sequenciamento pode ser obtida por meio de diversas técnicas referenciadas dentro da literatura e, entre elas destacam-se: regras de programação, pesquisa operacional e metaherísticas.

Regras de seguenciamento

Segundo TUBINO (1997) as regras de programação são heurísticas usadas para selecionar, a partir de informações sobre os lotes e sobre o estado do sistema produtivo, qual dos lotes esperando na fila de um grupo de recursos terá prioridade de processamento bem como qual recurso deste grupo será carregado com esta

ordem. Há várias regras de sequenciamento que são utilizadas em sistemas de gestão de operações, e algumas delas são ilustradas na Figura 2.7.

Regras de sequenciamento usuais para determinar prioridades em *job-shops*		
	Sigla	**Definição**
1	FIFO	*First In First Out* – primeira tarefa a chegar no centro de trabalho é a primeira a se atendida.
3	FSFO	*First in the System, First Out* – primeira tarefa a chegar à unidade produtiva é a primeira a ser atendida.
4	SOT	*Shortest Operation Time* – tarefa com o menor tempo de operação no centro de trabalho é a primeira a ser atendida.
5	SOT1	Mesma SOT, mas com limitante de tempo máximo de espera para evitar que ordens longas esperem muito.
6	EDD	*Earliest Due Date* – a tarefa com data prometida mais próxima é processada antes.
7	SS	*Static Slack* – folga estática, calculada como "tempo até a data prometida menos tempo de operação restante."
8	DS	*Dynamic Slack* – folga dinâmica, calculada como "folga estática dividida pelo número de operações por executar".
9	CR	*Critical Ratio* – razão crítica, calculada como "tempo até a data prometida dividido pelo tempo total de operação restante".

Figura 2.7 – Regras de sequenciamento

FONTE: CORRÊA & CORRÊA (2007, p. 581)

2.10.2 Pesquisa operacional

A pesquisa operacional caracteriza-se pelo uso de técnicas e métodos científicos qualitativos por equipes interdisciplinares, no esforço de determinar a melhor utilização de recursos limitados e para a programação otimizada das operações de uma empresa.

Segundo ANDRADE (2004) no enfoque clássico, derivado do quantitativo

> A Pesquisa Operacional é definida como a arte de aplicar técnicas de modelagem a problemas de tomada de decisão, e resolver os modelos identificados por meios de métodos matemáticos e estatísticos visando à obtenção de uma solução ótima, sob uma abordagem sistêmica (ANDRADE, 2004, p. 5).

Ainda, conforme o autor, no enfoque atual, derivado do conceito qualitativo

> O esforço despendido para a modelagem de um problema leva a uma compreensão mais profunda do próprio problema, identificando melhor seus elementos internos, suas interações com o ambiente externo, as informações necessárias e os resultados possíveis de obter (ANDRADE, 2004, p. 6).

2.10.3 Metaheurísticas

As metaheurísticas consistem de procedimentos destinados a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico. Têm caráter geral por aplicarem-se a quaisquer tipos de problemas e o sucesso depende de:

- Adaptação a instâncias especiais;
- Escapar de ótimos locais;
- Fazer uso da estrutura do problema;
- Estrutura eficiente de dados;
- Pre-processamento;
- Boas técnicas para construir soluções iniciais;
- Reinicializar procedimentos;
- Melhoria de solução através de busca local;
- Randomização controlada;
- Diversificação de busca quando nenhuma melhoria adicional parece possível;
- Intensificação da busca em regiões promissoras.

Dentre as metaheurísticas destaca-se: Busca Tabu, *Multi-start*, GRASP (*Greedy Randomized Adaptive Procedure*), Algoritmos Genéticos, *Variable Neighborhood Search* (VNS) e Colônia de Formigas (*Ant Colonies*).

Como já mencionado anteriormente, este trabalho, limita-se ao estudo dos algoritmos genéticos para o sequenciamento de terminais portuários.

3. ALGORITMOS GENÉTICOS

3.1 COMPUTAÇÃO EVOLUTIVA

A computação evolutiva define ferramentas para construir sistemas inteligentes aos moldes do comportamento inteligente com capacidade de criar ferramentas inteligentes com capacidade de criar ferramentas computacionais para resolver problemas práticos e são baseadas nos moldes dos mecanismos de evolução biológica e seleção natural (HEITKOETTER e BEASLEY, 1995).

A computação Evolutiva é um campo que compreende cinco áreas: Algoritmos Genéticos, Programação Evolutiva, Sistemas de classificação, Programação Genética e Estratégias de Evolução.

3.1.1 Programação evolutiva

A Programação Evolutiva (PE) foi originalmente concebida por Lawrence J. Fogel em 1960 como uma estratégia de otimização estocástica similar aos AGs, mas que enfatiza o relacionamento comportamental entre progenitores e sua descendência, ao invés de tentar emular operadores genéticos específicos observados na natureza. São algoritmos similares aos Algoritmos Genéticos, mas não implementam cruzamentos. Ao invés disso, eles confiam na aptidão para sobrevivência e mutação.

3.1.2 Sistemas de classificação

Os Sistemas de Classificação foram originalmente propostos por HOLLAND (1985) como sistemas capazes de perceber e classificar os acontecimentos seu ambiente e reagir a eles apropriadamente. Para a construção de tais sistemas é necessário: um ambiente, receptores que informam ao sistema sobre o que está ocorrendo, efectores que permitem ao sistema manipular o seu ambiente, e o sistema em si.

A idéia geral dos Sistemas Classificadores é começar sem nenhum conhecimento empregando uma população classificadora aleatoriamente gerada e deixar o sistema aprender o seu programa por indução. Isto reduz o fluxo de entrada a padrões de *inputs* recorrentes que podem ser repetidos diversas vezes permitindo classificar sua situação/contexto e reagir produzindo continuações apropriadas.

Toda a base de conhecimentos gerada é tratada através de regras se-então que são aplicadas sucessivamente.

3.1.3 Programação Genética

Foi projetada para evoluir geneticamente os programas de computador e, ela geralmente usa cromossomos com formato de membros individuais de uma população. Os paradigmas da Computação Evolutiva geralmente diferenciam-se da busca tradicional, e os paradigmas de otimização tem três áreas principais.

- Utilizam uma população de pontos (soluções potenciais) em suas pesquisas. Uso direto das funções de aptidão ao invés de funções derivadas ou outros relacionados ao conhecimento.
- Uso probabilístico ao invés de determinístico e transição de regras.

A Programação Genética (PG) é uma extensão no espaço de programas do modelo genético de aprendizado, isto é, os objetos que a constituem não são *strings* de caracteres de tamanho fixo que codificam possíveis soluções para um determinado problema. São na verdade programas, que quando executados oferecem diferentes soluções. Usualmente tais programas são representados em PG como árvores, ao invés de linhas de código.

3.1.4 Estratégias de evolução

O campo das Estratégias de Evolução (EE) foi concebido para solucionar problemas técnicos de otimização (PTO) e até recentemente era empregado quase que exclusivamente em engenharia civil, como uma alternativa às soluções convencionais. Normalmente não há uma função analítica objetiva fechada para os

PTO e portanto nenhum método aplicável de otimização além da intuição do engenheiro.

3.2 ALGORITMOS GENÉTICOS

Segundo LINDEN (2006, p. 38), os algoritmos genéticos (AG) são uma técnica de busca baseada numa metáfora do processo biológico da evolução natural, sendo considerados como heurísticas de otimização global. Os algoritmos genéticos utilizam-se dos operadores genéticos (seleção, recombinação e mutação), efetuando uma avaliação de uma das características de cada indivíduo e, eventualmente acabando por gerar um indivíduo que caracterizará uma boa solução ao problema

Conforme GOLBARG e LUNA (2005) os primeiros trabalhos nessa linha são originados de John Holland, em 1962 e 1970 (*apud* GOLBARG e LUNA, 2005), e objetivam reaplicar os processos utilizados pelos sistemas auto-adaptativos em contexto computacional. O trabalho de John Holland utilizou uma lista de símbolos binários (0,1) para representar as cadeias de ácido nucléico que fundamentaram uma teoria geral permitindo a aplicação imediata na determinação de máximos e mínimos de funções matemáticas. Na continuidade, diversos pesquisadores contribuíram para a consolidação dos algoritmos genéticos dentro da comunidade acadêmica.

Na figura 3.1 encontram-se a nomenclatura utilizada na linguagem natural e sua correspondência com a linguagem usualmente utilizada na comunidade acadêmica.

Linguagem natural	**Algoritmo Genético**
Cromossomo	Individuo, string, cromossomo, arvore
Gen	Característica
Alelo	Valor
Locus	Posição
Genótipo	Estrutura
Fenótipo	Conjunto de parâmetros

Figura 3.1 – Correspondência entre a linguagem natural e algoritmo genético

FONTE: LINDEN (2006, p. 44)

3.3 BREVE HISTÓRICO DOS ALGORITMOS GENÉTICOS APLICADOS A PROBLEMAS DE SEQUENCIAMENTO (*SCHEDULLING*)

DAVIS (1985) foi um dos primeiros pesquisadores que sugeriu a aplicação de algoritmos genéticos aos problemas de sequenciamento (*scheduling*). No artigo, ele observa a utilidade de métodos probabilísticos de busca local em ambientes com grandes áreas de busca e sugere uma representação indireta, no qual o AG opera sobre uma lista que tem que ser decodificada para formar a seqüência de produção (*schedulling*).

A partir desse fato, numerosas aplicações foram propostas, voltadas aos vários tipos de sequência de produção existentes e suas variações. A representação para uma classe de problemas também pode ser aplicada a outras classes. Porém pequenas modificações no problema podem exigir representações diferentes.

Em função da abordagem de representação do problema de *job-shop* em algoritmos genéticos, estes podem ser divididos em duas categorias, a representação direta e a representação indireta.

Na abordagem direta, a solução é codificada num cromossomo e o algoritmo genético é utilizado para evoluí-lo numa sequência de produção (*scheduling*) melhor.

NAKANO e YAMADA (1991) efetuaram uma das primeiras abordagens direta e criaram uma codificação binária baseada na relação de precedência das operações na mesma máquina. Uma estratégia chamada “forçar” (*forcing*) também é adotada, sendo que ela modifica um cromossomo se uma operação puder ser deslocada à esquerda.

Uma das abordagens mais diretas é a das chaves aleatórias (BEAN, 1994). Para NORMAN e BENA (1997), cada gene consiste de duas partes: um inteiro do conjunto {1, 2, ..., *m*} e uma fração gerada aleatoriamente de (0,1). A parte inteira do gene é a atribuição da máquina enquanto a parte fracionária, classificada em ordem não decrescente, determina a seqüência de operações em cada máquina.

As representações como a baseada em operações (*operation-based)* (KUBOTA, 1995), em trabalhos (*job-based*) (HOLSAPPLE *et al*, 1993), na relação de pares de trabalhos (*job-pair-relation-based*) (NAKANO e YAMADA, 1991), no tempo de conclusão (*completion time-based)* (YAMADA e NAKANO, 1992), e a com chaves aleatórias (*random key representation*) (BENA, 1994; NORMAN e BENA, 1995a; NORMAN e BENA, 1995b) pertencem a essa classe.

Na abordagem de representação indireta, uma seqüência de regras que qualificam os trabalhos é codificada num cromossomo e algoritmos genéticos são utilizados para evoluir esses cromossomos e conduzi-los a determinar a melhor seqüência de regras para designar os trabalhos. A programação (*schedule*) é então construída a partir destas regras. Nessa categoria de representação estão a representação baseada em lista de preferências (*preference-list-based representation*) (DAVIS, 1985), em lista de prioridades (*priority-rule-based representation*) (DORNDORF e PESCH, 1995), em grafo disjuntivo (*disjunctive-graph-based representation*) (TAMAKI and NISHIKAWA, 1992) e, a em máquinas (*machine-based representation*) (DORNDORF e PESCH, 1995).

FALKENAUER e BOUFFOUIX (1991) estenderam a abordagem de Davis e codificaram as operações a serem processadas numa máquina como uma lista de preferências que consiste numa string de símbolos. DELLA CROCE *et al.* (1995) também adotam esta estratégia de codificação e operador de *crossover*, mas com um método *look-ahead* a fim de gerar programações ativas (*active schedules*).

KOBAYASHI *et al.* (1995), efetuaram uma representação baseada em lista, onde um cromossomo é uma string de símbolos de tamanho n e cada símbolo identifica uma operação a ser processado numa maquina. TAMAKI e NISHIKAWA (1992) aplicam uma representação indireta baseada no grafo disjuntivo. Um cromossomo consiste numa string binária correspondendo a uma lista ordenada de preferências de arestas disjuntivas.

BIERWIRTH (1995) criou um algoritmo genético de permutação generalizada a fim de melhorar métodos existentes. Um cromossomo representa a permutação de trabalhos. Em outro artigo, BIERWIRTH *et al.* (1996) analisam três operadores de *crossover* que preservam a ordem relativa, posicional e absoluta de permutação das operações. Mais recentemente, SHI (1997) aplicou uma técnica de *crossover* que divide aleatoriamente um casal arbitrariamente escolhido em dois subconjuntos, dos quais os filhos são produzidos.

Vários trabalhos indicam que algoritmos genéticos não servem bem para ajuste fino de estruturas que estão muito próximas da solução ótima (DORNDORF e PESCH, 1995, BIERWIRTH, 1995) em virtude dos operadores de *crossover* geralmente perderem sua eficiência para gerar *schedules* praticáveis.

Para superar alguns desses problemas, uma forma de computação evolutiva conhecida como *Genetic Local Search* (GLS) ou busca local populacional ou busca

memética (GREFENSTETTE, 1987; MOSCATO, 1989; ULDER *et al.* 1991) é aplicada. O GLS incorpora vizinhanças de busca local na estratégia do algoritmo.

Dentro da estrutura GLS, um filho criado pelos operadores do algoritmo genético é usado como solução inicial para a subseqüente busca na vizinhança que perturba o filho para a mais próxima seqüência ótima local. O mínimo local é então colocado na próxima geração e modificado pelos operadores genéticos tradicionais de mutação e recombinação.

A superioridade do GLS sobre AGs é destacada por DELLA CROCE *et al.* (1994) que incorpora várias estruturas de vizinhança em seu algoritmo genético e fornece resultados melhores. Um dos mais conhecidos trabalhos em GLS é o de DORNDORF e PESCH (1995) que propõe duas abordagens para resolver os algoritmos genéticos. O primeiro método guia uma combinação probabilística de doze regras de prioridade de expedição e é chamado de P-GA, enquanto que a segunda abordagem, referida como SB-GA, controla uma seleção de nós para SBII, *partial enumeration tree*.

Resultados indicam que o SB-GA é superior ao P-GA. Uma estrutura similar de evolução do cromossomo foi aplicada por PESCH (1993) que utiliza uma estratégia GLS para controlar seleções de subproblemas na sua abordagem de decomposição. Os subproblemas são solucionados por uma abordagem de propagação de restrições que acha boas soluções fixando direções dos arcos.

YAMADA e NAKANO (1995b) geram dois *schedules* para se reproduzirem, o mais distante possível, de acordo com a distância do grafo disjuntivo. Começando a busca no primeiro genitor substituem iterativamente uma solução na população atual com uma seqüência melhorada inclinada para o segundo genitor. Melhorias adicionais a esse método foram feitas por YAMADA e NAKANO (1996b, 1996c) no qual um sistema estocástico inclinado de recolocação, *biased stochastic replacement scheme*, que favorece soluções mais próximas do segundo genitor e uma estratégia de bloco crítico de vizinhança são aplicados. Ambos os métodos são baseados na idéia de *path relinking* na busca tabu (GLOVER e LAGUNA,1997).

Segundo PETROVICK (2005, *apud* LINDEN, 2006) resolve o problema usando uma estrutura de cromossomos em duas camadas, dividindo as tarefas e as máquinas em conjuntos, designando um conjunto de tarefas para um conjunto de máquinas. No primeiro nível associa um conjunto de tarefas a um conjunto de

máquinas e o segundo nível tem uma posição para cada tarefa na máquina em que ela será executada.

3.3 CONCEITOS BÁSICOS DE ALGORITMOS GENÉTICOS

O funcionamento dos AGs é decomposto nas etapas de inicio, avaliação, seleção, cruzamento, mutação, atualização e fim. De uma maneira geral os algoritmos genéticos podem ser representados pelo fluxograma apresentado na figura 3.2.

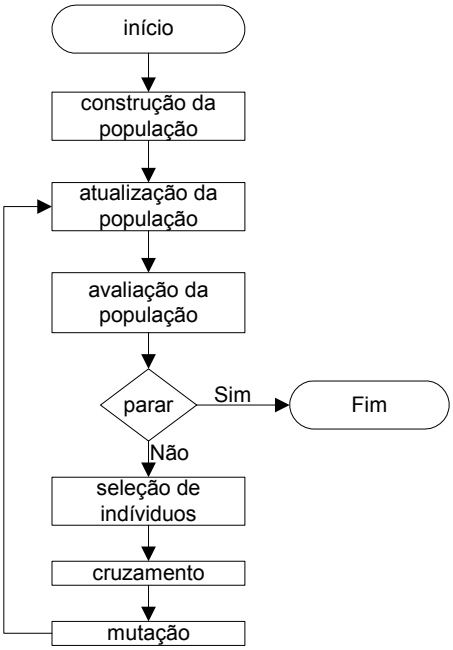

Figura 3.2- Fluxograma que descreve um AG

Basicamente, o que um algoritmo genético faz é criar uma população de possíveis respostas para o problema a ser tratado para, em seguida, submetê-la ao processo de evolução, constituído pelas seguintes etapas:

- Avaliação: avalia-se a aptidão das soluções (indivíduos da população) — é feita uma análise para que se estabeleçam quão bem elas respondem ao problema proposto.
- Seleção: indivíduos são selecionados para a reprodução. A probabilidade de uma dada solução *i* ser selecionada é proporcional à sua aptidão.

- Cruzamento: características das soluções escolhidas são recombinadas, gerando novos indivíduos.
- Mutação: características dos indivíduos resultantes do processo de reprodução são alteradas, acrescentando-se, assim, variedade à população.
- Atualização: os indivíduos criados através do cruzamento e da mutação são inseridos na população, gerando uma população nova.
- Finalização: verificam se as condições de encerramento da evolução foram atingidas, retornando para a etapa de avaliação em caso negativo e, encerrando a execução em caso positivo.

De forma geral, atendem às características gerais da teoria evolucionária, que já são amplamente aceitas:

- A seleção natural é um processo que atua sobre os cromossomos e, portanto, sobre os seres vivos que eles codificam.
- A seleção natural é o elo entre cromossomos e o desempenho das suas estruturas decodificadas. O processo de seleção natural faz com que os cromossomos que codificam estruturas bem sucedidas se reproduzam mais vezes e com maior probabilidade do que as estruturas mal sucedidas.
- O processo de reprodução é o ponto no qual a evolução acontece. Mutações podem provocar mudanças nos cromossomos dos filhos, fazendo com que eles sejam diferentes dos padrões genéticos dos seus pais, e processos de recombinação podem criar diferentes cromossomos para os filhos, pela combinação dos cromossomos dos pais.
- A evolução biológica não tem memória. Tudo o que se sabe sobre como produzir indivíduos bem adaptados ao seu meio ambiente está contido no seu genoma - o conjunto de cromossomos carregados pelos indivíduos da população atual – e na estrutura dos cromossomos decodificados.

3.4 REPRESENTAÇÃO DO INDIVÍDUO

A representação normal dos indivíduos é binária (0,1), porém outras representações podem ser utilizadas em função do tipo do problema e das respostas desejadas. Nos casos de otimização combinatória deve-se ter uma representação

baseada em lista com todos os elementos presentes no problema colocados em determinada ordem (adaptado de LINDEN, 2006).

Utilizando a representação fenotípica, em lista, têm-se os seguintes casos:

- (A D G F G C E B) e (G F B C A H E D) são indivíduos válidos para um problema envolvendo sete nós;
- (A F D G E C H) não é um indivíduo válido, visto que o elemento B não está presente na lista;
- (F H B C G A E D D) não é um indivíduo factível, visto que o elemento D aparece duas vezes.

3.5 OPERADOR RECOMBINAÇÃO OU CRUZAMENTO

Para os caso dos algoritmos genéticos pode-se efetuar diferentes tipos de *crossover*, tal como *crossover* dois pontos, *crossover* baseado em maioria, etc, porém, conforme ROCHA, *et al* (2000), existem operadores genéticos específicos para o caso de problemas de otimização combinatória:

- *Crossover* com ordem preservada (*OPX - Ordem Preserving Crossover)*. Este operador preserva a ordem relativa dos genes de ambos os pais. O algoritmo executa um corte aleatório, levando do Pai 1 todos os genes até o ponto de corte na ordem em que estão e, completando o filho com os genes com os genes do Pai 2, mantendo a ordem relativa que estes ocupam;
- *Crossover* de ordem uniforme preservada (*UOPX - Uniform Order Preserving Crossover*). Este operador é similar ao anterior. Trabalha como uma máscara binária para definir os genes do Pai 1 e do Pai 2 que comporão o filho. Os genes que na mascara binária são identificados com “1” são do Pai 1 e os genes identificados na mascara binária com “0” são genes do Pai 2, porém preservada a ordem realtiva destes;
- Emparelhamento parcial de *crossover* (*PMX - Partially Matched Crossover*). Escolhe dois pontos de corte aleatoriamente e efetua a troca de posição para posição dos genes entre os dois pais;

- *Crossover* cíclico (*CYCX – Cycle Crossover*). Executa a recombinação considerando a restrição existente de cada gene tem de vir de um pai ou de outro;
- *Crossover* de extremidade (*EDGX – EDGe Crossover*). Este *crossover* é baseado no principio de conservação dos genes de adjacência. Ele trabalha colecionado informações da vizinhança de cada um dos genes dos pais e procura construir um filho com estas informações;

Tem-se especial interesse no *crossover* uniforme, que segundo LINDEN (2006) o seu funcionamento é descrito como: para cada gene é sorteado um número, zero ou um. Se o valor sorteado for igual a um, o filho número um recebe o gene da posição corrente do primeiro pai e o segundo filho recebe um gene corrente do segundo pai. Por outro lado, se o valor sorteado for zero, as atribuições serão invertidas: o primeiro filho recebe o gene da posição corrente do segundo pai e o segundo filho recebe o gene corrente do primeiro pai.

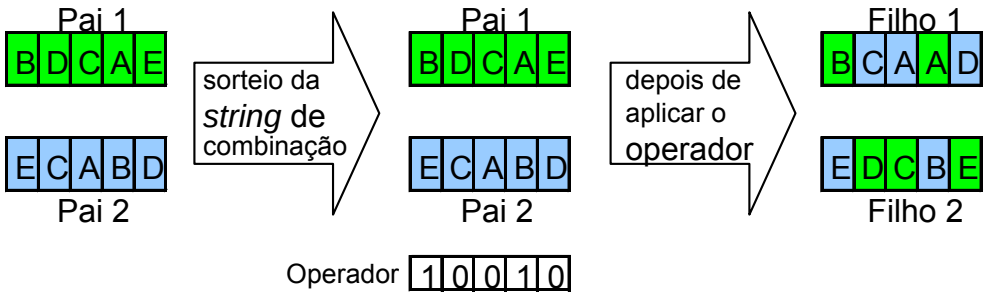

Figura 3.3 - Funcionamento de um *crossover* uniforme
Adaptado de LINDEN (2006)

Na figura 3.3, tem-se a representação do funcionamento do *crossover* uniforme, e pode-se observar que os filhos gerados apresentam duas situações inválidas. No filho 1, o elemento A foi repetido e não aparece o elemento E, enquanto que, no filho 2, o elemento E foi repetido e não aparece o elemento A, o que caracteriza que nenhum dos filhos é um indivíduo válido.

Para contornar o problema e gerar filhos factíveis algumas alterações devem ser feitas no *crossover* uniforme, assim o pseudo-código, da figura 3.4 representa o *crossover* baseado em ordem em que os filhos gerados são indivíduos válidos.

Passo 1	Gere uma seqüência de bits aleatória do mesmo tamanho que os elementos
Passo 2	Copie para o filho 1 os elementos do pai 1 referentes àquelas posições onde a *string* de bits possui 1
Passo 3	Faça uma lista dos elementos do pai 1 referentes a zeros da *string* de bits
Passo 4	Permute esta lista de forma que os elementos apareçam na mesma ordem que no pai 2
Passo 5	Coloque estes elementos nos espaços do filho 1 na ordem gerado no passo anterior
Passo 6	Repita o processo para gerar o filho 2, substituindo o pai 1 pelo pai 2

Figura 3.4 - Pseudo-código do *crossover* baseado em ordem

Fonte: LINDEN (2006, p. 177)

Na figura 3.5 tem-se a representação do *crossover* baseado em ordem, obtido através do pseudo-código descrito anteriormente. Pode-se observar que o indivíduo (filho) gerado é válido, pois apresenta todos os elementos, porém em posições diferentes, representado, dessa forma, outra solução para o problema.

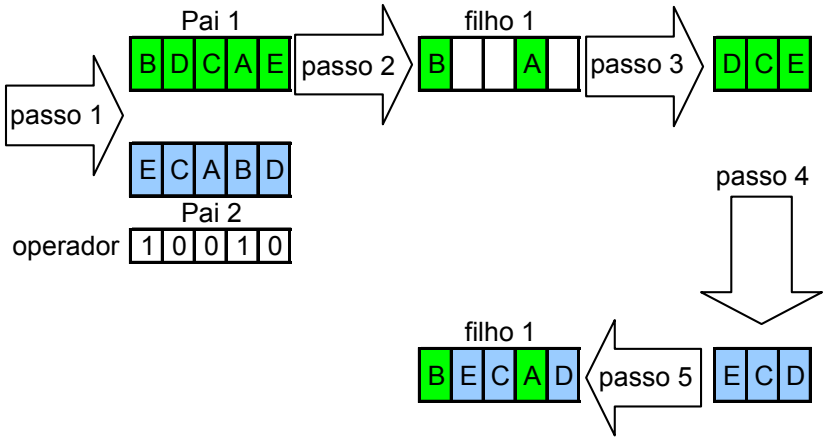

Figura 3.5 - Exemplo de atuação do *crossover* baseado em ordem

3.6 OPERADOR DE MUTAÇÃO

O operador de mutação é fundamental para um algoritmo genético, pois é ele que garante a continuidade de diversidade genética na população. Basicamente, o operador de mutação é uma heurística exploratória que injeta novos indivíduos na população e permite que os AGs busquem soluções fora dos limites da população inicial (VOSE, 2004).

Caso o valor da probabilidade atribuída ao operador de mutação for muito baixo, o algoritmo genético estagnará muito rápido, visto que a população não terá diversidade. Por outro lado, se for muito alto, o AG perde suas principais características e passa a se assemelhar a um algoritmo aleatório. (adaptado de LINDEN, 2006)

O princípio básico dos operadores de mutação reside na efetivação de mudanças locais. Porém, nos casos de indivíduos baseados em ordem, não há bits a inverter e não é possível designar valores aleatoriamente, pois podem ocorrer repetições ou alguns elementos podem ficar de fora gerando indivíduos inválidos.

Segundo LINDEN (2006, p 115), a maioria dos trabalhos, nas áreas de algoritmos genéticos usa um valor de 0,5% a 1%, devido a razões históricas. Para os casos de otimização de cromossomos binários, a taxa de mutação é de 1/*L*, sendo *L* é igual ao número de variáveis binárias.

Para o caso de cromossomos baseados em ordem não se localizou na literatura algum critério para determinar a taxa de mutação, apenas é referenciado que a taxa ótima de mutação depende da representação que está sendo utilizada conforme menciona DEB (1998, *apud* LINDEN, 2006).

Existem diversos operadores de mutação para indivíduos baseados em ordem como, por exemplo, a recombinação de arestas e o mapeamento parcial. Os operadores que tenham funções de otimização embutidas têm sido alvo de recentes pesquisas. Finalmente, existem operadores que desempenham simultaneamente a função do *crossover* e mutação como o caso do operador denominado *inver-over*.

A análise dos operadores de mutação será limitada às três formas básicas para efetuar a mutação em indivíduos: a permutação, a mistura de sublista e a inversão de sublista.

3.7 PERMUTAÇÃO DE ELEMENTOS

A permutação de elementos é simples; escolhem-se dois elementos, aleatoriamente, dentro do indivíduo e trocam-se as suas posições. Na figura 3.5 tem-se a representação do operador de mutação por permutação de elementos.

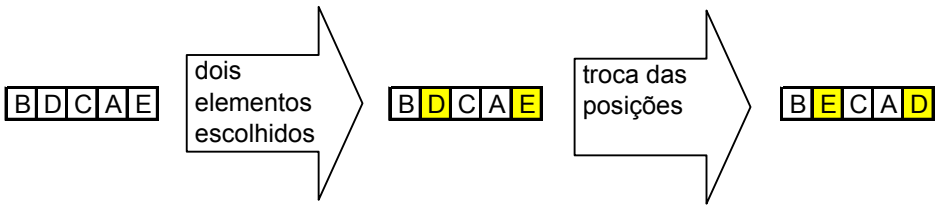

Figura 3.6 - Operador de mutação – permutação

3.7.1 Mistura de Sublista

O operador de mutação que utiliza a sublista é igualmente simples. Escolhem-se dois pontos aleatoriamente dentro do indivíduo para delimitar uma sublista, e efetua-se uma mutação dos elementos da sublista. Um exemplo de um operador de mutação baseado em sublista é mostrado na figura 3.6.

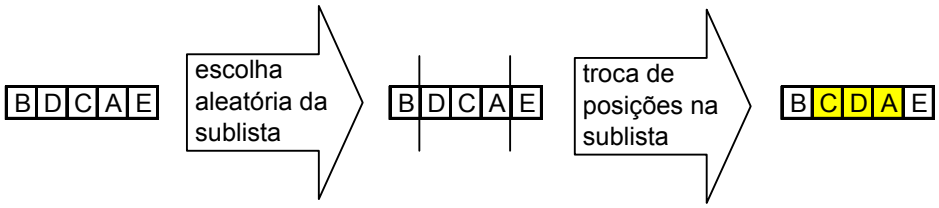

Figura 3.7 - Operador de mutação baseado em mistura de sublista

3.7.2 Inversão de Sublista

O operador de mutação por meio da inversão de sublista consiste em inverter a ordem da sublista sorteada. A figura 3.7 apresenta um exemplo desse operador de mutação.

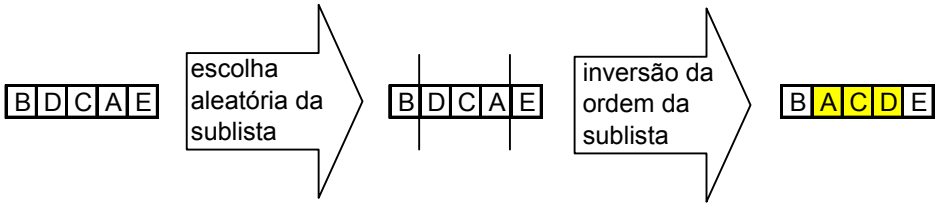

Figura 3.8 - Operador de mutação baseado em inversão de sublista

3.8 FUNÇÃO DE AVALIAÇÃO (*FITNESS*)

A função de avaliação exerce no AG papel similar ao papel do meio ambiente na teoria da evolução natural das espécies e pode ser mais facilmente entendida como sendo a nota dada a um indivíduo na resolução do problema. Esta nota será utilizada para a seleção dos pais na próxima geração, sendo a forma de diferenciar as boas das más soluções (LINDEN, 2006).

Para CATARINA (2006), o elemento de ligação entre o AG e o problema a ser resolvido é a função de avaliação. A função de avaliação, denominado de função *fitness*, toma como entrada um indivíduo (cromossomo) e retorna um número, ou lista de números, que representam a medida de desempenho do indivíduo com relação ao problema a ser resolvido.

Atualmente, várias formas de avaliação são utilizadas: em casos de otimização de funções matemáticas, o próprio valor de retorno destas tende a ser escolhido e, em problemas com muitas restrições, funções baseadas em penalidades são mais comuns. A função de avaliação também é chamada de função objetivo em um grande número de trabalhos.

3.9 SELEÇÃO POR MONTE CARLO

A mais conhecida e utilizada forma de fazer a seleção é a roleta viciada, ou algoritmo Monte Carlo (DAVIS, 1996). Na seleção por meio do algoritmo Monte Carlo, também conhecida como seleção por roleta viciada, cada indivíduo da população é representado numa roleta proporcionalmente ao seu índice de aptidão. Assim, aos indivíduos com alta aptidão é dada uma porção maior da roleta, enquanto aos de aptidão mais baixa é dada uma porção relativamente menor da roleta (adaptado de CATARINA, 2006).

Finalmente, a roleta é “girada” e são escolhidos os indivíduos que participarão da próxima geração. Um exemplo de aplicação do método da roleta é apresentado na tabela 3.1.

TABELA 3.1- EXEMPLO DO MÉTODO DA ROLETA

indivíduo	avaliação (*fitness*)	avaliação relativa
E C A B D	1	1,61%
B D C A E	9	14,52%
B E C A D	16	25,81%
B C D A E	36	58,06%
Total	62	100,00%

A roleta viciada para o grupo da tabela 3.1 é demonstrada na figura 3.9.

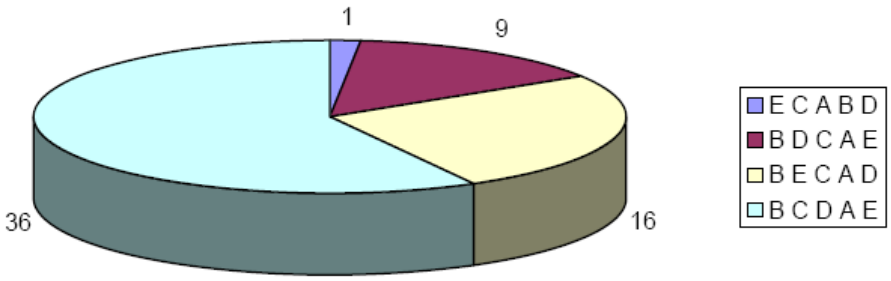

Figura 3.9 - Roleta viciada para os indivíduos da tabela 3.1
FONTE: O autor (2007)

O indivíduo (B C D A E) recebe um pedaço igual a 36/62≈58% e, ao rodar a roleta ele tem a probabilidade aproximada de ser selecionado três vezes a cada cinco sorteios. Em contrapartida o indivíduo (E C A B D) recebe um pedaço igual a 1/36≈1,6% e, ao rodar a roleta este indivíduo tem a chance de ser selecionada uma vezes a cada 36 sorteios (LINDEN, 2006).

3.10 ELITISMO

O elitismo é uma técnica que pode ser utilizada para melhorar a convergência dos algoritmos genéticos. Ele foi primeiramente introduzido por Kenneth De Jong, em 1975 (*apud* LINDEN 2006) e, é uma adição aos métodos de seleção que força os AGs a reterem certo número de "melhores" indivíduos em cada geração (YEPES, 2000). Tais indivíduos podem ser perdidos se não forem selecionados para reprodução ou se forem destruídos por cruzamento ou mutação.

O elitismo seleciona os melhores cromossomos de uma população e os transporta à geração seguinte. Esta técnica consiste basicamente em realizar o processo de seleção em duas etapas:

a) Seleciona-se uma elite de *r* membros entre os melhores da população inicial, os quais são incorporados diretamente na população final.
b) O restante da população final é obtido a partir dos (*n* - *r*) elementos restantes da população inicial de tamanho *n*.

Em geral, a elite tem um tamanho reduzido, com *r* = 1 ou 2 para *n* = 50. Quando é utilizada a técnica do elitismo, o algoritmo converge mais rapidamente. Como na natureza, os indivíduos mais aptos podem, além de se reproduzirem mais, terem uma vida mais longa, muitas vezes, sobrevivendo de uma geração para a outra. O efeito negativo desta estratégia prende-se ao fato de que a população inicial pode convergir para uma população homogênea de superindivíduos, não explorando outras soluções (CATARINA, 2006).

3.11 ESQUEMAS

Os esquemas são gabaritos que descrevem um subconjunto dentre o conjunto dos elementos possíveis, mais especificamente, descrevem a similaridade entre os indivíduos que pertencem a este subconjunto. Basicamente, identificam quais posições dos seus genomas são idênticas.

Um esquema consiste numa descrição mais grosseira, com menos detalhes da representação, o que permite que múltiplos indivíduos possam se adequar a ela em um mesmo instante.

Conforme LINDEN (2006), o esquema é definido como sendo uma *string* $s=\{s_1,s_2...s_1\}$, de comprimento *n*, cujas as posições pertencem ao conjunto Γ (alfabeto usado) + {∗} (símbolo de *wildcard,* que significa que não importa). Cada posição da *string* dada por $s_k \neq$ '∗' é chamada de especificação, enquanto que o *wildcard* representa o fato de que aquela posição pode assumir qualquer valor dentro do conjunto.

A quantidade de esquemas presentes em um determinado individuo é dependente do comprimento da *string* e do número de opções presentes no alfabeto

de codificação. O número de esquemas na população é n^t, sendo n a quantidade de símbolos utilizadas e t o tamanho da *string*. Uma *string x* satisfaz um esquema se todo o símbolo s_k pertence à *string s* definidora do esquema diferente do símbolo *wildcard,* tem-se $s_k=x_k$. Isto garante que em toda posição k (k=1,2,3,...,n) que não é igual ao símbolo *wildcard*, a *string x*, contém o mesmo caractere que *s*.

De uma forma geral, o AG é um manipulador de esquemas. São os esquemas que contém as características positivas ou negativas que podem levar a uma boa ou má avaliação e o AG probabilisticamente propaga os bons esquemas por toda a população durante a sua execução.

As características de um esquema são: i) ordem, denotada por $O(H)$, que corresponde ao número de posições neste esquema diferente de {∗}, ii) e o tamanho, denotado por $\delta(H)$, que se refere ao número de pontos de corte entre a primeira e a última posição diferente de ∗ dentro do esquema.

O quadro 3.2 mostra exemplos de esquemas e de indivíduos que representam estes esquemas, sua ordem e seu tamanho, considerando o conjunto $\Gamma = \{a,b,c,d,e\}$.

Esquema	**Indivíduos que representa**	**Ordem**	**Tamanho**
a∗	*ab, ac, ad, ae*	1	0
a∗b	*acb, adb, aeb*	2	2
a∗∗∗b	*acdeb, adecb, ...*	2	4
a∗∗b∗	*acdbe, adebc, ...*	2	3
abcde	*abcde*	5	4

QUADRO 3.1 - Exemplo de esquemas

4 SEQUENCIAMENTO DE CARREGAMENTOS DE NAVIOS GRANELEIROS

Neste capítulo, descreve-se o problema com profundidade enquadrando-o dentro da teoria da Administração de Produção. Além disso, serão efetuados alguns comentários com relação à quantidade de soluções que o problema pode apresentar.

4.1 CONSIDERAÇÕES SOBRE O SEQUENCIAMENTO

Os berços de atracação podem ser considerados como centros de trabalho, o serviço de carregamento como o produto e, assim, o carregamento de navios graneleiros pode ser visto como um problema *job-shop*. É importante destacar os seguintes objetivos na solução do problema de sequenciamento de carregamento de navios graneleiros:

a) Evitar a ocorrência de embarque simultâneo de um mesmo TP por dois, ou mais navios, o que caracteriza que a programação a ser efetuada deve respeitar as capacidades de carregamento. Desta forma a programação é finita.
b) Eliminar ou minimizar a quantidade de horas paradas durante o processo de embarque e, conseqüentemente, liberar o navio carregado o mais cedo possível.
c) Não há data limite para o embarque e este se inicia assim que o navio atracar, o que caracteriza a programação *forward*.

4.2 CONFLITO E DESLOCAMENTO

Os conflitos ocorrem numa seqüência de embarque quando um TP está definido para efetuar o carregamento em dois ou mais navios no mesmo momento temporal. Para facilitar a compreensão dos conflitos, utiliza-se como exemplo os dados reais de uma necessidade de carregamento de três navios ocorrida em março de 2007. Os dados estão expressos na tabela 4.1.

TABELA 4.1 - EXEMPLO DE CARREGAMENTO (TONELADAS)

	navio 1	**navio 2**	**navio 3**
Terminal portuário	**berço 212**	**berço 213**	**berço 214**
A			
B	21.545		
C		30.000	40.033
D			
E		4.500	12.000
F			
G	8.000		
H	10.401	666	
I	18.149	13.206	2.121
Total	**58.095**	**48.372**	**54.154**

FONTE: APPA – Administração dos Portos de Paranaguá e Antonina

A partir dos dados da tabela 4.1, foi elaborada uma seqüência de carregamento, de acordo com uma das técnicas mais simples de sequenciamento de produção a regra denominada SOT (*Shortest Operation Time*). Nesta regra a tarefa com o menor tempo de operação no centro de trabalho é a primeira a ser atendida. Considerando a capacidade de carga de cada um dos terminais, obtém-se a tabela 4.2, que representa o tempo de carregamento em cada um dos terminais.

TABELA 4.2 - TEMPOS NECESSÁRIOS PARA CARREGAMENTO (HORAS)

		navio 1	**navio 2**	**navio 3**
Terminal portuário	**capacidade (t/h)**	**berço 212**	**berço 213**	**berço 214**
A	500			
B	2000	10,77		
C	2000		15,00	20,02
D	2000			
E	2000		2,25	6,00
F	2000			
G	2000	4,00		
H	2000	5,20	0,33	
I	1000	18,15	13,21	2,12
	Total	**38,12**	**30,79**	**28,14**

FONTE: O autor (2007)

A seqüência de carregamento passa a ser {H_2, I_3, E_2, G_1, H_1, E_3, B_1, I_2, C_2, I_1, C_3}. Nesta representação o índice 1 representa o navio 1 atracado no berço 212, o índice 2 representa o navio 2 atracado berço 213 e o índice 3 representa o navio 3 atracado no berço 214.

TABELA 4.3 – SEQUÊNCIA DE CARREGAMENTO DO EXEMPLO

	Sequência de carregamento			Navio 1		Navio 2		Navio 3		Conflitos (h)			
TP	**Navio 1**	**Navio 2**	**Navio 3**	**Início**	**Fim**	**Início**	**Fim**	**Início**	**Fim**	**1∩2**	**1∩3**	**2∩3**	**TOTAL**
A				-	-	-	-	-	-	0,00	0,00	0,00	0,00
B	3			9,20	19,97	-	-	-	-	0,00	0,00	0,00	0,00
C		4	3	-	-	15,79	30,79	8,12	28,14	0,00	0,00	12,35	12,35
D				-	-					0,00	0,00	0,00	0,00
E		2	2	-	-	0,33	2,58	2,12	8,12	0,00	0,00	0,46	0,46
F				-	-	-	-	-	-	0,00	0,00	0,00	0,00
G	1			0,00	4,00	-	-	-	-	0,00	0,00	0,00	0,00
H	2	1		4,00	9,20	0,00	0,33	-	-	0,00	0,00	0,00	0,00
I	4	3	1	19,97	38,12	2,58	15,79	0,00	2,12	0,00	0,00	0,00	0,00
										0,00	0,00	12,81	12,81

FONTE: O autor (2007)

Na tabela 4.3, os valores estão expressos em horas, e considerou-se o início das operações no momento zero, verifica-se que o carregamento do TP “C”, nos navios 2 e 3, produz conflito de 12,35 horas obtido por meio da intersecção dos intervalos, ou seja:

$$[15{,}79 \quad 30{,}79] \cap [8{,}12 \quad 28{,}14] \Rightarrow [15{,}79 \quad 28{,}14] \Rightarrow 12{,}35h \tag{1}$$

Tal fato, também, ocorre no carregamento do terminal E, nos navios 2 e 3, sendo o conflito de 0,46 horas, obtido através da intersecção:

$$[0{,}33 \quad 2{,}58] \cap [2{,}12 \quad 8{,}12] \Rightarrow [2{,}12 \quad 2{,}58] \Rightarrow 0{,}46h \tag{2}$$

Da análise dos conflitos observados, conclui-se que o conflito ocorre quando o início de carregamento do TP de um navio ocorre antes do término do outro navio. Os conflitos devem ser analisados nas duas direções: do navio$_i$ para o navio$_{i+1}$ e do navio$_{i+1}$ para o navio$_i$. A figura 4.1 apresenta o sequenciamento na forma do gráfico de Gantt. Na parte inferior do gráfico os conflitos estão representados pelas áreas hachuradas.

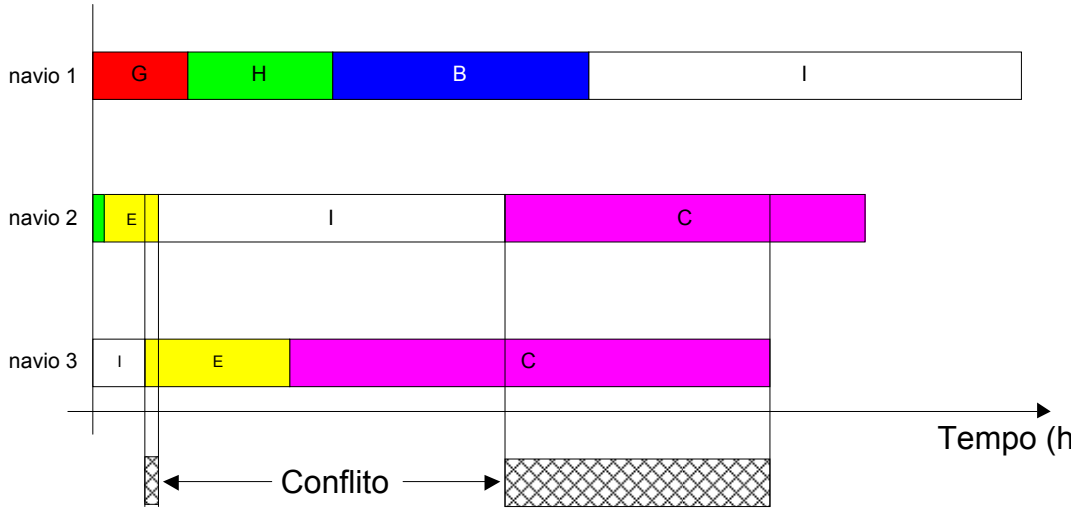

Figura 4.1 - Gráfico de Gantt do sequenciamento do exemplo

A forma proposta para a solução dos conflitos é deslocar o início dos intervalos que possuem conflito. Na tabela 4.4 apresentam-se os horários da tabela 4.3 deslocados (valores destacados). Desta forma não ocorrem mais conflitos, ocorrem períodos de inatividade.

TABELA 4.4 - SEQUÊNCIA DE CARREGAMENTO DO EXEMPLO COM DESLOCAMENTO

	Sequência de carregamento			**navio 1**		**navio 2**		**navio 3**		**Períodos de inatividade**			
TP	**Navio 1**	**Navio 2**	**Navio 3**	**Início**	**Fim**	**Início**	**Fim**	**Início**	**Fim**	**Navio 1**	**Navio 2**	**Navio 3**	**TOTAL**
A				-	-	-	-	-	-	0,00	0,00	0,00	0,00
B	3			9,20	19,97	-	-	-	-	0,00	0,00	0,00	0,00
C		4	3	-	-	**28,60**	43,60	8,58	28,60	0,00	12,81	0,00	12,81
D				-	-	-	-	-	-	0,00	0,00	0,00	0,00
E		2	2	-	-	0,33	**2,58**	2,58	8,58	0,00	0,00	0,46	0,46
F				-	-	-	-	-	-	0,00	0,00	0,00	0,00
G	1			0,00	4,00	-	-	-	-	0,00	0,00	0,00	0,00
H	2	1		4,00	9,20	0,00	0,33	-	-	0,00	0,00	0,00	0,00
I	4	3	1	19,97	38,12	2,58	15,79	0,00	2,12	0,00	0,00	0,00	0,00
										0,00	12,81	0,46	13,27

FONTE: O autor (2007)

É importante destacar que o período total de inatividade não é igual ao período total de conflitos, em virtude dos deslocamentos efetuados acabarem por afetar todo o sistema. Além disso, quanto mais para o início do sequenciamento

forem feitos os deslocamentos dos embarques, maiores serão os impactos, pois as seqüências posteriores serão igualmente afetadas. Finalmente, também pode ocorrer que operações que não apresentavam conflitos passem a apresentá-los em função do deslocamento efetuado. A figura 4.2 exibe o deslocamento efetuado, e as áreas hachuradas representam os períodos de inatividade.

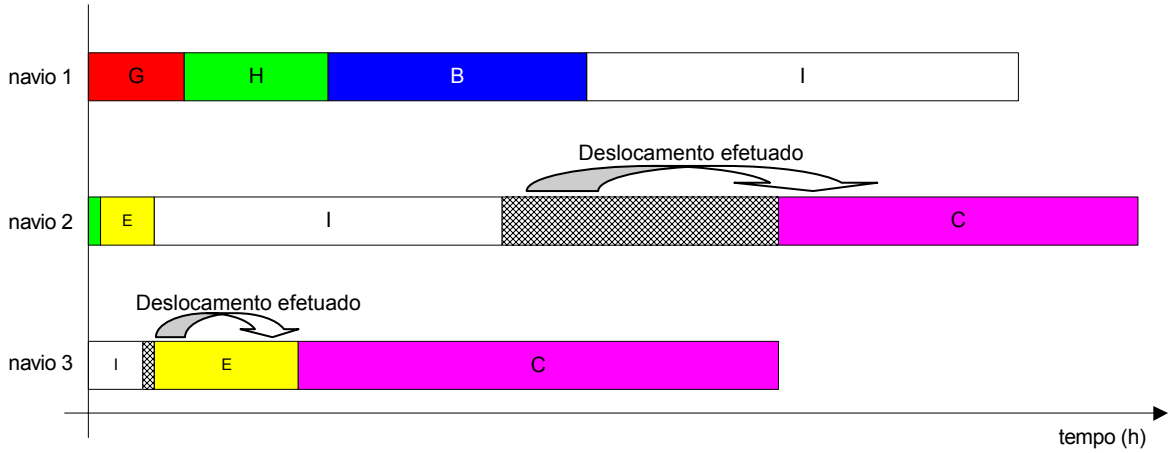

Figura 4.2 - Gráfico de Gantt do sequenciamento com deslocamentos.

4.3 DIMENSÕES DO PROBLEMA

Segundo RIBEIRO (2005) o problema *Job-shop Scheduling* é classificado na literatura como NP-difícil, sendo um problema para o qual não existem algoritmos exatos que o resolva, pois se trata de um problema de otimização combinatória. REIS (1996) afirma que, devido à inerente complexidade, muitos problemas de otimização, são NP-difíceis e nem sempre é possível para esses algoritmos encontrarem a solução ótima em tempo adequado.

Neste mesmo sentido, LINDEN (2006, p. 171) afirma que: “os problemas de otimização combinatória são, em geral, NP-completos, visto que seu espaço de busca é praticamente infinito”.

Num primeiro momento, pode-se supor que as soluções que apresentem o mesmo TP na mesma posição devam ser descartadas, entretanto, isto não é verdade para todos os casos. A única situação em que tal descarte deve ser feito é quando ocorre a coincidência na primeira posição (primeiro TP a ser embarcado) de cada navio. Nas demais posições não significam necessariamente, que o conflito vá

ocorrer em virtude dos tempos de embarque serem, na maioria dos casos, distintos. A figura 4.3 mostra, claramente, esta possibilidade.

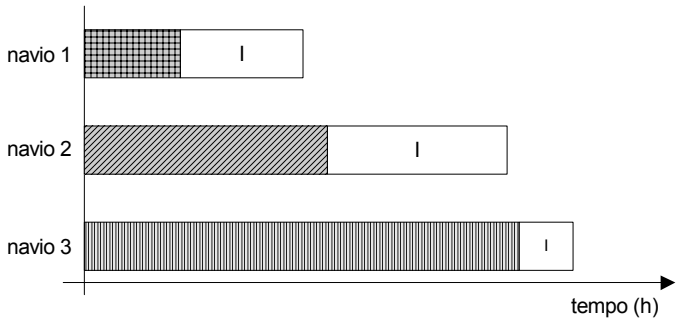

Figura 4.3 - Contra exemplo de programação com carregamento do mesmo terminal na segunda posição.

Na figura 4.3 observa-se que, em função dos diferentes tempos de carregamento utilizados na primeira posição, pode-se manter o carregamento do TP *I* na segunda posição sem, contudo, gerar conflitos. Em virtude da possibilidade de carregamento de um mesmo TP na mesma ordem ser permitida a partir da segunda posição e, da pequena proporção de soluções inviáveis em função do carregamento simultâneo de um mesmo TP na primeira posição, optou-se por considerar todas as soluções como viáveis. Neste caso, assume-se, como situação extrema, que os três navios vão embarcar mercadorias dos nove TPs.

Em função da consideração a ser adotada na solução do problema, a quantidade de soluções possíveis de *n* terminais efetuando o carregamento em m navios, apresenta o universo de solução $(n!)^m$, o que no presente caso assume $9!^3$. (CORRÊA e CORRÊA, 2007, p. 577)

5 SEQUENCIAMENTO DE CARREGAMENTO DE NAVIOS GRANELEIROS UTILIZANDO AG

O conceito utilizado pelos AGs é simples, porém é necessário projetá-lo cuidadosamente de forma a garantir o correto funcionamento da estrutura. As definições efetuadas em relação à representação do indivíduo, aos operadores, à função de avaliação, e à seleção devem ser visualizadas sempre como um conjunto, pois se o conjunto for equilibrado serão obtidos resultados satisfatórios.

5.1 REPRESENTAÇÃO DOS INDIVÍDUOS

O primeiro aspecto a ser considerado é a representação a ser utilizada nos indivíduos. No caso do sequenciamento de navios graneleiros é preciso identificar: o berço em que o navio está atracado, os TPs dos quais vai embarcar mercadoria e o tempo de embarque de cada TP em cada navio.

Optou-se por fazer uma abordagem direta. Cada terminal portuário é identificado pelas letras {A, B, C, D, E, F, G, H, I} e a relação de equivalência válida é apresentada no quadro 5.1.

Terminal Portuário	Sigla	Representação utilizada no AG
Terminal APPA - farelo	SV	A
Terminal APPA	SH	B
Terminal Coimbra	CB	C
Terminal Cargill	CG	D
Terminal CBL	CL	E
Terminal Coamo	CO	F
Terminal Centro Sul	CS	G
Terminal Cotriguaçú	CT	H
Terminal AGTL	PA	I

QUADRO 5.1 - Representação por terminal

A representação dos indivíduos (índividuo) possui 27 elementos e é dividida em três partes de nove elementos, as quais representam os nove TPs em cada um dos três berços de atracação. A figura 5.1 mostra a representação do indivíduo utilizado.

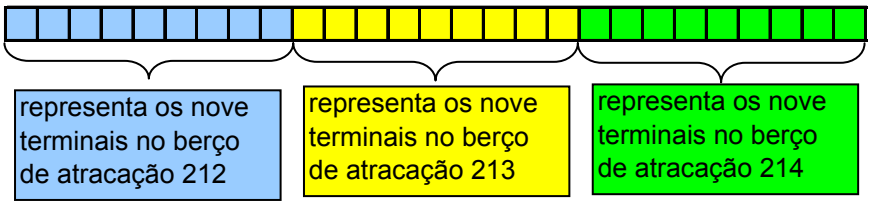

Figura 5.1 - Representação do individuo
FONTE: O autor (2007)

Como o interesse é na seqüência de embarque, não é necessário carregar informações no AG referentes ao seu tempo. O cálculo dos tempos de embarque serão feitos pela da função de avaliação (*fitness*).

5.2 POPULAÇÃO INICIAL

A população inicial é gerada aleatoriamente e o pseudo-código para a sua geração está representado na figura 5.2.

```
Ind=1
de ind= 1 até tamanho da população
        j=1, i=1
        de j=1 até 3, faça
                de i=1 até 9, faça
                        selecionar aleatoriamente um elemento da lista (A, B, C, D, E, F, G, H, I)
                        se o elemento já foi selecionado, selecione outro elemento, se não, escreva
                        na posição
                fim
        fim
 fim
```

Figura 5.2 - Pseudo-código para geração da população inicial

5.3 FUNÇÃO DE AVALIAÇÃO (*FITNESS)*

A função de avaliação obtém o valor do deslocamento total efetuado, e é composta por dois módulos distintos: i) o primeiro tem a finalidade de construir a seqüência de embarque em horas para cada um dos navios, atuando, portanto, em cada uma das três componentes do individuo, ii) o segundo módulo efetua os

deslocamentos necessários e totaliza o valor. Para o primeiro módulo a seqüência de operações para cada uma das três componentes do individuo é:

- Marque o início do período (zero) para o TP que está na primeira posição.
- Acrescente o tempo necessário de embarque do primeiro TP, para determinar o horário de termino do carregamento do TP que está na primeira posição.
- O horário de termino da operação do TP que está na primeira posição passa a ser o horário de início de carregamento do TP que está na segunda posição.
- Acrescente o tempo necessário de carregamento do TP que está na segunda posição, para determinar o horário de termino do carregamento do TP que está na segunda posição;
- Repetir o procedimento até que seja determinado o horário de término do TP que está na nona posição.

Tomando como base os dados contidos na tabela 4.1: exemplo de carregamento (p. 42), o funcionamento do primeiro módulo, considerando que a sequência obtida pelo AG seja aquela apresentada na figura 5.3.

Figura 5.3 - Indivíduo obtido como uma das solução do AG

O primeiro módulo da função de avaliação constrói a sequência de embarque determinando o momento de início e término de embarque de cada um dos TPs em cada um dos navios. Neste módulo, os conflitos, se existirem, ainda não foram identificados e, também, não foram efetuados os deslocamentos, caso sejam necessários.

Na tabela 5.2, que mostra os resultados obtidos do indivíduo representado na figura 5.3, observa-se que os TPs estão colocados na ordem do seu embarque em cada um dos navios.

TABELA 5.1 - SEQUÊNCIA OBTIDA PELO PRIMEIRO MÓDULO DA FUNÇÃO DE AVALIAÇÃO COM BASE NO INDIVÍDUO DA FIGURA 5.3

(a)			(b)			(c)		
Navio 1			**Navio 2**			**Navio 3**		
TP	**Início**	**Fim**	**TP**	**Início**	**Fim**	**TP**	**Início**	**Fim**
E	0,00	0,00	D	0,00	0,00	B	0,00	0,00
B	0,00	10,77	F	0,00	0,00	I	0,00	2,12
F	10,77	10,77	H	0,00	0,33	G	2,12	2,12
G	10,77	14,77	I	0,33	13,54	C	2,12	22,14
H	14,77	19,97	G	13,54	13,54	D	22,14	22,14
A	19,97	19,97	B	13,54	13,54	H	22,14	22,14
I	19,97	38,12	E	13,54	15,79	F	22,14	22,14
D	38,12	38,12	C	15,79	30,79	E	22,14	28,14
C	38,12	38,12	A	30,79	30,79	A	28,14	28,14

É no segundo módulo da função de avaliação que é determinado o valor do *fitness* e, inicialmente são reescritas as seqüências obtidas, porém agora colocando os TPs na ordem original {A, B, C, D, E, F, G, H, I}. A tabela 5.3 mostra a nova condição dos dados da tabela 5.2.

TABELA 5.2 - SEQUÊNCIA DO INDIVÍDUO DA FIGURA 5.3 REESCRITA EM ORDEM DOS TPS

	Navio 1		Navio 2		Navio 3	
TP	**início**	**fim**	**início**	**fim**	**início**	**fim**
A	19,97	19,97	30,79	30,79	28,14	28,14
B	0,00	10,77	13,54	13,54	0,00	0,00
C	38,12	38,12	15,79	30,79	2,12	22,14
D	38,12	38,12	0,00	0,00	22,14	22,14
E	0,00	0,00	13,54	15,79	22,14	28,14
F	10,77	10,77	0,00	0,00	22,14	22,14
G	10,77	14,77	13,54	13,54	2,12	2,12
H	14,77	19,97	0,00	0,33	22,14	22,14
I	19,97	38,12	0,33	13,54	0,00	2,12

O próximo passo do segundo módulo é descartar os TPs sem embarque em cada um dos três navios, caracterizado pelo mesmo momento de início e fim do embarque. Tal procedimento permite que seja possível efetuar o cálculo dos conflitos e efetuar os deslocamentos necessários. A tabela 5.4 mostra o resultado deste procedimento.

TABELA 5.3 - SEQUÊNCIA DO INDIVÍDUO DA FIGURA 5.3, COM DESCARTE DOS TPS SEM EMBARQUE

	Navio 1		Navio 2		Navio 3	
TP	**início**	**fim**	**início**	**fim**	**início**	**fim**
A	-	-	-	-	-	-
B	0,00	10,77	-	-	-	-
C	-	-	15,79	30,79	2,12	22,14
D	-	-	-	-	-	-
E	-	-	13,54	15,79	22,14	28,14
F	-	-	-	-	-	-
G	10,77	14,77	-	-	-	-
H	14,77	19,97	0,00	0,33	-	-
I	19,97	38,12	0,33	13,54	0,00	2,12

FONTE: O autor (2007)

Na sequência, identificam-se as intersecções não nulas e efetua-se o menor deslocamento do horário de inicio de um dos navios. Destaca-se que o cálculo do deslocamento é efetuado logo após o cálculo do conflito (intersecção), e para a escolha do embarque que será deslocado em qual navio, desloca-se o horário de início que estiver no menor deslocamento produzido entre o módulo do fim do navio$_i$ e o inicio do navio$_{i+1}$ e o módulo do fim do navio$_{i-1}$ e o inicio do navio$_i$. Atribui-se, então, o horário de início do embarque deslocado igual ao horário de termino do embarque que não foi deslocado.

$$deslocamento_{TP} = \begin{cases} menor\left\{\left|ht_{i,TP} - hi_{i+1,TP}\right|, \left|ht_{i+1,TP} - hi_{i,TP}\right|\right\} \\ ou \\ menor\left\{\left|ht_{i,TP} - hi_{i+2,TP}\right|, \left|ht_{i+2,TP} - hi_{i,TP}\right|\right\} \end{cases} \quad (3)$$

Onde, ht$_{i,TP}$ é o horário de termino de embarque do navio *i* do terminal portuário TP, e hi$_{i,TP}$ o horário de inicio de embarque do terminal portuário TP, com *i* variando de 1 à 3, e TP variando de A à I.

Esse procedimento deve ser executado entre todos os navios, efetuando a avaliação entre o navio 1 e o navio 2, navio 1 e navio 3, e navio 2 e navio 3. Após esse procedimento todos os horários de início e término de embarque dos demais TPs, do navio que sofreu a alteração e, que se encontram seqüenciados após o TP modificado também são deslocados.

A somatória dos deslocamentos efetuados é o *fitness* do indivíduo avaliado. Também é necessário após o cálculo de todos os deslocamentos, efetuar o procedimento novamente para verificar se tais deslocamentos não produziram conflitos e outros TPs. Exemplificando o processo para o caso do TP C, como os dados da tabela 5.4, tem-se:

$$\begin{aligned} deslocamento_C &= menor\left\{\left|ht_{2,C} - hi_{3,C}\right|, \left|ht_{3,C} - hi_{2,C}\right|\right\} \\ deslocamento_C &= menor\left\{\left|30,79 - 2,12\right|, \left|22,14 - 15,79\right|\right\} \\ deslocamento_C &= menor\left\{28,60; 6,35\right\} \\ deslocamento_C &= 6,35 \end{aligned} \tag{4}$$

Em função do critério de deslocamento o horário de início do navio 2, no TP C, assume $hi_{2,C}$=22,14.

Na tabela 5.4, verifica-se ainda, a existência de conflitos no TP I, entre o navio 2 e o navio 3, e efetuando o deslocamento obtém-se $hi_{2,I}$=2,12, e $deslocamento_I$=2,12. Ocorre que em função desse deslocamento alterou-se os momentos de inicio e de termino dos TP E e C do navio 2. Efetuando novamente o procedimento obtém-se $hi_{2,C}$=22,14, e $deslocamento_C$=4,56. Quando se efetua novamente o procedimento, não se observa nenhuma alteração, portanto o módulo é encerrado e o fitness obtido, que é igual a somatório dos deslocamentos é de 6,68.

Na tabela 5.5, mostra-se os horários de inicio e termino de cada um dos TPs, em cada um dos navios após a execução dos procedimentos acima descritos.

TABELA 5.4 - SEQUÊNCIA DO INDIVÍDUO DA FIGURA 5.3, COM DESLOCAMENTOS DOS TPS COM CONFLITOS

	Navio 1		Navio 2		Navio 3	
TP	**início**	**fim**	**início**	**fim**	**Início**	**fim**
A	-	-	-	-	-	-
B	0,00	10,77	-	-	-	-
C	-	-	22,14	37,14	2,12	22,14
D	-	-	-	-	-	-
E	-	-	15,33	17,58	22,14	28,14
F	-	-	-	-	-	-
G	10,77	14,77	-	-	-	-
H	14,77	19,97	0,00	0,33	-	-
I	19,97	38,12	2,12	15,33	0,00	2,12

Verifica-se na tabela 5.5 que não ocorrem mais conflitos. Para facilitar a compreensão e visualização desse fato elaborou-se o gráfico de Gantt, conforme figura 5.4. As áreas hachuradas representam os deslocamentos efetuados.

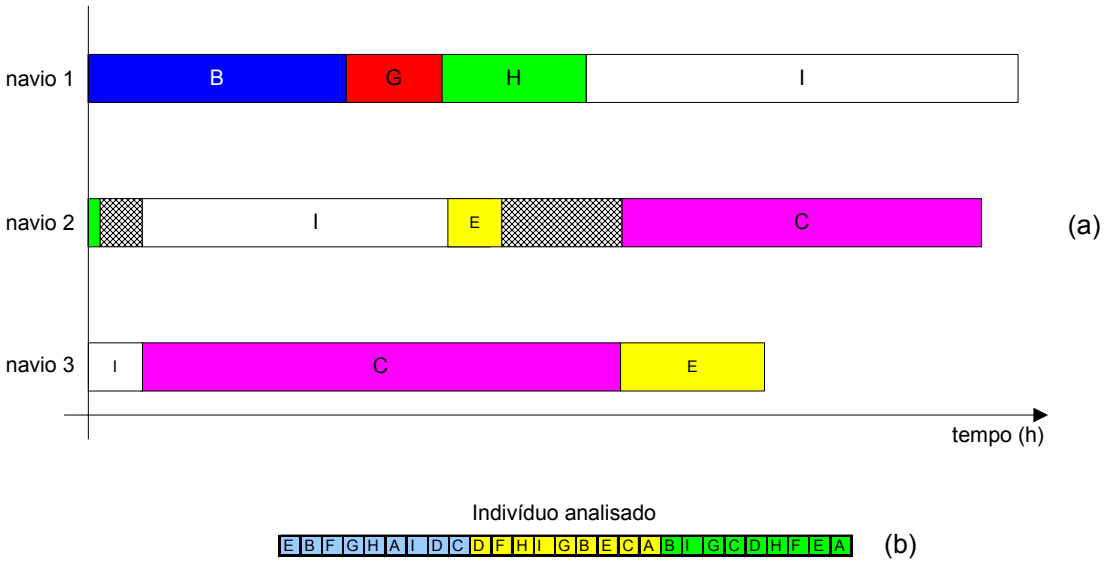

Figura 5.4 - (a) Gráfico de Gantt da seqüência obtida no indivíduo da figura 5.3, considerando os deslocamentos necessários; (b) indivíduo analisado.

Na figura 5.5, mostra-se o pseudo-código para o calcula do deslocamento total obtido em função da seqüência de carregamento obtida pelo AG. O segundo passo força que o deslocamento seja calculado novamente, somente encerrando o processo quando deslocamento igual zero, o que significa que em virtude dos deslocamentos efetuados não ocorreu um novo conflito.

cont=1, $deslocamento_0$=0
enquanto $deslocamento_{cont-1} \neq deslocamento_{cont}$, faça
    cont=*cont*+1
    TP=1
    Repetir de *TP*=1 até 9
        $deslocamento_{TP}$=0
        se $hi_{1,TP}=ht_{1,TP}$ ou $hi_{2,TP}=ht_{2,TP}$, se não faça
            Se $ht_{2,TP}<hi_{1,TP}$ faça $hi_{1,TP}=ht_{2,TP}$
            $deslocamento_{TP}=deslocamento_{TP}+ht_{2,TP}-hi_{1,TP}$
            fim se
            Se $ht_{1,TP}<hi_{2,TP}$ faça $hi_{2,TP}=ht_{1,TP}$

$deslocamento_{TP}$=$deslocamento_{TP}$+$ht_{1,TP}$-$hi_{2,TP}$
Fim se
fim se
se $hi_{1,TP}$=$ht_{1,TP}$ ou $hi_{3,TP}$=$ht_{3,TP}$, se não faça
Se $ht_{3,TP}$<$hi_{1,TP}$ faça $hi_{1,TP}$=$ht_{3,TP}$
$deslocamento_{TP}$=$deslocamento_{TP}$+$ht_{3,TP}$-$hi_{1,TP}$
Fim se
Se $ht_{1,TP}$<$hi_{2,TP}$ faça $hi_{3,TP}$=$ht_{1,TP}$
$deslocamento_{TP}$=$deslocamento_{TP}$+$ht_{1,TP}$-$hi_{3,TP}$
Fim se
fim se
se $hi_{2,TP}$=$ht_{2,TP}$ ou $hi_{3,TP}$=$ht_{3,TP}$, se não faça
Se $ht_{3,TP}$<$hi_{2,TP}$ faça $hi_{2,TP}$=$ht_{3,TP}$
$deslocamento_{TP}$=$deslocamento_{TP}$+$ht_{3,TP}$-$hi_{2,TP}$
Fim se
Se $ht_{3,TP}$<$hi_{2,TP}$ faça $hi_{3,TP}$=$ht_{2,TP}$
$deslocamento_{TP}$=$deslocamento_{TP}$+$ht_{2,TP}$-$hi_{3,TP}$
Fim se
Fim se
$deslocamento_{cont}$=$deslocamento_{cont}$+$deslocamento_{TP}$
Fim repetir
Fim enquanto

Figura 5.5 - Pseudo-código para o calculo do deslocamento

5.4 SELEÇÃO DOS INDIVÍDUOS

Para a seleção dos indivíduos é utilizado o método de Monte Carlo, descrito na seção 3.7. Porém, para o caso específico utiliza-se a função inversa, em virtude do objetivo ser minimizar os deslocamentos. Quanto menor o deslocamento produzido pela seqüência, maior será a probabilidade desse individuo de ser escolhido. O *fitness* visualizado é o deslocamento, porém para o calculo do *fitness* a ser maximizado é utilizada a função:

$$deslocamento^{*} = \frac{1}{1 + deslocamento} \tag{5}$$

É necessário que o denominador da função de avaliação seja da forma (1+deslocamento), para evitar possíveis erros quando deslocamento for igual a zero.

5.5 OPERADOR *CROSSOVER*

O operador *crossover* utilizado é aquele apresentado na seção 3.4, ou seja, um *crossover* baseado em ordem como o pseudo-código representado na figura 3.3. E para garantir uma maior diversidade nos indivíduos optou-se por gerar uma seqüência de bits distinta para cada uma das três partes do indivíduo.

Passo 1	Gere uma seqüência de bits aleatória do mesmo tamanho que os elementos
Passo 2	Copie para o filho 1 os elementos do pai 1 referentes àquelas posições onde a *string* de bits possui 1
Passo 3	Faça uma lista dos elementos do pai 1 referentes a zeros da *string* de bits
Passo 4	Permute esta lista de forma que os elementos apareçam na mesma ordem que no pai 2
Passo 5	Coloque estes elementos nos espaços do filho 1 na ordem gerado no passo anterior
Passo 6	Repita o processo para gerar o filho 2, substituindo o pai 1 pelo pai 2

Figura 5.6 - Pseudo-código do operador crossover cada uma das três partes do indivíduo.
Fonte: LINDEN (2006, p. 177)

5.6 OPERADOR DE MUTAÇÃO

Para a mutação dos indivíduos foi adotado o operador de mutação baseado em permutação de elementos explicitado na seção 3.5.1, aplicando-se independentemente para cada uma das três partes do indivíduo, o que garante uma maior diversidade. O operador de mutação é aplicado em função de tal método gerar maior diversidade na população. E da mesma forma que o operador *crossover*, para garantir uma maior diversidade o processo de mutação é aplicado independentemente em cada uma das três partes do indivíduo.

5.7 ELITISMO

Também foi considerado o elitismo, mantendo dessa forma os melhores indivíduos nas populações seguintes, isto define que a população é mista, ou seja, os melhores indivíduos podem continuar existindo até a última população.

5.8 IMPLANTAÇÃO DO ALGORITMO

O algoritmo foi implementado em linguagem Progress tendo como entrada, além dos dados de embarque nos navios os seguintes parâmetros: tamanho da população, número de gerações, percentual de mutação, percentual de sobrevivência.

O *software* foi executado num micro computar com processador AMD Turion 64 *mobile*, tecnologia MK-36 com 797 MHz e memória RAM de 1 Giga. A figura 5.7 mostra a tela inicial do *software* desenvolvido.

O campo A, exibe o *finess*, ou seja, a somatória dos deslocamentos efetuados, o campo B a seqüência de carregamento (indivíduo), o campo C, o número da geração do individuo, o campo D a posição que o individuo ocupa na roleta e o campo E (sim/não) mostra se o indivíduo será utilizado na próxima geração.

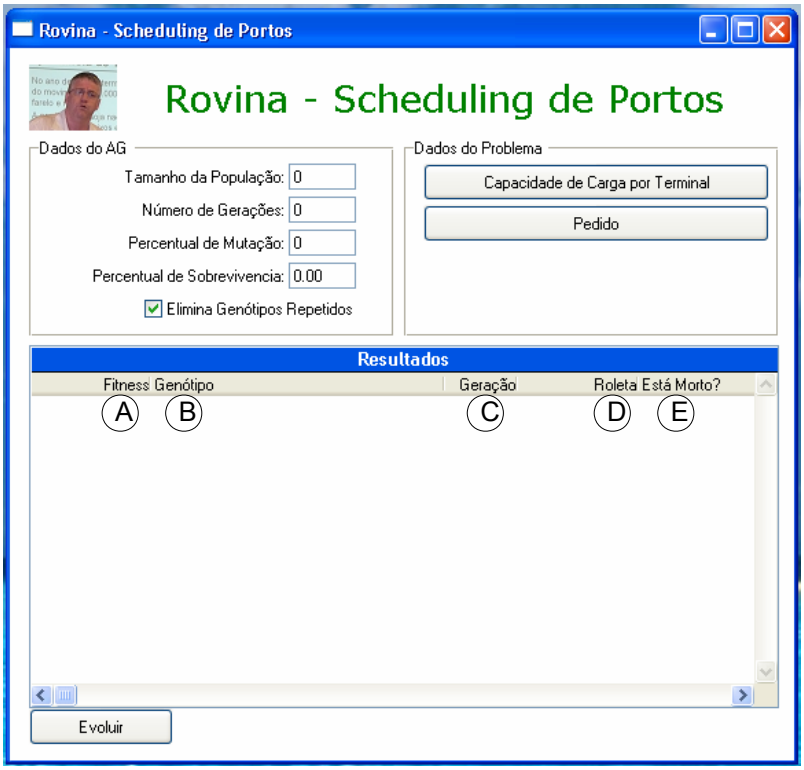

Figura 5.7 - Tela inicial

O botão *"Capacidade de Carga por Terminal"* permite cadastrar os TPs e as suas capacidades, dadas em toneladas por hora (t/h), e a figura 5.8 mostra o cadastro efetuado.

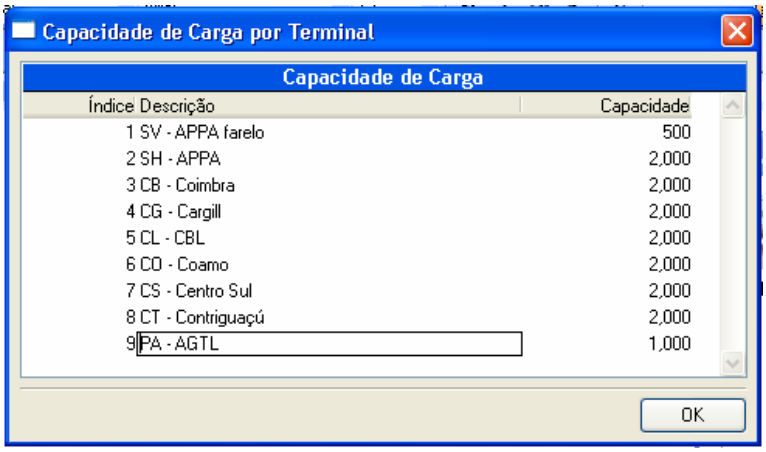

Figura 5.8 - Cadastro dos terminais e capacidades de carga

O Botão *"Pedido"* permite informar, ao sistema, as quantidades, em toneladas (t), a carregar de cada um dos TPs em cada um dos navios. O cadastro dos navios é efetuado em função do berço de atracação em que ele se encontra (Berço 212, berço 213, berço 214).

Finalmente o botão *"Evoluir"* realiza a execução do algoritmo com os parâmetros definidos. Acrescentou-se, ainda, uma condição de parada para quando o *fitness* for igual a zero, visto que essa é uma solução ótima.

No anexo A encontra-se o programa desenvolvido.

5.9 DEFINIÇÃO DOS PARÂMETROS

Para a definição dos parâmetros, tamanho da população, numero de gerações e percentual de sobrevivência executou-se 35 evoluções com diversas alternativas. Em todas as evoluções o percentual de mutação foi mantido em 10%, sendo sua definição efetuada arbitrariamente.

A tabela 5.6 mostra os valores obtidos nas evoluções, considerando os dados de carregamento mostrado na tabela 4.1, fornecendo ainda informações referentes ao tempo de execução e a primeira solução de cada uma das execuções.

TABELA 5.5 - VALORES OBTIDO NA EVOLUÇÃO

nº do teste	Tamanho da população	Número de gerações	Percentual de sobrevivência (%)	Tempo de execução (min:s)	Melhor *fitness*	Quantidade de soluções com *fitness* igual	Solução
1	50	50	10	00:46	6,3920	2	GHBFCEIDA IEBHFGDAC CIEFBHADG
2	50	50	20	00:46	6,3485	1	ABFGCHDIE BDGHIAEFC ICHEGAFBD
3	50	50	30	00:45	6,3485	3	ABFGCHDIE BDGHIAEFC ICHEGAFBD
4	50	50	40	00:46	6,3485	2	GAFHEDBIC BIGHFAEDC HICADBGFE
5	50	50	50	00:45	6,3485	2	GBCHIDAEF DBEAHIFGC IBHGDCAEF
6	50	50	60	00:46	6,3485	2	GDHEABCIF HDIEBCFAG BDICEFGHA
7	50	50	70	00:46	8,6420	1	BAEGFHIDC HDICEABGF BDCIEHFGA
8	50	100	10	01:32	6,3485	6	DHCBAFGEI GDBIHCAFE ICADGFHEB
9	50	100	20	01:31	6,3485	10	FEDGAHCBI DGIEBHFCA ADGHICBEF
10	50	100	30	01:33	6,3485	14	HFCGBIEAD EAIHFCBGD AIGHBCDEF
11	50	100	40	01:31	6,3485	2	EBHCGADFI EBDHAFIGC DIAHCFGBE
12	50	100	50	01:31	6,3485	2	CGHFEDBIA BGFIHDECA GIBAHDCEF
13	50	100	60	01:32	6,3485	10	DBHFCAEGI GDHFIABEC GIDAHBFCE
14	50	100	70	01:35	6,3485	1	DFGBAHICE HGDBAIFEC FGDAIBHCE
15	50	200	10	03:05	6,3485	5	DFACBHEIG IDFBEHACG AGIDCFHBE
16	50	200	20	03:03	6,3485	10	DCEBFHGAI IEHGCDFBA ABICFHEGD
17	50	200	30	03:04	6,3485	14	GHAFDBEIC BDAHEGFIC FDICHEGBA
18	50	200	40	03:05	6,3485	20	FAGDECBHI GEFAHICDB IACFBDGHE
19	50	200	50	03:05	6,3485	6	GEHCBFADI DBGEIHACF DIBHACGFE
20	50	200	60	03:06	6,3485	15	HABGFCIED IBHDEFGCA IBFACHEGD
21	50	200	70	03:09	6,3485	6	EGCDHFBIA HGAEBICFD ICBHAFGDE
22	100	50	10	02:11	6,3920	1	AFHCEGBID GBHIEDACF CIBHAEGFD
23	100	50	20	02:06	6,3485	3	FCEHGBAID IGHFEDBCA IHGBCDAEF
24	100	50	30	02:06	8,9315	1	HBDICEFGA DGHBICEFA IGBDFHACE
25	100	50	40	02:06	12,6815	1	HCAGEFDBI DHIFGECBA AIFDEBHGC
26	100	50	50	02:06	6,3485	2	HACBDFIEG DIGEFHABC BIHDFCGAE
27	100	50	60	02:06	6,3485	1	CDEFBHGIA IDABEHFGC DICHEAFBG
28	100	50	70	02:10	6,3485	2	HAGECDFBI IGEHAFCDB AGFIHCEDB
29	100	100	10	04:21	6,3485	8	CABFGEHDI HBDEIACGF IGHCFEABD
30	100	100	20	04:21	6,3485	4	DHGBCIFEA EAIGHCFBD AHBIGDFCE
31	100	100	30	04:19	6,3485	2	DHFGBAEIC GDEIHAFBC IFBCDGEHA
32	100	100	40	04:30	6,3485	2	DAEGCFBHI BHEIAGDCF DBIGACHEF
33	100	100	50	04:25	6,3485	6	CGHABIEDF EIFBDAGHC HBICEAGFD
34	100	100	60	04:24	6,3485	1	CHEAGBDIF AIHDBECGF IHBAFDCEG
35	100	100	70	04:27	6,3485	2	CAFGBHIED HIGBADFEC DIBAHCGFE

FONTE: O autor (2007)

Em virtude de todos os TPs serem considerados em todos os navios mesmo que não possuam embarque definido, muitas dos indivíduos fornecidos na última geração acabam por representar a mesma solução, assim um dos critérios de escolha dos parâmetros passa a ser a quantidade de soluções iguais obtidas dentro da última geração.

A figura 5.9 mostra os resultados obtidos. A série 50x50 refere-se aos testes 1 a 7, com tamanho da população de igual a 50 e com 50 gerações. A série 50x100 refere-se aos testes 8 a 14, com tamanho da população igual a 50 e 100 gerações. A série 50x200 refere-se aos testes 14 a 21, com tamanho da população igual a 50 e 200 gerações.

A série 100x50 refere-se aos testes 22 a 28, com tamanho da população igual a 100 e 50 gerações. A série 100x100 refere-se aos testes número 29 a 35 com tamanho da população igual a 100 e 100 gerações.

O eixo das abcissas refere-se aos valores utilizados para o percentual de sobrevivência e o eixo das ordenadas refere-se as quantidades de soluções iguais obtidas na execução do algoritmo.

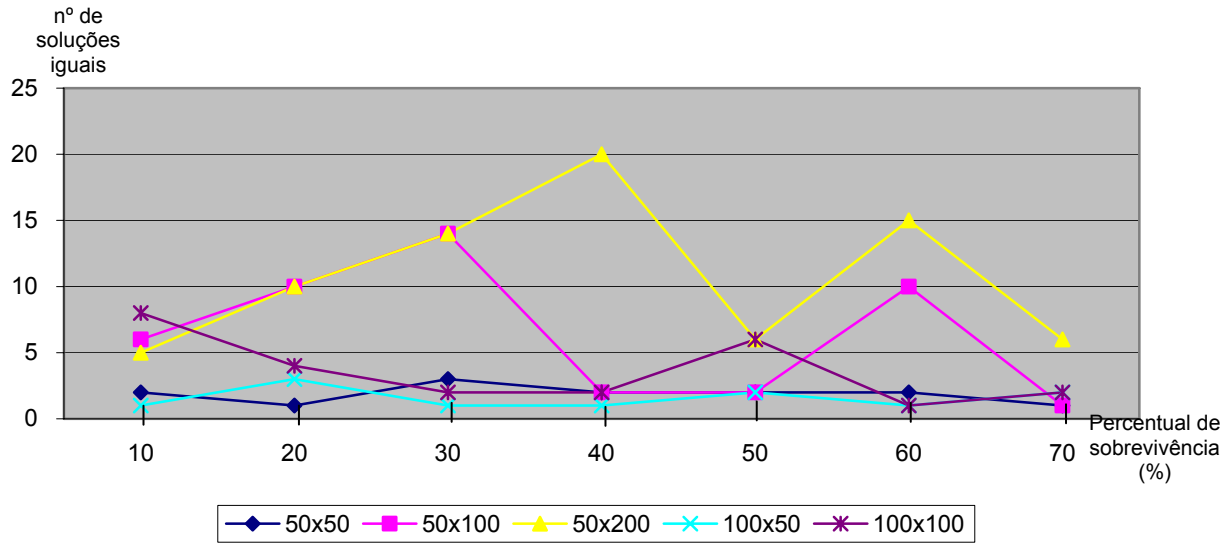

Figura 5.9 - Gráfico Percentual de mutação x quantidade de soluções iguais

Da análise da figura 5.9 definem-se os seguintes parâmetros para o caso em estudo: a) tamanho da população 50, b) quantidade de gerações 200, c) percentual de mutação 40.

Para garantir que tais parâmetros são os ideais, efetuou-se mais 10 testes e os resultados obtidos são mostrados na tabela 5.7.

TABELA 5.6 - AVALIAÇÃO DOS PARÂMETROS DEFINIDOS

nº do teste	Tempo de execução	Melhor *fitness*	Quantidade de soluções com *fitness* igual	Solução
36	00:03:04	6,3485	16	HDGBEAFCI AIHEFBCDG IACEGDFBH
37	00:03:06	6,3485	15	HDABEFCGI DAEBIFHGC AHIBCDFEG
38	00:03:04	6,3485	15	BCGAFDEHI BFAGDEHIC IHDCFBEGA
39	00:03:08	6,3485	10	EBFCHDGAI GHFBDAIEC IFADCBHGE
40	00:03:04	6,3485	9	HBGEDCFIA IEGHAFDCB ICFHABGED
41	00:03:04	6,3485	15	HBDECFIAG DBEIGFHAC BGHAIDFCE
42	00:03:04	6,3485	16	GEHBICAFD GBDHFEAIC FAGICDEHB
43	00:03:03	6,3485	5	EDFCGHBAI BIEHDFACG HBDFIGACE
44	00:03:06	6,3485	15	CAGDHEBIF DBIEAGHFC DIAGBHCFE
45	00:03:04	6,3485	15	CGDAHBFEI GHAIBECFD IDBFCGAHE

Verifica-se que a escolha dos parâmetros produziu em todos os casos gerados o mesmo *fitness* (deslocamento) e, em mais de uma das soluções. Portanto a sua escolha, para o presente caso atende as necessidades.

A única referência localizada em relação ao percentual de mutação define que para cromossomos binários a taxa de mutação é de 1/L, com L é igual ao número de variáveis binárias. No presente caso o cromossomo é baseado em ordem, o numero de posições é de 27, porém dividido em três partes iguais de 9 genes. Assim, por esse critério, se obteria uma taxa de mutação de 11,11%. Optou-se por manter a taxa de mutação de 10% visto os resultados satisfatórios já obtidos com tal valor.

5.10 VALIDAÇÃO DO ALGORITMO

Para algoritmos probabilísticos, como os AGs, é difícil algum tipo de prova formal da convergência para resultados ótimos após certo número de interações, pois seu comportamento não é previsível. A maioria das provas e teoremas se baseia no comportamento médio ou esperado ao longo do tempo, (LINDEN 2006, p. 93). Ainda, segundo o referido autor, “Algoritmos genéticos são um pesadelo em termos de análise, dado sua natureza (qual operador vai operar e como) é probabilística por natureza.”

5.10.1 Comportamento Esperado

O comportamento esperado é a obtenção de uma seqüência de embarque que produza conflito zero e, se tal fato não for possível, então, que se obtenha o menor tempo de deslocamentos. Para a analise de tal situação, utilizou-se a primeira solução do teste 42 (anexo B), que foi gerado em função dos dados contidos na tabela 4.1.

O resultado obtido é expresso pelo indivíduo de *fittness* (deslocamento) de 6,3485 h, e o indivíduo gerado é {GEHBICAFD GBDHFEAIC FAGICDEHB}. A tabela 5.8 mostra o *scheduling* gerado, considerando os deslocamentos.

TABELA 5.7 - *SCHEDULING* GERADO ATRAVÉS DO AG

	Navio 1		Navio 2		Navio 3	
TP	**início**	**fim**	**início**	**fim**	**início**	**fim**
A	-	-	-	-	-	-
B	9,2005	19,9730	-	-	-	-
C	-	-	22,1375	37,1375	2,1210	22,1375
D	-	-	-	-	-	-
E	-	-	0,3330	2,5830	22,1375	28,1375
F	-	-	-	-	-	-
G	0,0000	4,0000	-	-	-	-
H	4,0000	9,2005	0,0000	0,3330	-	-
I	19,9730	38,1220	2,5830	15,7890	0,0000	2,1210

A figura 5.10, mostra o gráfico de Gantt do *scheduling* da tabela 5.8.

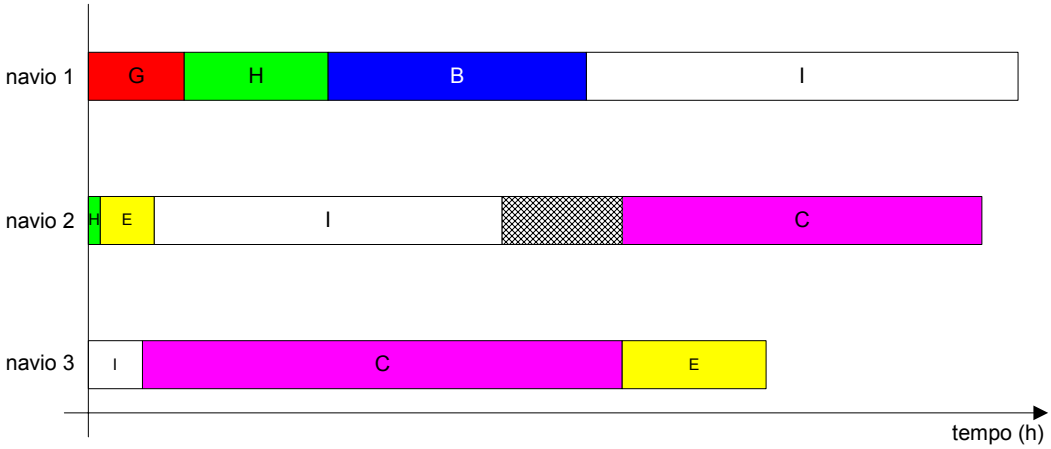

Figura 5.10 - Gráfico de Gantt da solução produzida pelo AG

Da simples observação da figura 5.10, não se pode afirmar que esta é a solução com menor deslocamento, porém em função do já exposto supõe-se que seja. Para verificar se tal fato é verdadeiro, efetuou-se a eliminação do conflito.

O causador do conflito é o embarque do TP C, no navio 3. Reduzindo-se a quantidade de embarque do TP C de 40.033t para 27.336t, que significa a redução de 6,3485h, espera-se obter uma seqüência sem a existência de nenhum conflito. A tabela 5.9 mostra os dados utilizados para a evolução do AG.

Executando o algoritmo obtém-se o *scheduling* da tabela 5.10, ocorrendo o comportamento esperado, o algoritmo desenvolvido obteve como solução a seqüência de embarque {HBEDGFCAI ADIEBGHCF CAGDHBIEF}, com *fitness* (deslocamento) igual a zero. A tabela 5.10 mostra o *scheduling* gerado.

TABELA 5.8 - DADOS ALTERADOS COM OBJETIVO DE PRODUZIR UM *SCHEDULING* COM *FITNESS* IGUAL A ZERO

	navio 1	**navio 2**	**navio 3**
Terminal portuário	**berço 212**	**berço 213**	**berço 214**
A			
B	21.545		
C		30.000	27.336
D			
E		4.500	12.000
F			
G	8.000		
H	10.401	666	
I	18.149	13.206	2.121
Total	**58.095**	**48.372**	**31.457**

TABELA 5.9 - SCHEDULING GERADO SEM DESLOCAMENTOS NECESSÁRIOS

	Navio 1		Navio 2		Navio 3	
TP	**início**	**fim**	**início**	**fim**	**Início**	**fim**
A	-	-	-	-	-	-
B	5,2005	15,9730	-	-	-	-
C	-	-	15,7890	30,7890	0,0000	13,6830
D	-	-	-	-	-	-
E	-	-	13,2060	15,4560	15,8040	21,8040
F	-	-	-	-	-	-
G	15,9730	19,9730	-	-	-	-
H	0,0000	5,2005	15,4560	15,7890	-	-
I	19,9730	38,1220	0,0000	13,2060	13,6830	15,8040

Para facilitar a visualização, a figura 5.11 exibe a sequência obtida na forma do gráfico de Gantt.

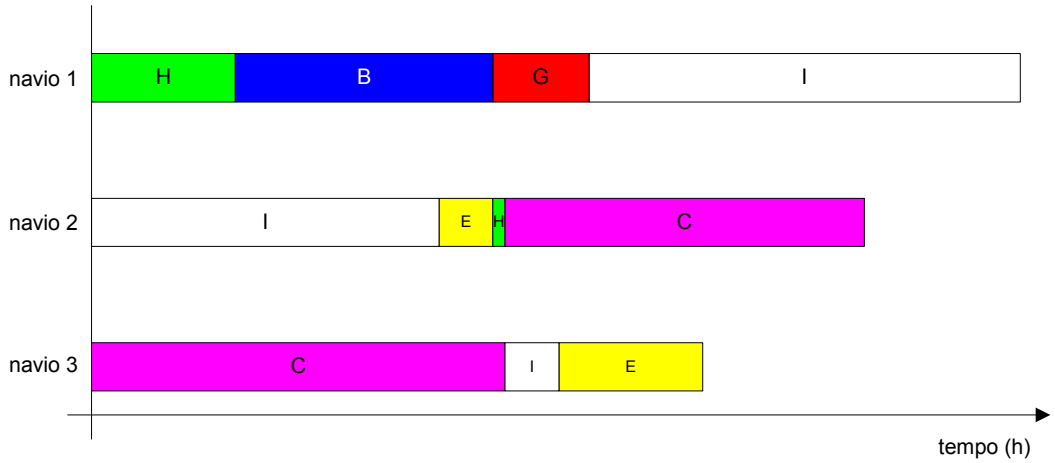

Figura 5.11 - Gráfico de Gantt da programação sem ocorrência de deslocamentos

Além disso, para assegurar que o algoritmo proposto produz soluções viáveis, elaborou-se um embarque de três navios, utilizando todos os nove TPs, com carregamento de tempos iguais, com isso espera-se que a solução obtida pelo AG seja um deslocamento igual a zero. A tabela 5.11 mostra os valores utilizados nestas condições.

TABELA 5.10 - EXEMPLO DE EMBARQUE COM TEMPOS IGUAIS EM TODOS OS TPS

	navio 1	**navio 2**	**navio 3**
Terminal portuário	**berço 212**	**berço 213**	**berço 214**
A	1.250	1.250	1.250
B	5.000	5.000	5.000
C	5.000	5.000	5.000
D	5.000	5.000	5.000
E	5.000	5.000	5.000
F	5.000	5.000	5.000
G	5.000	5.000	5.000
H	5.000	5.000	5.000
I	2.500	2.500	2.500
Total	**38.750**	**38.750**	**38.750**

No anexo C, são mostrados os resultados obtidos no teste 47. O AG obteve como resposta a sequência {DHGFEBCIA AEBIHFGDC FIABDCHGE}. A tabela 5.12 mostra o *sheduling* gerado, onde se verifica que o esperado aconteceu, ou seja, retornou um *fitness* (deslocamento) igual a zero, obtido na décima oitava geração.

TABELA 5.11 - *SCHEDULING* GERADO COM OS VALORES DA TABELA 5.12

	Navio 1		Navio 2		Navio 3	
TP	**início**	**fim**	**início**	**fim**	**início**	**fim**
A	20,0000	22,5000	0,0000	2,5000	5,0000	7,5000
B	12,5000	15,0000	5,0000	7,5000	7,5000	10,0000
C	15,0000	17,5000	20,0000	22,5000	12,5000	15,0000
D	0,0000	2,5000	17,5000	20,0000	10,0000	12,5000
E	10,0000	12,5000	2,5000	5,0000	20,0000	22,5000
F	7,5000	10,0000	12,5000	15,0000	0,0000	2,5000
G	5,0000	7,5000	15,0000	17,5000	17,5000	20,0000
H	2,5000	5,0000	10,0000	12,5000	15,0000	17,5000
I	17,5000	20,0000	7,5000	10,0000	2,5000	5,0000

A figura 5.12 mostra o gráfico de Gantt da seqüência obtida na tabela 5.11.

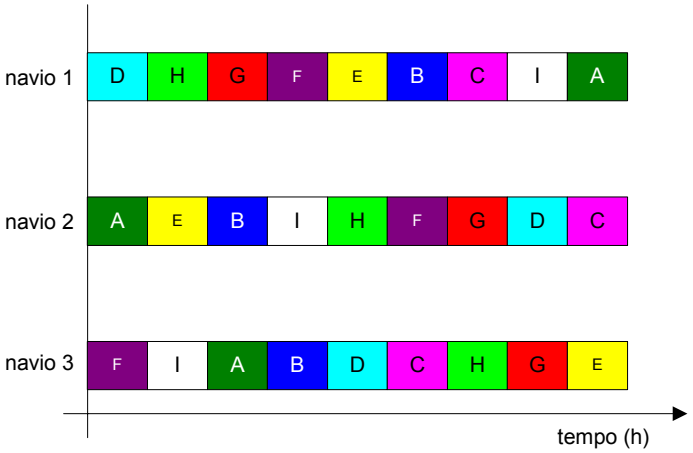

Figura 5.12 - Gráfico de Gantt do exemplo de carga com tempos iguais dos nove TPs nos três navios

5.10.2 Formação de Esquemas

Outra forma de validar o algoritmo é verificar se há a formação dos esquemas. Utilizou-se o teste 42 para a análise e verificação da existência de esquemas. Entre os teste gerados, o teste 42, com a população definida com 50 indivíduos, taxa de mutação de 10%, uma sobrevivência de 40%, o AG apresentou 16 indivíduos com o mesmo *fitenss* de 6,3485h.

Para o AG, todos os indivíduos são diferentes, porém em virtude do embarque requerido ser somente dos TPs {B, G, H, I} para o navio 1 (berço 212), {C,E, H, i} para o navio 2 (berço 213) e {C, E, I} para o navio 3 (berço 214), é a posição de desses TPs que interessam. Nos indivíduos que apresentam o mesmo

fitness como solução do AG no teste 42, eliminou-se os TPs que não efetuam embarque, a tabela 5.13 mostra essa situação.

TABELA 5.12 - INDIVÍDUOS DO TESTE 42, COM MESMO *FITNESS*, EXCLUÍDO OS TPS SEM EMBARQUE

indivíduo nº	Navio 1				Navio 2				Navio 3		
1	H	B	I	G	I	C	E	H	I	C	E
2	H	B	I	G	I	E	H	C	I	C	E
3	H	B	G	I	H	I	C	E	I	C	E
4	H	B	I	G	I	E	H	C	I	C	E
5	H	G	B	I	I	E	H	C	I	C	E
6	G	H	B	I	H	E	I	C	I	C	E
7	G	H	B	I	H	E	I	C	I	C	E
8	G	H	B	I	E	H	I	C	C	E	I
9	G	H	B	I	I	E	H	C	I	C	E
10	G	H	B	I	H	E	I	C	I	C	E
11	G	H	B	I	I	H	E	C	I	C	E
12	G	H	B	I	H	E	I	C	I	C	E
13	G	H	B	I	I	C	H	E	I	C	E
14	G	H	B	I	E	H	I	C	I	C	E
15	G	H	B	I	I	E	H	C	I	C	E
16	G	B	H	I	I	H	C	E	I	C	E

Na tabela 5.12, os indivíduos número 6, 7,10 e 12, linhas em destaque, representam exatamente a mesma seqüência de embarque, mesmo que suas posições sejam diferentes no indivíduo com os noves TPs em cada navio.

A formação dos esquemas acaba por ocorrer somente nos TPs com embarque definido e, da observação da tabela 5.13, verifica-se que um dos esquemas produzidos pelo AG pode ser conforme a figura 5.13.

Figura 5.13 - Esquema produzido pelo AG

É relevante salientar que os embarques efetuados no navio 3, apresentam quase que na sua totalidade (93,75%), a mesma sequência de embarque. Justifica-se tal fato, pois este navio é o causador dos conflitos, fato este comprovado quando reduzindo a quantidade de embarque, tabela 5.9, produziu-se uma sequência sem conflitos conforme tabela 5.10 e figura 5.11.

Nos navios 1 e 2, as sequências são dependentes, porém na maioria dos casos as seqüências são conforme o esquema da figura 5.13.

5.11 TESTES EFETUADOS

Com os dados de diversas necessidades de embarque (anexo D) fornecidos pela APPA, em diversas datas e, utilizando o algoritmo proposto com os parâmetros já definidos, efetuou-se o sequenciamento dos TPs.

As datas foram escolhidas pelos operadores do sistema aleatoriamente, porém objetivando abranger a maior diversidade possível de situações. Também foram analisadas situações onde um dos terminais encontrava-se em manutenção, como a sequência de embarque de 24/10/2007 e 26/10/2007.

Em alguns dos casos o inicio do embarque nos navios ocorre em momentos diferentes, porém para efeitos de análise foram considerados como iniciando no mesmo instante.

A tabela 5.13 que exibe os resultados obtidos pode-se verificar que na maioria dos embarques (78%) o algoritmo proposto obteve uma sequência de embarque sem a ocorrência de conflitos.

A necessidade de embarque do dia 22/03/2007 é utilizada anteriormente para a análise do problema (capítulo 4), sendo também utilizada para os testes e validação do algoritmo no presente capitulo, porém o resultado demonstrado na tabela 5.14 foi obtido através de nova execução do algoritmo. Desse fato e da observação dos testes anteriores verifica-se que a solução obtida pelo algoritmo, mesmo obtendo seqüências de embarque distintas acaba sempre por produzir o mesmo deslocamento (*fitness*).

TABELA 5.13 - RESULTADOS OBTIDOS COM CASOS REAIS

Data de inicio de embarque	Tempo de execução (min:ss)	nº da geração	*Fitness* (deslocamento) em horas	Melhor solução apresentada
22/3/2007	03:04	148	6,3845	GHFCDAEBI HGIAFBCDE DAHIFBGEC
2/4/2007	00:02	2	0,0000	GDFAECBHI FCABEIGHD FDAHBECGI
4/4/2007	00:04	6	0,0000	FACIDHBEG IHEABGCDF DBFAGIHEC
7/4/2007	03:03	187	1,9000	BEAGIFCHD ABIHCEFDG DBFHECGIA
29/4/2007	03:10	109	1,3500	FCHAIEGDB CGHEDBFIA GBDEFCAIH
1/5/2007	00:01	0	0,0000	GBCFIADHE BAIEHFCDG DHCIFGEAB

continua

Continuação da Tabela 5.13

Data de inicio de embarque	Tempo de execução (min:ss)	nº da geração	*Fitness* (deslocamento) em horas	Melhor solução apresentada
22/5/2007	02:49	2	0,5855	IGFBEADHC DHCEGBAFI GBEFHACID
25/5/2007	00:01	1	0,0000	GDCBFHEAI HBIFGECAD CAFGDHBIE
28/5/2007	00:05	5	0,0000	IEDFCBAHG EHACDIBFG FEGHDCIAB
30/5/2007	01:11	80	0,0000	FCDBHGEAI EBGFCDIHA GEDACHIBF
17/8/2007	00:01	0	0,0000	DGBACIEFH CDFEBGHAI ADBGCIEFH
26/8/2007	00:01	1	0,0000	GDCFEHIAB ADEGHICFB DABHGCIFE
28/8/2007	00:01	1	0,0000	IGCHFDBEA IDEHCABGF GAHECFIDB
16/9/2007	00:05	6	0,0000	ACGDEFHBI IDBGHCAFE FBICEDAHG
27/9/2007	00:01	0	0,0000	ABCGFEHID IBGADFECH FBEHAIGDC
1/10/2007	00:44	48	0,0000	BAGIHDEFC CFHAEIGBD HEGBCDIAF
24/10/2007	00:01	0	0,0000	AEFDCBIGH BHFCEIDAG FCEDIGBAH
26/10/2007	00:01	0	0,0000	BIDAECGHF ADEHGCFIB EAGIBDHFC

Para o problema de sequenciamento de embarque de graneis sólidos a situação onde não ocorra nenhum conflitos é a ótima, dessa forma foi implantado no *software* o critério de parada quando o conflito é igual a zero. Como se observa na Tabela 5.13 em algumas das situações um dos indivíduos gerados na população inicial não apresenta nenhum conflito (significado do *fitness* igual a zero), sendo dessa forma o programa encerrado.

Tais fatos ocorrem em função da quantidade de TPs solicitados para embarque em cada um dos navios, sendo nesses casos verificado poucas, ou quase nenhuma possibilidade da ocorrência de conflitos.

É importante destacar que a quantidade de TPs no carregamento de cada navio não é um número fixo, podendo em cada caso variar entre 1 e 9, sendo 9 a quantidade de TPs existentes.

6 CONCLUSÃO E SUGESTÃO PARA TRABALHOS FUTUROS

Da análise do atual modelo de sequenciamento de embarques de grãos no Porto de Paranguá verificou-se que nenhuma técnica formal é utilizada. Atualmente o *scheduling* é efetuado considerando-se o conhecimento e a experiência dos programadores.

A administração de produção trata dos problemas de sequenciamento e o embarque de grãos de diversos terminais em diversos navios é um problema que considerou-se como uma programação finita e para frente. Da análise do problema dentro dessa ótica surgiu a construção do algoritmo proposto utilizando para tanto os AGs.

A contribuição de maior relevância do presente trabalho reside no desenvolvimento da função objetivo do AG que consegue identificar a ocorrência de embarque de um mesmo TP e em dois ou mais navios que denominou-se de conflito e deslocar o momento de inicio do embarque em um dos navios para eliminar o referido conflito é ponto chave da solução do problema.

O AG conseguiu eliminar os conflitos na maioria dos casos e quando não foi possível tal fato determinou os menores deslocamentos. Assim, o algoritmo proposto atende aos objetivos e obtém resultados satisfatórios.

6.1 SUGESTÕES PARA TRABALHOS FUTUROS

Nos trabalhos futuros podem ser avaliados dois grupos distintos. O primeiro grupo relaciona-se com as melhorias do algoritmo proposto e no segundo, a comparação do algoritmo proposto em relação a outras técnicas de seguenciamento.

6.1.1 Melhorias do algoritmo proposto

Após a conclusão dos trabalhos identificou-se algumas melhorias no algoritmo proposto que poderão melhorar o seu desempenho e aumentar o campo de abrangência dos problemas de sequenciamento em TPs.

- A definição de um TP como o primeiro da seqüência de embarque, em cada um dos berços, permitindo que possa ser gerada uma nova sequência a partir de um carregamento em execução.
- A utilização da data e horário para o início dos embarques em cada um dos navios.
- A escolha dos próximos navios a serem embarcados, tomando como base a fila de espera, de tal forma que não ocorram conflitos, para tanto utilizando um AG em duas camadas conforme PETROVICK (2005, *apud* LINDEN, 2006), onde a primeira camada determina que navio deva embarcar e em qual terminal e a segunda camada determina a sequência de embarque de cada navio.
- O efeito das marés pode alterar o momento de liberação e atracação dos navios nos berços influindo no tempo global da operação. Analisar o problema considerando uma programação para trás (*forward*) pode eliminar tais possibilidades.

6.1.2 Comparações

Comparar os resultados obtidos através do algoritmo proposto com outros métodos de sequenciamento tais como a Teoria das Restrições, Busca Tabu e outros, além disso comparar aos resultados obtidos através dos *softwares* comerciais como Preactor da CIMulation Center (Inglaterra), Scheduler da Manigistcs (Canadá) entre outros.

REFERENCIAS BIBLIOGRÁFICAS

Administração dos Portos de Paranaguá e Antonina, disponível em: http://www.portosdoparana.pr.gov.br/, último acesso em 24/07/2007

Andrade, Eduardo Leopoldino de (2004). **Introdução à Pesquisa Operacional – Métodos e Modelos para a Análise de Decisões**, LTC Editora, Rio de Janeiro, RJ, 3ª edição.

Bean, J. (1994). **Genetic Algorithms and Random Keys for Sequencing and Optimization**, ORSA J. Computing, 6(2), 154-160.

Bierwirth, C. (1995). **A Generalized Permutation Approach to Job-Shop Scheduling with Genetic Algorithms**, OR Spektrum, 17(2-3), 87-92.

Bierwirth, C., Mattfeld, D. C. e Kopfer, H. (1996). **On Permutation Representations for Scheduling Problems**, in Voigt, H. M. et al. (eds) PPSN'IV Parallel Problem Solving from Nature, Springer-Verlag, Berlin, Heidelberg, Germany, pp. 310-318.

Buzzo, W.R., Mocellin, J.V. (2000). **Programação da Produção em Sistemas Flow Shop Utilizando Um método Heurítico Híbrido Algoritmos Genéticos- Simulated Annealing**, Gestão & Produção, v.7, n.3, p.364-377,

Carvalho, M.A.M., Santos, A.G e Mateus, G.R. (2005). A**lgoritmo Genético aplicado ao Problema Set Covering multiobjetivo: uma etapa do Problema de Escalonamento de Tripulações**, XXV Congresso da Sociedade Brasileira de computação, São Leopoldo, RS, 2005, p. 1078-1081.

Catarina, A.S. (2006). **Um Algoritmo Genético com Representação Explícita de Relacionamentos Espaciais para Modelagem Sócio-Ambiental**, Proposta de Tese de Doutorado em Computação Aplicada, Instituto Nacional de Pesquisas Espaciais – IMPE, São José dos Campos.

Coli, C.C.M. (2004). **Análise da Demanda por Transporte Ferroviário: O Caso do Transporte de Grãos e Farelo de Soja na Ferronorte**, Dissertação, Administração, Universidade Federal do Rio de Janeiro – UFRJ.

Consentino, A. e Erdmann, R. H. (1999). **O Planejamento e Controle da Produção na Pequena e Micro Empresa do Setor de Confecções com a Utilização do Programa PCP-PME**. Caderno de Administração, Santa Catarina, Ano 1, nº 1, fev/1999.

Corrêa, H.L., Gianesi, I.G.N. e Caon M. (2001). **Planejamento, Programação e Controle da Produção**, Editora Atlas, São Paulo, 4ª edição.

Corrêa, H.L. e Corrêa, C.A. (2007). **Administração de Produção e Operações**, Editora Atlas, 2ª edição, São Paulo.

Daru, G.H. (2005). **Uma Heurística para o Sequenciamento da Produção Baseada na Teoria das Restrições**, Dissertação, Métodos Numéricos de Engenharia, Universidade Federal do Paraná – UFPR.

Davis, L. (1985). **Job-Shop Scheduling with Genetic Algorithm**, in Grefenstette J. J. (ed) Proceedings of the 1st International Conference on Genetic Algorithms and their Applications, Pittsburgh, PA, USA, Lawrence Erlbaum, pp. 136-140.

Davis, L (1989). **Adapting operator probabilities in Genetic Algorithms**. In: NTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 3., 1989, Fairfax. **Proceedings…** San Mateo : Morgan Kaufmann, 1989.

Davis, L (1996). **Handbook of Genetic Algorithms**. Reissue edition. Stamford: International Thomson Publishing, 1996.

Dell'Amico, M. e Trubian, M. (1993). **Applying Tabu Search to the Job-Shop Scheduling Problem, Annals of Operations Research**, vol 41, p. 231-252.

Della Croce, F., Menga, G., Tadei, R., Cavalotto, M. e Petri, L. (1993). **Cellular Control of Manufacturing Systems**, European Journal of Operational Research, vol 69, 498-509.

Della Croce, F., Tadei, R. e Rolando, R. (1994). **Solving a Real World Project Scheduling Problem with a Genetic Approach**, Belgian Journal of Operations Research, Statistics and Computer Science, 33(1-2), p. 65-78.

Della Croce, F., Tadei, R. e Volta, G. (1995). **A Genetic Algorithm for the Job Shop Problem**, Computers and Operations Research, Jan, 22(1), p. 15-24.

Dorndorf, U. e Pesch, E. (1995). **Evolution Based Learning in a Job-Shop Scheduling Environment**, Computers and Operations Research, Jan, 22(1), p. 25-40.

Falcone, M.A.G. (2004). **Estudos Comparativo Entre Algoritimos Genéticos e Evolução Diferencial Para Otimização de um Modelo de Cadeia de Suprimento Simplificada**, Dissertação, Engenharia de Produção e Sistemas, Pontifícia Universidade Católica – PUC, Curitiba, PR.

Falkenauer, E. e Bouffouix, S. (1991). **A Genetic Algorithm for the Job-Shop**, Proceedings of the IEEE International Conference on Robotics and Automation, Sacremento, California, USA, p. 824-829.

Fang, H. L. (1994). A**lgorithms in Timetabling and Scheduling**. Phd Thesis, at The University of Edinburgh.

Gadioli, J. A. S. (2003). **Programação com Capacidade Finita e APS no Setor de Serviços**. Dissertação de mestrado. Engenharia de Produção. Universidade Federal de Santa Catarina.

Giffler, B. e Thompson, G. L. (1960). **Algorithms for Solving Production Scheduling Problems**, OperationsResearch, 8(4), p. 487-503.

Glover, F. e Laguna, M. (1997). **Tabu Search**, Kluwer Academic Publishers, Norwell, MA.

Goebel, D. (1996). **Logística – Otimização do Transporte e Estoques na Empresa**. Estudos em Comércio Exterior Vol. I nº 1 – jul/dez.

Golbarg, M.C. e Luna, H.P. (2005). **Otimização Combinatória e Programação Linear**, Editora Campus, Rio de Janeiro.

Gonçalves, J.F., Mendes, J.M., e Resende, M.C., (2002). **A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem** - AT&T Labs Research Technical Report TD-5EAL6J.

Grefenstette, J. J. (1987). I**ncorporating Problem Specific Knowledge into Genetic Algorithms**, in Davis, L. (ed) Genetic Algorithms and Simulated Annealing, Pitman, p. 42-60.

Holland, J. H (1975). **Adaptation in natural and artificial systems**. Ann Arbor: University of Michigan Press.

Holsapple, C., Jaco, C., Pakath, R. e Zaveri, T. (1993). **A genetic-based hybrid scheduler for generating static schedules in flexible manufacturing contexts**. IEEE Transactions on Systems, Man. And Cybernetics, vol. 23, p.953-971.

Holland, J.H, et al. (1986). **Induction Processes of Inference, Logic and Discovery**. Cambridge, MA, MIT Press.

Hax, A. e Candea, D (1984). **Production and Inventory Management**. Prentice Hal,

Heizer, J. e Render, B. (2001). **Administração de Operações: Bens e Serviços**. 5ª edição, Rio de Janeiro, Editora LTC.

Heitkoetter, Joerg e Beasley David (1995). ***The Hitch-Hiker's Guide to Evolutionary Computation*. FAQ in comp.ai.genetic**, disponível em: http://www.faqs.org/faqs/ai-faq/genetic/part1/preamble.html, ultimo acesso em 10/02/2008.

Kobayashi, S., Ono, I. e Yamamura, M. (1995). **An Efficient Genetic Algorithm for Job Shop Scheduling Problems**, Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, San Francisco, CA, p. 506-511.

Kubota, A. (1995). **Study on Optimal Scheduling for Manufactoring System by Genetic Algorithms**. Master's thesis, Ashikaga Institute of Technology, Ashikaga, Japan.

Landmann, R. (2005). **Um Modelo Heurístico para a Programação da Produção em Fundições com a Utilização da Lógica Fuzzy**, Tese de doutorado, Engenharia de Produção, Universidade Federal de Santa Catarina.

Linden, R. (2006). **Algoritmos Genéticos: Uma Importante Ferramenta**, Brasfort livros e multimídia, Rio de Janeiro.

Lucas, D.C. (2002). **Algoritmos Genéticos: Uma Introdução**, Apostila da Disciplina Ferramentas da Inteligência Artificial, Universidade Federal do Rio Grande do Sul – UFRGS.

Martins, P.G. e Laugeni F.P. (2003). **Administração da Produção**, Editora Saraiva, São Paulo.

Mattfeld, D. C., Kopfer, H. e Bierwirth, C. (1994). **Control of Parallel Population Dynamics by Social-Like Behaviour of GA-Individuals**, PPSN'3 Proceedings of the Third International Conference on Parallel Problem Solving from Nature, Springer-Verlag, p. 15-25.

Mattfeld, D. C. (1996). **Evolutionary Search and the Job Shop: Investigations on Genetic Algorithms for Production Scheduling**, Physica-Verlag, Heidelberg, Germany.

Mattfeld, D. C., Bierwirth, C. e Kopfer, H. (1998). **A Search Space Analysis of the Job Shop Scheduling Problem**, to appear in Annals of Operations Research.

Moscato, P. (1989). **On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms**, C3P Report 826, Caltech Concurrent Computation Program, Caltech, California, USA.

Nakano, R. e Yamada, T. (1991). **Conventional Genetic Algorithm for Job-Shop Problems**, in Kenneth, M. K. and Booker, L. B. (eds) Proceedings of the 4th International Conference on Genetic Algorithms and their Applications, San Diego, USA, p. 474-479.

Norman, B. A. and Bena, J. (1997). **Random Keys Genetic Algorithm for Job Shop Scheduling**, Engineering Design and Automation, vol 3, p.145-156.

Norman, B. e Bena, J. (1995a). **Random Keys Genetic Algorithm for Job-shop Scheduling: Unabridged Version**. Techinical Report, University of Michigan, Ann Arbor.

Norman, B. e Bena, J. (1995b). **Random Keys Genetic Algorithm for Scheduling**. Techinical Report, University of Michigan, Ann Arbor.

Programa de Aceleração do Crescimento, disponível em: http://www.brasil.gov.br/pac/, acesso em 21/10/2007.

Pedroso, M. e Corrêa H. L (1996). **Sistemas de Programação da Produção com Capacidade Finita: uma Decisão estratégica?**, RAE - Revista de Administração de Empresas**,** v. 36, n. 4, p.60-73, out/nov/dez.

Pesch, E. (1993). **Machine Learning by Schedule Decomposition**, Working Paper, Faculty of Economics and Business Administration, University of Limburg, Maastricht.

Pesch, E. e Tetzlaff, U. A. W. (1996). **Constraint Propagation Based Scheduling of Job Shops**, INFORMS Journal on Computing, Spring, 8(2), p. 144-157

Reis, J. (1996). **Uma introdução ao Scheduling**. Instituto Superior de Ciências do Trabalho eda Empresa. Lisboa. Portugal.

Ribeiro, S.A. (2005). **Algoritmo Genético e Job Shop Scheduling Aplicados ao Contexto de Sistemas Imunes Artificiais**, Dissertação, Informática, Universidade Federal do Espírito Santo – UFES.

Rocha, Miguel, Vilela, Carla e Neves José (2000). **A study of order based genetic and evolutionary algorithms in combinatorial optimization problems**, Universidade do Minho Braga. Disponível na internet: http://hdl.handle.net/1822/4288, último acesso: 10/04/2008.

Shi, G. (1997). **A Genetic Algorithm Applied to a Classic Job-Shop Scheduling Problem**, International Journal of Systems Science, 28(1), p. 25-32

Slack, N., Stuart C., Harland C., Harison A. e Jhonson R. (1997), **Administração da Produção**, Editora Atlas SA, São Paulo.

Tamaki, H. e Nishikawa, Y. (1992). **A Parallelled Genetic Algorithm Based on a Neighbourhood Model and its Application to the JobShop Scheduling**, in Männer, R. and Manderick, B. (eds) PPSN'2 Proceedings of the 2nd International Workshop on Parallel Problem Solving from Nature, Brussels, Belgium, p. 573-582.

Tubino, D.F. (1997), **Manual de Planejamento e Controle da Produção**, Editora Atlas,São Paulo.

Ulder, N. L. J., Aarts, E. H. L., andelt, H.-J., Van Laarhooven, P. J. M. e Pesch, E. (1991) **Genetic Local Search Algorithm for the Travelling Salesman Problem**, Lecture Notes in Computer Science, vol 496, p. 109- 116.

Wang, L. e Zheng, D. (2001). **An effective hybrid optimisation strategy for job-shop scheduling problems**, Computers & Operations Research, Vol. 28, p. 585-596.

Yamada, T. e Nakano, R. (1997). **Genetic Algorithms for Job Shop Scheduling Problems. Proceedings of Moderns Heuristic for Decision Support**, p. 67-81, UNICOM seminar, 18-19 March 1997, London.

Yamada, T. e Nakano, R. (1992). **A Genetic Algorithm Applicable to Large-Scale Job-Shop Problems**, in Männer, R. and Manderick, B. (eds) PPSN'2 Proceedings of

the 2nd International Workshop on Parallel Problem Solving from Nature, Brussels, p. 281-290.

Yamada, T. e Nakano, R. (1995a). **Job-Shop Scheduling by Simulated Annealing Combined with Deterministic Local Search**, MIC'95 Meta-heuristics International Conference, Hilton, Breckenridge, Colorado, USA, July 22-26, p. 344-349.

Yamada, T. e Nakano, R. (1995b). **A Genetic Algorithm with Multi-Step Crossover for Job-Shop Scheduling Problems**, GALESIA'95 Proceedings of the Int. Conf. on GAs in Eng. Sys., p. 146-151.

Yamada, T. e Nakano, R. (1996a). **Job-Shop Scheduling by Simulated Annealing Combined with Deterministic Local Search, Meta-heuristics: Theory and Applications**, Kluwer Academic Publishers, MA,USA, p. 237-248.

Yamada, T. e Nakano, R. (1996b). **Scheduling by Genetic Local Search with Multi-Step Crossover**, PPSN'IV Fourth International Conference on Parallel Problem Solving from Nature, Berlin, Germany, Sept 22-26, p. 960-969.

Yamada, T. e Nakano, R. (1996c). **A Fusion of Crossover and Local Search**, ICIT'96 IEEE International Conference on Industrial Technology, Shanghai, China, Dec 2-6, p. 426-430.

YEPES, I (2000). **Uma incursão aos algoritmos genéticos**. Disponível em: <http://www.geocities.com/igoryepes/>. Acesso em: 01/10/2007.

ANEXO A: PROGRAMA UTILIZADO PARA A IMPLANTAÇÃO DO ALGORITMO

```
&ANALYZE-SUSPEND _VERSION-NUMBER AB_v10r12 GUI
&ANALYZE-RESUME
&Scoped-define WINDOW-NAME C-Win
&ANALYZE-SUSPEND _UIB-CODE-BLOCK _CUSTOM _DEFINITIONS C-Win
/*------------------------------------------------------------------------

  File:

  Description:

  Input Parameters:
      <none>

  Output Parameters:
      <none>

  Author:

  Created:

------------------------------------------------------------------------*/
/*           This .W file was created with the Progress AppBuilder.      */
/*------------------------------------------------------------------------*/

/* Create an unnamed pool to store all the widgets created
     by this procedure. This is a good default which assures
     that this procedure's triggers and internal procedures
     will execute in this procedure's storage, and that proper
     cleanup will occur on deletion of the procedure. */

CREATE WIDGET-POOL.

/* ***************************  Definitions  ************************** */

/* Parameters Definitions ---                                           */

/* Local Variable Definitions ---                                       */
DEFINE VARIABLE cIndividuoTeste AS CHARACTER   NO-UNDO.

/* Vari veis de Controle */

/* Vari veis Auxiliares */
DEFINE VARIABLE iGeracao AS INTEGER     NO-UNDO.
DEFINE VARIABLE iQtFilhos AS INTEGER     NO-UNDO.
DEFINE VARIABLE deSomaRoleta AS DECIMAL     NO-UNDO.
DEFINE VARIABLE deMaximoFitness AS DECIMAL     NO-UNDO.
DEFINE VARIABLE chProgressBar   AS COM-HANDLE NO-UNDO.
DEFINE VARIABLE iContador AS INTEGER     NO-UNDO.

DEFINE TEMP-TABLE tt-individuo
    FIELD geracao AS INT
    FIELD genotipo AS CHAR FORMAT "X(9)" EXTENT 3
    FIELD genotipo_total AS CHAR FORMAT "X(27)"
    FIELD fitness AS DEC
    FIELD morto AS LOG
    FIELD valido AS LOG
    FIELD roleta AS DEC
    INDEX idx-fitness fitness
    INDEX idx-mata fitness DESCENDING geracao DESCENDING.

DEFINE BUFFER bfInd FOR tt-individuo.

DEFINE TEMP-TABLE tt-capacidade
    FIELD termin AS INT
    FIELD nome AS CHAR
    FIELD capacidade AS INT
    INDEX idx-principal termin.
```

```
DEFINE TEMP-TABLE tt-pedido
   FIELD termin AS INT
   FIELD berco AS INT
   FIELD quantidade AS INT
   INDEX idx-principal termin berco.

DEFINE TEMP-TABLE tt-reta
   FIELD berco AS INT
   FIELD termin AS INT
   FIELD tamanho AS DEC
   FIELD inicio AS DEC
   FIELD fim AS DEC
   INDEX idx-prim berco termin
   INDEX idx-fitness inicio.

DEFINE TEMP-TABLE tt-usa
   FIELD gene AS CHAR
   FIELD tempo-ini AS DEC
   FIELD tempo-fim AS DEC
   INDEX idx-pri gene tempo-ini.

DEFINE BUFFER bfReta FOR tt-reta.
DEFINE BUFFER bfUsa FOR tt-usa.


RUN criaTTs.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME


&ANALYZE-SUSPEND _UIB-PREPROCESSOR-BLOCK

/* ********************  Preprocessor Definitions  ******************** */

&Scoped-define PROCEDURE-TYPE Window
&Scoped-define DB-AWARE no

/* Name of designated FRAME-NAME and/or first browse and/or first query */
&Scoped-define FRAME-NAME DEFAULT-FRAME
&Scoped-define BROWSE-NAME brResult

/* Internal Tables (found by Frame, Query & Browse Queries)             */
&Scoped-define INTERNAL-TABLES tt-individuo

/* Definitions for BROWSE brResult                                      */
&Scoped-define FIELDS-IN-QUERY-brResult tt-individuo.fitness tt-individuo.genotipo_total tt-individuo.geracao tt-
individuo.roleta tt-individuo.morto tt-individuo.valido
&Scoped-define ENABLED-FIELDS-IN-QUERY-brResult
&Scoped-define SELF-NAME brResult
&Scoped-define QUERY-STRING-brResult FOR EACH tt-individuo WHERE NOT tt-individuo.morto
&Scoped-define OPEN-QUERY-brResult OPEN QUERY {&SELF-NAME} FOR EACH tt-individuo WHERE NOT tt-
individuo.morto.
&Scoped-define TABLES-IN-QUERY-brResult tt-individuo
&Scoped-define FIRST-TABLE-IN-QUERY-brResult tt-individuo


/* Definitions for FRAME DEFAULT-FRAME                                  */
&Scoped-define OPEN-BROWSERS-IN-QUERY-DEFAULT-FRAME ~
   ~{&OPEN-QUERY-brResult}

/* Standard List Definitions                                            */
&Scoped-Define ENABLED-OBJECTS btPed btCapac ElimRepet TamPopulacao ~
NumGeracoes PercMutacao brResult btVai recados PercSobrevivencia IMAGE-5 ~
RECT-2 RECT-3
&Scoped-Define DISPLAYED-OBJECTS ElimRepet TamPopulacao NumGeracoes ~
PercMutacao recados PercSobrevivencia

/* Custom List Definitions                                              */
/* List-1,List-2,List-3,List-4,List-5,List-6                            */

/* _UIB-PREPROCESSOR-BLOCK-END */
&ANALYZE-RESUME
```

```
/* ***********************  Control Definitions  ********************** */

/* Define the widget handle for the window                              */
DEFINE VAR C-Win AS WIDGET-HANDLE NO-UNDO.

/* Definitions of handles for OCX Containers                            */
DEFINE VARIABLE CtrlFrame AS WIDGET-HANDLE NO-UNDO.
DEFINE VARIABLE chCtrlFrame AS COMPONENT-HANDLE NO-UNDO.

/* Definitions of the field level widgets                               */
DEFINE BUTTON btCapac
     LABEL "Capacidade de Carga por Terminal"
     SIZE 36 BY 1.13.

DEFINE BUTTON btPed
     LABEL "Pedido"
     SIZE 36 BY 1.13.

DEFINE BUTTON btVai
     LABEL "Evoluir"
     SIZE 15 BY 1.13.

DEFINE VARIABLE NumGeracoes AS INTEGER FORMAT ">>>9":U INITIAL 200
     LABEL "N£mero de Gera‡äes"
     VIEW-AS FILL-IN
     SIZE 7 BY .79 NO-UNDO.

DEFINE VARIABLE PercMutacao AS INTEGER FORMAT ">>9":U INITIAL 10
     LABEL "Percentual de Muta‡Æo"
     VIEW-AS FILL-IN
     SIZE 7 BY .79 NO-UNDO.

DEFINE VARIABLE PercSobrevivencia AS DECIMAL FORMAT ">>9.99":U INITIAL 40
     LABEL "Percentual de Sobrevivencia"
     VIEW-AS FILL-IN
     SIZE 7 BY .79 NO-UNDO.

DEFINE VARIABLE recados AS CHARACTER FORMAT "X(256)":U
      VIEW-AS TEXT
     SIZE 78 BY .42
     FONT 11 NO-UNDO.

DEFINE VARIABLE TamPopulacao AS INTEGER FORMAT ">>>9":U INITIAL 100
     LABEL "Tamanho da Popula‡Æo"
     VIEW-AS FILL-IN
     SIZE 7 BY .79 NO-UNDO.

DEFINE IMAGE IMAGE-5
     FILENAME "image/rovina.bmp":U
     STRETCH-TO-FIT
     SIZE 9.14 BY 2.

DEFINE RECTANGLE RECT-2
     EDGE-PIXELS 2 GRAPHIC-EDGE  NO-FILL
     SIZE 38 BY 5.75.

DEFINE RECTANGLE RECT-3
     EDGE-PIXELS 2 GRAPHIC-EDGE  NO-FILL
     SIZE 39 BY 5.75.

DEFINE VARIABLE ElimRepet AS LOGICAL INITIAL yes
     LABEL "Elimina Gen¢tipos Repetidos"
     VIEW-AS TOGGLE-BOX
     SIZE 22 BY .83 NO-UNDO.

/* Query definitions                                                    */
&ANALYZE-SUSPEND
DEFINE QUERY brResult FOR
      tt-individuo SCROLLING.
&ANALYZE-RESUME

/* Browse definitions                                                   */
DEFINE BROWSE brResult
&ANALYZE-SUSPEND _UIB-CODE-BLOCK _DISPLAY-FIELDS brResult C-Win _FREEFORM
  QUERY brResult DISPLAY
      tt-individuo.fitness COLUMN-LABEL "Fitness" WIDTH 12 FORMAT ">>>>,>>>,>>>,>>9.9999"
```

```
   tt-individuo.genotipo_total COLUMN-LABEL "Gen¢tipo" WIDTH 30
   tt-individuo.geracao COLUMN-LABEL "Gera‡Æo" WIDTH 7
   tt-individuo.roleta COLUMN-LABEL "Roleta"  WIDTH 12 FORMAT ">>>>,>>>,>>>,>>9.99"
   tt-individuo.morto COLUMN-LABEL "Est  Morto?" FORMAT "Sim/NÆo" WIDTH 12
   tt-individuo.valido COLUMN-LABEL "□ V lido?" FORMAT "Sim/NÆo" WIDTH 8
/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME
   WITH NO-ROW-MARKERS SIZE 78 BY 11
        FONT 1
        TITLE "Resultados".


/* ************************  Frame Definitions  *********************** */

DEFINE FRAME DEFAULT-FRAME
     btPed AT ROW 5.75 COL 44 WIDGET-ID 62
     btCapac AT ROW 4.5 COL 44 WIDGET-ID 60
     ElimRepet AT ROW 8.5 COL 15 WIDGET-ID 44
     TamPopulacao AT ROW 4.5 COL 28 COLON-ALIGNED WIDGET-ID 10
     NumGeracoes AT ROW 5.5 COL 28 COLON-ALIGNED WIDGET-ID 26
     PercMutacao AT ROW 6.5 COL 28 COLON-ALIGNED WIDGET-ID 28
     brResult AT ROW 10 COL 3 WIDGET-ID 200
     btVai AT ROW 21 COL 3 WIDGET-ID 30
     recados AT ROW 22.17 COL 3 NO-LABEL WIDGET-ID 34
     PercSobrevivencia AT ROW 7.5 COL 28 COLON-ALIGNED WIDGET-ID 42
     "Dados do AG" VIEW-AS TEXT
          SIZE 10 BY .54 AT ROW 3.75 COL 4 WIDGET-ID 20
     "Rovina - Scheduling de Portos" VIEW-AS TEXT
          SIZE 64 BY 2 AT ROW 1.5 COL 17 WIDGET-ID 52
          FGCOLOR 2 FONT 13
     "Dados do Problema" VIEW-AS TEXT
          SIZE 14 BY .54 AT ROW 3.75 COL 43 WIDGET-ID 58
     IMAGE-5 AT ROW 1.5 COL 3 WIDGET-ID 54
     RECT-2 AT ROW 4 COL 3 WIDGET-ID 18
     RECT-3 AT ROW 4 COL 42 WIDGET-ID 56
    WITH 1 DOWN NO-BOX KEEP-TAB-ORDER OVERLAY
         SIDE-LABELS NO-UNDERLINE THREE-D
         AT COL 1 ROW 1
         SIZE 81.57 BY 21.58
         BGCOLOR 15 FONT 1 WIDGET-ID 100.


/* ********************** Procedure Settings ************************ */

&ANALYZE-SUSPEND _PROCEDURE-SETTINGS
/* Settings for THIS-PROCEDURE
   Type: Window
   Allow: Basic,Browse,DB-Fields,Window,Query
   Other Settings: COMPILE
 */
&ANALYZE-RESUME _END-PROCEDURE-SETTINGS

/* ************************  Create Window  ************************** */

&ANALYZE-SUSPEND _CREATE-WINDOW
IF SESSION:DISPLAY-TYPE = "GUI":U THEN
  CREATE WINDOW C-Win ASSIGN
         HIDDEN             = YES
         TITLE              = "Rovina - Scheduling de Portos"
         HEIGHT             = 21.58
         WIDTH              = 81.57
         MAX-HEIGHT         = 31.29
         MAX-WIDTH          = 182.86
         VIRTUAL-HEIGHT     = 31.29
         VIRTUAL-WIDTH      = 182.86
         RESIZE             = no
         SCROLL-BARS        = no
         STATUS-AREA        = no
         BGCOLOR            = ?
         FGCOLOR            = ?
         KEEP-FRAME-Z-ORDER = yes
         THREE-D            = yes
         MESSAGE-AREA       = no
         SENSITIVE          = yes.
ELSE {&WINDOW-NAME} = CURRENT-WINDOW.
/* END WINDOW DEFINITION                                        */
```

```
&ANALYZE-RESUME


/* ***********  Runtime Attributes and AppBuilder Settings  ********** */

&ANALYZE-SUSPEND _RUN-TIME-ATTRIBUTES
/* SETTINGS FOR WINDOW C-Win
  VISIBLE,,RUN-PERSISTENT                                    */
/* SETTINGS FOR FRAME DEFAULT-FRAME
   FRAME-NAME Custom                                         */
/* BROWSE-TAB brResult PercMutacao DEFAULT-FRAME */
/* SETTINGS FOR FILL-IN recados IN FRAME DEFAULT-FRAME
   ALIGN-L                                                   */
IF SESSION:DISPLAY-TYPE = "GUI":U AND VALID-HANDLE(C-Win)
THEN C-Win:HIDDEN = no.

/* _RUN-TIME-ATTRIBUTES-END */
&ANALYZE-RESUME


/* Setting information for Queries and Browse Widgets fields            */

&ANALYZE-SUSPEND _QUERY-BLOCK BROWSE brResult
/* Query rebuild information for BROWSE brResult
    _START_FREEFORM
OPEN QUERY {&SELF-NAME} FOR EACH tt-individuo WHERE NOT tt-individuo.morto.
    _END_FREEFORM
    _Query           is OPENED
*/  /* BROWSE brResult */
&ANALYZE-RESUME




/* **********************  Create OCX Containers  ********************** */

&ANALYZE-SUSPEND _CREATE-DYNAMIC

&IF "{&OPSYS}" = "WIN32":U AND "{&WINDOW-SYSTEM}" NE "TTY":U &THEN

CREATE CONTROL-FRAME CtrlFrame ASSIGN
      FRAME          = FRAME DEFAULT-FRAME:HANDLE
      ROW            = 21.25
      COLUMN         = 19
      HEIGHT         = .75
      WIDTH          = 62
      WIDGET-ID      = 36
      HIDDEN         = no
      SENSITIVE      = yes.
     CtrlFrame:NAME = "CtrlFrame":U .
/* CtrlFrame OCXINFO:CREATE-CONTROL from: {0713E8D2-850A-101B-AFC0-4210102A8DA7} type: ProgressBar */
     CtrlFrame:MOVE-AFTER(brResult:HANDLE IN FRAME DEFAULT-FRAME).

&ENDIF

&ANALYZE-RESUME /* End of _CREATE-DYNAMIC */


/* ************************  Control Triggers  ************************ */

&Scoped-define SELF-NAME C-Win
&ANALYZE-SUSPEND _UIB-CODE-BLOCK _CONTROL C-Win C-Win
ON END-ERROR OF C-Win /* Rovina - Scheduling de Portos */
OR ENDKEY OF {&WINDOW-NAME} ANYWHERE DO:
  /* This case occurs when the user presses the "Esc" key.
     In a persistently run window, just ignore this.  If we did not, the
     application would exit. */
  IF THIS-PROCEDURE:PERSISTENT THEN RETURN NO-APPLY.
END.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME


&ANALYZE-SUSPEND _UIB-CODE-BLOCK _CONTROL C-Win C-Win
```

```
ON WINDOW-CLOSE OF C-Win /* Rovina - Scheduling de Portos */
DO:
  /* This event will close the window and terminate the procedure.  */
  APPLY "CLOSE":U TO THIS-PROCEDURE.
  QUIT.
  RETURN NO-APPLY.
END.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME


&Scoped-define SELF-NAME btCapac
&ANALYZE-SUSPEND _UIB-CODE-BLOCK _CONTROL btCapac C-Win
ON CHOOSE OF btCapac IN FRAME DEFAULT-FRAME /* Capacidade de Carga por Terminal */
DO:

    RUN cap_carga.w (INPUT-OUTPUT TABLE tt-capacidade).

END.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME


&Scoped-define SELF-NAME btPed
&ANALYZE-SUSPEND _UIB-CODE-BLOCK _CONTROL btPed C-Win
ON CHOOSE OF btPed IN FRAME DEFAULT-FRAME /* Pedido */
DO:

    RUN pedidos.w (INPUT TABLE tt-capacidade,
                   INPUT-OUTPUT TABLE tt-pedido).

END.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME


&Scoped-define SELF-NAME btVai
&ANALYZE-SUSPEND _UIB-CODE-BLOCK _CONTROL btVai C-Win
ON CHOOSE OF btVai IN FRAME DEFAULT-FRAME /* Evoluir */
DO:

    DEFINE VARIABLE lInicia AS LOGICAL     NO-UNDO INITIAL YES.
    DEFINE VARIABLE iCont AS INTEGER     NO-UNDO.
    DEFINE VARIABLE iTime AS INTEGER     NO-UNDO.

    form
      tt-individuo.fitness COLUMN-LABEL "Fitness" FORMAT ">>>>>>>>>>>>9.9999"
      tt-individuo.genotipo COLUMN-LABEL "Gen¢tipo"
      tt-individuo.geracao COLUMN-LABEL "Gera‡Æo"
      tt-individuo.valido COLUMN-LABEL "□ V lido?" FORMAT "Sim/NÆo"
    with no-box down frame fIndividuo STREAM-IO.

    ASSIGN iTime = TIME.

    FOR EACH tt-individuo:
      DELETE tt-individuo.
    END.

    ASSIGN INPUT FRAME {&FRAME-NAME} TamPopulacao NumGeracoes PercMutacao PercSobrevivencia ElimRepet.

    IF NumGeracoes > 0 THEN DO:
      ASSIGN chProgressBar:Max = NumGeracoes
           chProgressBar:VALUE = 0
           chProgressBar:VISIBLE = TRUE.
    END.

    /* Valida‡Æo dos Par ƒmetros */
    IF TamPopulacao <= 1 THEN DO:
      MESSAGE "Tamanho da Popula‡Æo deve ser maior que 1"
        VIEW-AS ALERT-BOX INFO BUTTONS OK.
      RETURN "NOK".
    END.
    IF NumGeracoes < 0 THEN DO:
```

```
    MESSAGE "N£mero de Gera‡äes deve ser positivo"
       VIEW-AS ALERT-BOX INFO BUTTONS OK.
    RETURN "NOK".
  END.
  IF PercMutacao < 0 THEN DO:
    MESSAGE "Percentual de Muta‡Æo deve ser positivo"
       VIEW-AS ALERT-BOX INFO BUTTONS OK.
    RETURN "NOK".
  END.

  blk-geracao:
  REPEAT iGeracao = 0 TO NumGeracoes:
    ASSIGN recados:SCREEN-VALUE = "Gera‡Æo: " + STRING(iGeracao).
    ASSIGN chProgressBar:VALUE = iGeracao.
    IF lInicia THEN DO:
       RUN inicializaPopulacao.
    END.
    ELSE DO:
       ASSIGN iCont = 1.
       FOR EACH tt-individuo
          WHERE NOT tt-individuo.morto:
          ASSIGN iCont = iCont + 1.
       END.
       REPEAT iQtFilhos = iCont TO (TamPopulacao * 2):
          ASSIGN recados:SCREEN-VALUE = "Gera‡Æo: " + STRING(iGeracao) + "     Gerando Filho: " + STRING(iQtFilhos).
          RUN crossover.
       END.
    END.
    ASSIGN recados:SCREEN-VALUE = "Gera‡Æo: " + STRING(iGeracao) + "     Avaliando indiv¡duos...".
    RUN avaliarIndividuos.
    IF NOT lInicia THEN DO:
       ASSIGN recados:SCREEN-VALUE = "Gera‡Æo: " + STRING(iGeracao) + "     Controlando a popula‡Æo...".
       RUN controlaPopulacao.
    END.
    ASSIGN recados:SCREEN-VALUE = "Gera‡Æo: " + STRING(iGeracao) + "     Calculando Roleta...".
    RUN calculaRoleta.
    ASSIGN lInicia = NO.
    {&OPEN-QUERY-brResult}
    IF CAN-FIND(FIRST tt-individuo
              WHERE tt-individuo.fitness = 0
              AND   tt-individuo.valido
              AND   NOT tt-individuo.morto) THEN LEAVE blk-geracao.
  END.

  {&OPEN-QUERY-brResult}
  ASSIGN recados:SCREEN-VALUE = "".
  IF NumGeracoes > 0 THEN DO:
    ASSIGN chProgressBar:VALUE = 0
           chProgressBar:VISIBLE = FALSE.
  END.

  OUTPUT TO VALUE(SESSION:TEMP-DIRECTORY + "rovina_scheduling_porto_result.txt") CONVERT TARGET "iso8859-
1".

  PUT UNFORMATTED "RESULTADOS DA EXECU€ÇO DO PROGRAMA ROVINA - SCHEDULING DE PORTOS" SKIP
              "================================================================" SKIP(2)
              "PARAMETRIZA€åES:" SKIP(1)
              "Tamanho da Popula‡Æo:          " TamPopulacao SKIP
              "N£mero de Gera‡äes:           " NumGeracoes SKIP
              "Percentual de Muta‡Æo:         " PercMutacao SKIP
              "Percentual de Sobrevivˆncia:    " PercSobrevivencia SKIP
              "Elimina Gen¢tipos Repetidos:    " STRING(ElimRepet,"Sim/NÆo") SKIP
              "DADOS DA EXECU€ÇO:" SKIP(1)
              "Hora In¡cio:             " STRING(iTime,"HH:MM:SS") SKIP
              "Hora Fim:                " STRING(TIME,"HH:MM:SS") SKIP
              "Tempo Total de Execu‡Æo: " STRING((TIME - iTime),"HH:MM:SS") SKIP(2)
              "POPULA€ÇO FINAL:" SKIP(1).

    FOR EACH tt-individuo WHERE NOT tt-individuo.morto:
       DISP tt-individuo.fitness
          tt-individuo.genotipo
          tt-individuo.geracao
          tt-individuo.valido
          WITH FRAME fIndividuo.
       DOWN WITH FRAME fIndividuo.
    END.
```

```
   OUTPUT CLOSE.

END.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME


&Scoped-define BROWSE-NAME brResult
&UNDEFINE SELF-NAME

&ANALYZE-SUSPEND _UIB-CODE-BLOCK _CUSTOM _MAIN-BLOCK C-Win


/* ***************************  Main Block  *************************** */

/* Set CURRENT-WINDOW: this will parent dialog-boxes and frames.        */
ASSIGN CURRENT-WINDOW                = {&WINDOW-NAME}
       THIS-PROCEDURE:CURRENT-WINDOW = {&WINDOW-NAME}.

/* The CLOSE event can be used from inside or outside the procedure to  */
/* terminate it.                                                        */
ON CLOSE OF THIS-PROCEDURE
  RUN disable_UI.

/* Best default for GUI applications is...                              */
PAUSE 0 BEFORE-HIDE.

/* Now enable the interface and wait for the exit condition.            */
/* (NOTE: handle ERROR and END-KEY so cleanup code will always fire.    */
MAIN-BLOCK:
DO ON ERROR   UNDO MAIN-BLOCK, LEAVE MAIN-BLOCK
   ON END-KEY UNDO MAIN-BLOCK, LEAVE MAIN-BLOCK:
  RUN enable_UI.
  IF NOT THIS-PROCEDURE:PERSISTENT THEN
    WAIT-FOR CLOSE OF THIS-PROCEDURE.
END.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME


/* ************************  Internal Procedures  ********************** */

&ANALYZE-SUSPEND _UIB-CODE-BLOCK _PROCEDURE avaliarIndividuos C-Win
PROCEDURE avaliarIndividuos :
/*------------------------------------------------------------------------------
  Purpose:
  Parameters:  <none>
  Notes:
------------------------------------------------------------------------------*/

    /*DEFINE VARIABLE iCont AS INTEGER     NO-UNDO.
    DEFINE VARIABLE cGene AS CHARACTER   NO-UNDO.
    DEFINE VARIABLE iGene AS INTEGER     NO-UNDO.
    DEFINE VARIABLE iGenotipo AS INTEGER     NO-UNDO.
    DEFINE VARIABLE deAcum AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE deUltimo AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE iAux AS INTEGER     NO-UNDO.*/

    DEFINE VARIABLE deQtdHoras AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE iT AS INTEGER     NO-UNDO EXTENT 3.
    DEFINE VARIABLE iB AS INTEGER     NO-UNDO INITIAL 1.
    DEFINE VARIABLE cGene AS CHARACTER   NO-UNDO.
    DEFINE VARIABLE iGene AS INTEGER     NO-UNDO.
    DEFINE VARIABLE deTempo AS DECIMAL     NO-UNDO EXTENT 3.
    DEFINE VARIABLE deFolga AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE deTempoTot AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE iAux AS INTEGER     NO-UNDO.
    DEFINE VARIABLE deFim AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE deAux AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE deFolgaAux AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE iRepeat AS INTEGER     NO-UNDO.
    DEFINE VARIABLE iRpt AS INTEGER     NO-UNDO.
    DEFINE VARIABLE lLog AS LOGICAL     NO-UNDO INITIAL NO.
```

```
   FOR EACH tt-individuo
      WHERE tt-individuo.fitness = -999
      AND   NOT tt-individuo.valido:
      ASSIGN tt-individuo.morto = YES
             tt-individuo.fitness = 999999.
   END.

   ASSIGN iContador = 0.

   FOR EACH tt-individuo
      WHERE tt-individuo.fitness = -999
      AND   tt-individuo.valido:

      ASSIGN iContador = iContador + 1.
      ASSIGN recados:SCREEN-VALUE IN FRAME {&FRAME-NAME} = "Gera‡Æo: " + STRING(iGeracao) + "     Avaliando
indiv¡duos... " + STRING(iContador).

      FOR EACH tt-reta:
         DELETE tt-reta.
      END.
      FOR EACH tt-usa:
         DELETE tt-usa.
      END.

      /*REPEAT iGenotipo = 1 TO 3:

         ASSIGN deAcum = 0.

         REPEAT iCont = 1 TO 9:

            ASSIGN cGene = SUBSTRING(tt-individuo.genotipo[iGenotipo],iCont,1)
                   iGene = ASC(cGene) - 64.

            FIND FIRST tt-capacidade WHERE tt-capacidade.termin = iGene.
            FIND FIRST tt-pedido
               WHERE tt-pedido.termin = tt-capacidade.termin
               AND   tt-pedido.berco = iGenotipo.

            ASSIGN deQtdHoras = (tt-pedido.quantidade / tt-capacidade.capacidade).

            FIND FIRST tt-reta
               WHERE tt-reta.berco = iGenotipo
               AND   tt-reta.termin = iGene NO-ERROR.
            IF NOT AVAIL tt-reta THEN DO:
               CREATE tt-reta.
               ASSIGN tt-reta.berco = iGenotipo
                      tt-reta.termin = iGene.
            END.

            ASSIGN tt-reta.inicio = deAcum
                   tt-reta.tamanho = deQtdHoras
                   deAcum = deAcum + deQtdHoras
                   tt-reta.fim = deAcum.

         END.

      END.

      REPEAT iAux = 1 TO 9:
         FOR EACH tt-reta USE-INDEX idx-fitness
            WHERE tt-reta.tamanho > 0
            AND   tt-reta.termin = iAux:

            FOR EACH bfReta USE-INDEX idx-fitness
               WHERE bfReta.inicio < tt-reta.fim
               AND   bfReta.inicio > tt-reta.inicio
               AND   bfReta.tamanho > 0
               AND   bfReta.termin = iAux
               AND ROWID(bfReta) <> ROWID(tt-reta):
               ASSIGN tt-individuo.fitness = tt-individuo.fitness +
                                    (tt-reta.fim - bfReta.inicio).
            END.
         END.
      END.*/
```

```
IF tt-individuo.genotipo_total = cIndividuoTeste THEN
   ASSIGN lLog = YES.
ELSE
   ASSIGN lLog = NO.

ASSIGN iT[1] = 1
       iT[2] = 1
       iT[3] = 1
       deTempoTot = 0.

IF lLog THEN DO:
   OUTPUT TO VALUE(SESSION:TEMP-DIRECTORY + "avalia_ind_teste.txt") CONVERT TARGET "iso8859-1".
   PUT UNFORMATTED "001 Inicializando...".
END.

blk-rpt:
REPEAT iRepeat = 1 TO 5000:

   IF lLog THEN DO:
      PUT UNFORMATTED SKIP(2) "002 Procura ber‡o (iRepeat: " iRepeat " iT[1]=" iT[1] " iT[2]=" iT[2] " iT[3]=" iT[3] ")".
   END.

   IF iT[1] > 9
   AND iT[2] > 9
   AND iT[3] > 9 THEN LEAVE blk-rpt.

   ASSIGN deFim = 999999
          deFolgaAux = 0.

   /* Escolhe aquele que termina primeiro */
   blk-escolhe:
   REPEAT iAux = 1 TO 3:

      IF lLog THEN DO:
         PUT UNFORMATTED SKIP(2) "003 Avaliando ber‡o (iAux: " iAux " iT[iAux]=" iT[iAux] ")".
      END.

      IF iT[iAux] > 9 THEN NEXT blk-escolhe.

      blk-gene:
      REPEAT iRpt = 1 TO 9:

         IF lLog THEN DO:
            PUT UNFORMATTED SKIP "004 Avalia pr¢ximo terminal (iRpt: " iRpt " iT[iAux]=" iT[iAux] ")".
         END.

         IF iT[iAux] > 9 THEN NEXT blk-gene.

         ASSIGN cGene = SUBSTRING(tt-individuo.genotipo[iAux],iT[iAux],1)
                iGene = ASC(cGene) - 64.

         IF lLog THEN DO:
            PUT UNFORMATTED SKIP "005 cGene: " cGene " iGene: " iGene.
         END.

         FIND FIRST tt-capacidade WHERE tt-capacidade.termin = iGene.
         FIND FIRST tt-pedido
            WHERE tt-pedido.termin = tt-capacidade.termin
            AND   tt-pedido.berco = iAux.
         ASSIGN deQtdHoras = (tt-pedido.quantidade / tt-capacidade.capacidade).

         IF lLog THEN DO:
            PUT UNFORMATTED SKIP "006 Tem pedido? (deQtdHoras: " deQtdHoras " tt-pedido.quantidade: " tt-
pedido.quantidade " tt-capacidade.capacidade: " tt-capacidade.capacidade ")".
         END.

         IF deQtdHoras > 0 THEN DO:
            RUN procuraUsado (INPUT cGene,
                              INPUT deTempo[iAux],
                              INPUT (deTempo[iAux] + deQtdHoras),
                              OUTPUT deAux).

            IF lLog THEN DO:
               PUT UNFORMATTED SKIP "007 Procura conflito (deTempo[iAux]: " deTempo[iAux] " deAux: " deAux ")".
            END.
```

```
                IF deAux > 0 THEN DO:
                   IF lLog THEN DO:
                      PUT UNFORMATTED SKIP "008  a maior folga? [(deAux - deTempo[iAux]) > deFolgaAux: " ((deAux -
deTempo[iAux]) > deFolgaAux) "]".
                   END.
                   IF (deAux - deTempo[iAux]) > deFolgaAux THEN DO:
                      ASSIGN deFolgaAux = deAux - deTempo[iAux]
                             iB = iAux
                             deFim = deAux.
                      IF lLog THEN DO:
                         PUT UNFORMATTED SKIP "009 Usa o de maior folga (deFolgaAux: " deFolgaAux " iB: " iB " deFim: "
deFim ")".
                      END.
                   END.
                END.

                IF deFolgaAux = 0 THEN DO:
                   IF lLog THEN DO:
                      PUT UNFORMATTED SKIP "010  o que termina antes? [(deTempo[iAux] + deQtdHoras) < deFim: "
((deTempo[iAux] + deQtdHoras) < deFim) "]".
                   END.
                   IF (deTempo[iAux] + deQtdHoras) < deFim THEN DO:
                      ASSIGN iB = iAux
                             deFim = (deTempo[iAux] + deQtdHoras).
                      IF lLog THEN DO:
                         PUT UNFORMATTED SKIP "011 Usa o que termina antes (iB: " iB " deFim: " deFim ")".
                      END.
                   END.
                END.
                IF lLog THEN DO:
                   PUT UNFORMATTED SKIP "012 Verifica o pr¢ximo".
                END.
                LEAVE blk-gene.
             END.
             ELSE DO:
                ASSIGN iT[iAux] = iT[iAux] + 1.
                IF lLog THEN DO:
                   PUT UNFORMATTED SKIP "013 Pega pr¢ximo gene (iT[iAux]: " iT[iAux] ")".
                END.
             END.
          END.
       END.

       IF lLog THEN DO:
          PUT UNFORMATTED SKIP(2) "014 Carrega ber‡o (iB=" iB " iT[iB]: " iT[iB] ")".
       END.
       IF iT[iB] > 9 THEN NEXT blk-rpt.

       ASSIGN cGene = SUBSTRING(tt-individuo.genotipo[iB],iT[iB],1)
              iGene = ASC(cGene) - 64.

       IF lLog THEN DO:
          PUT UNFORMATTED SKIP "015 Usa terminal (cGene: " cGene " iGene: " iGene " deFolgaAux: " deFolgaAux ")".
       END.

       IF deFolgaAux > 0 THEN DO:
          ASSIGN deFolga = deFolga + deFolgaAux
                 deTempo[iB] = deTempo[iB] + deFolgaAux.
          IF lLog THEN DO:
             PUT UNFORMATTED SKIP "016 Atualiza folga (deFolga: " deFolga " deTempo[iB]: " deTempo[iB] ")".
          END.
       END.

       FIND FIRST tt-capacidade WHERE tt-capacidade.termin = iGene.
       FIND FIRST tt-pedido
          WHERE tt-pedido.termin = tt-capacidade.termin
          AND   tt-pedido.berco = iB.
       ASSIGN deQtdHoras = (tt-pedido.quantidade / tt-capacidade.capacidade).

       IF lLog THEN DO:
          PUT UNFORMATTED SKIP "017 Valida Qtd Horas (deQtdHoras: " deQtdHoras " tt-pedido.quantidade: " tt-
pedido.quantidade " tt-capacidade.capacidade: " tt-capacidade.capacidade ")".
       END.
       IF deQtdHoras > 0 THEN DO:

         CREATE tt-usa.
```

```
            ASSIGN tt-usa.gene = cGene
                   tt-usa.tempo-ini = deTempo[iB]
                   tt-usa.tempo-fim = deTempo[iB] + deQtdHoras.

            ASSIGN deTempo[iB] = deTempo[iB] + deQtdHoras
                   iT[iB] = iT[iB] + 1.

            IF lLog THEN DO:
               PUT UNFORMATTED SKIP "018 Atualiza tempo do ber‡o (deTempo[iB]: " deTempo[iB] " iT[iB]: " iT[iB] ")".
            END.
          END.

       END.

       IF lLog THEN DO:
          PUT UNFORMATTED SKIP "019 Termina execu‡Æo (Fitness: " deFolga ")".
          OUTPUT CLOSE.
       END.

       ASSIGN tt-individuo.fitness = deFolga.

    END.

    RETURN "OK".


END PROCEDURE.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME

&ANALYZE-SUSPEND _UIB-CODE-BLOCK _PROCEDURE calculaRoleta C-Win
PROCEDURE calculaRoleta :
/*------------------------------------------------------------------------------
  Purpose:
  Parameters:  <none>
  Notes:
------------------------------------------------------------------------------*/

    ASSIGN deMaximoFitness = 0.
    FOR EACH tt-individuo
       WHERE NOT tt-individuo.morto:
       IF tt-individuo.fitness > deMaximoFitness THEN
          ASSIGN deMaximoFitness = tt-individuo.fitness.
    END.

    ASSIGN deMaximoFitness = deMaximoFitness + 1
           deSomaRoleta = 0.

    FOR EACH tt-individuo
       WHERE NOT tt-individuo.morto:
       ASSIGN tt-individuo.roleta = deMaximoFitness - tt-individuo.fitness
              deSomaRoleta = deSomaRoleta + tt-individuo.roleta.
    END.

    RETURN "OK".


END PROCEDURE.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME

&ANALYZE-SUSPEND _UIB-CODE-BLOCK _PROCEDURE controlaPopulacao C-Win
PROCEDURE controlaPopulacao :
/*------------------------------------------------------------------------------
  Purpose:
  Parameters:  <none>
  Notes:
------------------------------------------------------------------------------*/

    DEFINE VARIABLE iNumIndividuos AS INTEGER     NO-UNDO INITIAL 0.
    DEFINE VARIABLE iNumIndGeracaoAnt AS INTEGER     NO-UNDO INITIAL 0.
    DEFINE VARIABLE iNumVivos AS INTEGER     NO-UNDO INITIAL 0.

    /* Limpa os j  mortos para melhorar a performance */
```

```
   FOR EACH tt-individuo
      WHERE tt-individuo.morto
      AND   tt-individuo.geracao < (iGeracao - 10):
      DELETE tt-individuo.
   END.

   FOR EACH tt-individuo
      WHERE NOT tt-individuo.morto:
      ASSIGN iNumIndividuos = iNumIndividuos + 1.
   END.

   /* Primeiro mata os indiv¡duos inv lidos */
   FOR EACH tt-individuo USE-INDEX idx-mata
      WHERE NOT tt-individuo.morto
      AND NOT tt-individuo.valido:
      ASSIGN tt-individuo.morto = YES
             iNumIndividuos = iNumIndividuos - 1.
      IF iNumIndividuos = TamPopulacao THEN
         LEAVE.
   END.

   /* Depois mata os indiv¡duos piores que tem gen¢tipo repetido */
   IF ElimRepet THEN DO:
      FOR EACH tt-individuo USE-INDEX idx-mata
         WHERE NOT tt-individuo.morto:
         IF CAN-FIND(FIRST bfInd
                     WHERE bfInd.genotipo[1] = tt-individuo.genotipo[1]
                     AND   NOT bfInd.morto
                     AND   ROWID(bfInd) <> ROWID(tt-individuo)) THEN DO:
            ASSIGN tt-individuo.morto = YES
                   iNumIndividuos = iNumIndividuos - 1.
            IF iNumIndividuos = TamPopulacao THEN
               LEAVE.
         END.
      END.
   END.

   /* Deixa somente no m ximo um percentual da popula‡Æo anterior */
   IF iNumIndividuos > TamPopulacao THEN DO:
      FOR EACH tt-individuo USE-INDEX idx-mata
         WHERE NOT tt-individuo.morto
         AND   tt-individuo.geracao < iGeracao:
         ASSIGN iNumIndGeracaoAnt = iNumIndGeracaoAnt + 1.
      END.
      ASSIGN iNumVivos = INT(iNumIndGeracaoAnt * (PercSobrevivencia / 100)).
      FOR EACH tt-individuo USE-INDEX idx-mata
         WHERE NOT tt-individuo.morto
         AND   tt-individuo.geracao < iGeracao:
         ASSIGN tt-individuo.morto = YES
                iNumIndividuos = iNumIndividuos - 1
                iNumIndGeracaoAnt = iNumIndGeracaoAnt - 1.
         IF iNumIndividuos = TamPopulacao
         OR iNumIndGeracaoAnt <= iNumVivos THEN
            LEAVE.
      END.
   END.

   /* Se ainda precisar matar mais indiv¡duos, mata os piores */
   IF iNumIndividuos > TamPopulacao THEN DO:
      FOR EACH tt-individuo USE-INDEX idx-mata
         WHERE NOT tt-individuo.morto:
         ASSIGN tt-individuo.morto = YES
                iNumIndividuos = iNumIndividuos - 1.
         IF iNumIndividuos = TamPopulacao THEN
            LEAVE.
      END.
   END.

   RETURN "OK".

END PROCEDURE.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME

&ANALYZE-SUSPEND _UIB-CODE-BLOCK _PROCEDURE control_load C-Win  _CONTROL-LOAD
```

```
PROCEDURE control_load :
/*------------------------------------------------------------------------------
  Purpose:     Load the OCXs
  Parameters:  <none>
  Notes:       Here we load, initialize and make visible the
               OCXs in the interface.
------------------------------------------------------------------------------*/

&IF "{&OPSYS}" = "WIN32":U AND "{&WINDOW-SYSTEM}" NE "TTY":U &THEN
DEFINE VARIABLE UIB_S    AS LOGICAL    NO-UNDO.
DEFINE VARIABLE OCXFile  AS CHARACTER  NO-UNDO.

OCXFile = SEARCH( "menu.wrx":U ).
IF OCXFile = ? THEN
  OCXFile = SEARCH(SUBSTRING(THIS-PROCEDURE:FILE-NAME, 1,
                   R-INDEX(THIS-PROCEDURE:FILE-NAME, ".":U), "CHARACTER":U) + "wrx":U).

IF OCXFile <> ? THEN
DO:
  ASSIGN
    chCtrlFrame = CtrlFrame:COM-HANDLE
    UIB_S = chCtrlFrame:LoadControls( OCXFile, "CtrlFrame":U)
  .
  RUN initialize-controls IN THIS-PROCEDURE NO-ERROR.
END.
ELSE MESSAGE "menu.wrx":U SKIP(1)
             "The binary control file could not be found. The controls cannot be loaded."
             VIEW-AS ALERT-BOX TITLE "Controls Not Loaded".

&ENDIF

END PROCEDURE.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME

&ANALYZE-SUSPEND _UIB-CODE-BLOCK _PROCEDURE criaTTs C-Win
PROCEDURE criaTTs :
/*------------------------------------------------------------------------------
  Purpose:
  Parameters:  <none>
  Notes:
------------------------------------------------------------------------------*/

DEFINE VARIABLE iAux AS INTEGER     NO-UNDO.
DEFINE VARIABLE iCont AS INTEGER     NO-UNDO.

IF SEARCH("teste.tst") <> ? THEN DO:
   CREATE tt-capacidade.
   ASSIGN tt-capacidade.termin = 1
          tt-capacidade.nome = "SV - APPA Farelo"
          tt-capacidade.capacidade = 500.

   CREATE tt-capacidade.
   ASSIGN tt-capacidade.termin = 2
          tt-capacidade.nome = "SH - APPA"
          tt-capacidade.capacidade = 2000.

   CREATE tt-capacidade.
   ASSIGN tt-capacidade.termin = 3
          tt-capacidade.nome = "CB - Coimbra"
          tt-capacidade.capacidade = 2000.

   CREATE tt-capacidade.
   ASSIGN tt-capacidade.termin = 4
          tt-capacidade.nome = "CG - Cargil"
          tt-capacidade.capacidade = 2000.

   CREATE tt-capacidade.
   ASSIGN tt-capacidade.termin = 5
          tt-capacidade.nome = "CL - CBL"
          tt-capacidade.capacidade = 2000.

   CREATE tt-capacidade.
   ASSIGN tt-capacidade.termin = 6
          tt-capacidade.nome = "CO - Coamo"
```

```
        tt-capacidade.capacidade = 2000.

    CREATE tt-capacidade.
    ASSIGN tt-capacidade.termin = 7
        tt-capacidade.nome = "CS - Centro Sul"
        tt-capacidade.capacidade = 2000.

    CREATE tt-capacidade.
    ASSIGN tt-capacidade.termin = 8
        tt-capacidade.nome = "CT - Cotrigua‡£"
        tt-capacidade.capacidade = 2000.

    CREATE tt-capacidade.
    ASSIGN tt-capacidade.termin = 9
        tt-capacidade.nome = "PA - AGTL"
        tt-capacidade.capacidade = 1000.
END.
ELSE DO:
    REPEAT iAux = 1 TO 9:
        CREATE tt-capacidade.
        ASSIGN tt-capacidade.termin = iAux
            tt-capacidade.capacidade = 1.
    END.
END.

REPEAT iAux = 1 TO 9:
    REPEAT iCont = 1 TO 3:
        CREATE tt-pedido.
        ASSIGN tt-pedido.termin = iAux
            tt-pedido.berco = iCont
            tt-pedido.quantidade = 0.
    END.
END.

IF SEARCH("teste.tst") <> ? THEN DO:
    RUN xunxeira.
END.

END PROCEDURE.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME

&ANALYZE-SUSPEND _UIB-CODE-BLOCK _PROCEDURE crossover C-Win
PROCEDURE crossover :
/*------------------------------------------------------------------------------
  Purpose:
  Parameters:  <none>
  Notes:
------------------------------------------------------------------------------*/

    DEFINE VARIABLE iRoleta1 AS INTEGER     NO-UNDO.
    DEFINE VARIABLE iRoleta2 AS INTEGER     NO-UNDO.
    DEFINE VARIABLE deSoma AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE cPai AS CHARACTER   NO-UNDO EXTENT 3.
    DEFINE VARIABLE cMae AS CHARACTER   NO-UNDO EXTENT 3.
    DEFINE VARIABLE iCont AS INTEGER     NO-UNDO.
    DEFINE VARIABLE cMascara AS CHARACTER   NO-UNDO.
    DEFINE VARIABLE cFilho AS CHARACTER   NO-UNDO EXTENT 3.
    DEFINE VARIABLE cResto AS CHARACTER   NO-UNDO.
    DEFINE VARIABLE iAux AS INTEGER     NO-UNDO.
    DEFINE VARIABLE cGene AS CHARACTER   NO-UNDO.
    DEFINE VARIABLE iBerco AS INTEGER     NO-UNDO.

    IF deSomaRoleta < 1 THEN
        ASSIGN deSomaRoleta = 1.

    /* Seleciona os pais */
    ASSIGN deSoma = 0
        cPai = ""
        cMae = ""
        iRoleta1 = RANDOM(0,INT(deSomaRoleta))
        iRoleta2 = RANDOM(0,INT(deSomaRoleta)).
    FOR EACH tt-individuo
        WHERE NOT tt-individuo.morto
        AND tt-individuo.valido:
```

```
      ASSIGN deSoma = deSoma + tt-individuo.roleta.
      IF deSoma >= iRoleta1
      AND cPai[1] = "" THEN DO:
         ASSIGN cPai[1] = tt-individuo.genotipo[1]
                cPai[2] = tt-individuo.genotipo[2]
                cPai[3] = tt-individuo.genotipo[3].
      END.
      IF deSoma >= iRoleta2
      AND cMae[1] = "" THEN DO:
         ASSIGN cMae[1] = tt-individuo.genotipo[1]
                cMae[2] = tt-individuo.genotipo[2]
                cMae[3] = tt-individuo.genotipo[3].
      END.
      IF cPai[1] <> "" AND cMae[1] <> "" THEN
         LEAVE.
   END.

   REPEAT iBerco = 1 TO 3:

      /* Cria uma m scara dos genes que serÆo copiados do pai e monta uma string
        com os genes nÆo utilizados na sequˆncia em que ocorrem na mÆe */
      ASSIGN cResto = cMae[iBerco]
             cMascara = "".
      REPEAT iCont = 1 TO 9:
         ASSIGN cMascara = cMascara + STRING(RANDOM(0,1),"9").
         IF SUBSTRING(cMascara,iCont,1) = "1" THEN DO:
            ASSIGN OVERLAY(cFilho[iBerco],iCont,1) = SUBSTRING(cPai[iBerco],iCont,1).
            IF SUBSTRING(cPai[iBerco],iCont,1) <> "" THEN
               ASSIGN cResto = REPLACE(cResto,SUBSTRING(cPai[iBerco],iCont,1),"").
         END.
      END.

      /* Distribui os genes da mÆe fechando os buracos que ficaram no filho */
      ASSIGN iAux = 1.
      REPEAT iCont = 1 TO 9:
         IF SUBSTRING(cMascara,iCont,1) = "0" THEN DO:
            ASSIGN OVERLAY(cFilho[iBerco],iCont,1) = SUBSTRING(cResto,iAux,1)
                   iAux = iAux + 1.
         END.
      END.

      /* Verifica se haver  muta‡Æo no filho */
      REPEAT iCont = 1 TO 9:
         IF RANDOM(0,100) < PercMutacao THEN DO:
            ASSIGN iAux = RANDOM(1,9)
                   cGene = SUBSTRING(cFilho[iBerco],iCont,1)
                   OVERLAY(cFilho[iBerco],iCont,1) = SUBSTRING(cFilho[iBerco],iAux,1)
                   OVERLAY(cFilho[iBerco],iAux,1) = cGene.
         END.
      END.

   END.

   IF INDEX(cFilho[1]," ") = 0 THEN DO:
      CREATE tt-individuo.
      ASSIGN tt-individuo.genotipo[1] = TRIM(cFilho[1])
             tt-individuo.genotipo[2] = TRIM(cFilho[2])
             tt-individuo.genotipo[3] = TRIM(cFilho[3])
             tt-individuo.genotipo_total = TRIM(cFilho[1]) + TRIM(cFilho[2]) + TRIM(cFilho[3])
             tt-individuo.geracao = iGeracao
             tt-individuo.fitness = -999
             tt-individuo.valido = YES.
   END.

   RETURN "OK".


END PROCEDURE.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME

&ANALYZE-SUSPEND _UIB-CODE-BLOCK _PROCEDURE disable_UI C-Win  _DEFAULT-DISABLE
PROCEDURE disable_UI :
/*------------------------------------------------------------------------------
  Purpose:     DISABLE the User Interface
```

```
  Parameters:  <none>
  Notes:       Here we clean-up the user-interface by deleting
               dynamic widgets we have created and/or hide
               frames.  This procedure is usually called when
               we are ready to "clean-up" after running.
------------------------------------------------------------------------------*/
  /* Delete the WINDOW we created */
  IF SESSION:DISPLAY-TYPE = "GUI":U AND VALID-HANDLE(C-Win)
  THEN DELETE WIDGET C-Win.
  IF THIS-PROCEDURE:PERSISTENT THEN DELETE PROCEDURE THIS-PROCEDURE.
END PROCEDURE.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME

&ANALYZE-SUSPEND _UIB-CODE-BLOCK _PROCEDURE enable_UI C-Win  _DEFAULT-ENABLE
PROCEDURE enable_UI :
/*------------------------------------------------------------------------------
  Purpose:     ENABLE the User Interface
  Parameters:  <none>
  Notes:       Here we display/view/enable the widgets in the
               user-interface.  In addition, OPEN all queries
               associated with each FRAME and BROWSE.
               These statements here are based on the "Other
               Settings" section of the widget Property Sheets.
------------------------------------------------------------------------------*/
  RUN control_load.
  DISPLAY ElimRepet TamPopulacao NumGeracoes PercMutacao recados
          PercSobrevivencia
      WITH FRAME DEFAULT-FRAME IN WINDOW C-Win.
  ENABLE btPed btCapac ElimRepet TamPopulacao NumGeracoes PercMutacao brResult
         btVai recados PercSobrevivencia IMAGE-5 RECT-2 RECT-3
      WITH FRAME DEFAULT-FRAME IN WINDOW C-Win.
  {&OPEN-BROWSERS-IN-QUERY-DEFAULT-FRAME}
  VIEW C-Win.
END PROCEDURE.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME

&ANALYZE-SUSPEND _UIB-CODE-BLOCK _PROCEDURE inicializaPopulacao C-Win
PROCEDURE inicializaPopulacao :
/*------------------------------------------------------------------------------
  Purpose:
  Parameters:  <none>
  Notes:
------------------------------------------------------------------------------*/

    DEFINE VARIABLE iCont AS INTEGER     NO-UNDO.
    DEFINE VARIABLE iAux AS INTEGER     NO-UNDO.
    DEFINE VARIABLE cDescItens AS CHARACTER   NO-UNDO.
    DEFINE VARIABLE cAux AS CHARACTER   NO-UNDO.
    DEFINE VARIABLE iGene AS INTEGER     NO-UNDO.
    DEFINE VARIABLE iIni AS INTEGER     NO-UNDO INITIAL 1.
    DEFINE VARIABLE iGenotipo AS INTEGER     NO-UNDO.

    IF SEARCH("teste.tst") <> ? THEN DO:
       CREATE tt-individuo.
       ASSIGN tt-individuo.geracao = 999
              tt-individuo.valido = YES
              tt-individuo.fitness = -999.
       INPUT FROM VALUE(SEARCH("teste.tst")).
       IMPORT UNFORMATTED tt-individuo.genotipo_total.
       INPUT CLOSE.
       ASSIGN tt-individuo.genotipo_total = TRIM(tt-individuo.genotipo_total)
              tt-individuo.genotipo[1] = SUBSTRING(tt-individuo.genotipo_total,1,9)
              tt-individuo.genotipo[2] = SUBSTRING(tt-individuo.genotipo_total,10,9)
              tt-individuo.genotipo[3] = SUBSTRING(tt-individuo.genotipo_total,19,9).
       MESSAGE "Teste do indiv¡duo: " tt-individuo.genotipo_total
          VIEW-AS ALERT-BOX INFO BUTTONS OK.
       ASSIGN cIndividuoTeste = tt-individuo.genotipo_total.
    END.

    REPEAT iCont = 1 TO 9:
       ASSIGN cDescItens = cDescItens + CHR(iCont + 64).
    END.
```

```
   REPEAT iCont = iIni TO TamPopulacao:
      CREATE tt-individuo.
      REPEAT iGenotipo = 1 TO 3:
         ASSIGN cAux = cDescItens.
         REPEAT iAux = 1 TO 9:
            IF LENGTH(cAux) > 1 THEN DO:
               ASSIGN iGene = RANDOM(1,LENGTH(cAux))
                     tt-individuo.genotipo[iGenotipo] = tt-individuo.genotipo[iGenotipo] + SUBSTRING(cAux,iGene,1)
                     cAux = REPLACE(cAux,SUBSTRING(cAux,iGene,1),"").
            END.
            ELSE DO:
               ASSIGN tt-individuo.genotipo[iGenotipo] = tt-individuo.genotipo[iGenotipo] + cAux.
            END.
         END.
      END.
      ASSIGN tt-individuo.genotipo_total = tt-individuo.genotipo[1] + tt-individuo.genotipo[2] + tt-individuo.genotipo[3]
           tt-individuo.valido = YES
           tt-individuo.fitness = -999.
   END.

   RETURN "OK".

END PROCEDURE.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME

&ANALYZE-SUSPEND _UIB-CODE-BLOCK _PROCEDURE initialize-controls C-Win
PROCEDURE initialize-controls :
/*------------------------------------------------------------------------------
  Purpose:
  Parameters:  <none>
  Notes:
------------------------------------------------------------------------------*/
    assign chProgressBar = chctrlframe:ProgressBar.
    RUN initialize-controls-local IN THIS-PROCEDURE NO-ERROR.
    ASSIGN chProgressBar:VISIBLE = NO.

END PROCEDURE.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME

&ANALYZE-SUSPEND _UIB-CODE-BLOCK _PROCEDURE procuraUsado C-Win
PROCEDURE procuraUsado :
/*------------------------------------------------------------------------------
  Purpose:
  Parameters:  <none>
  Notes:
------------------------------------------------------------------------------*/

    DEFINE INPUT  PARAMETER cGene AS CHARACTER   NO-UNDO.
    DEFINE INPUT  PARAMETER deIni AS DECIMAL     NO-UNDO.
    DEFINE INPUT  PARAMETER deFim AS DECIMAL     NO-UNDO.
    DEFINE OUTPUT PARAMETER deRet AS DECIMAL     NO-UNDO.

    DEFINE VARIABLE iRpt1 AS INTEGER     NO-UNDO.
    DEFINE VARIABLE iRpt2 AS INTEGER     NO-UNDO.

    ASSIGN deRet = -1.

    FIND FIRST tt-usa
       WHERE tt-usa.gene = cGene
       AND   ((tt-usa.tempo-ini <= deIni AND tt-usa.tempo-fim > deIni)
       OR    (tt-usa.tempo-ini < deFim AND tt-usa.tempo-fim >= deFim)
       OR    (tt-usa.tempo-ini >= deIni AND tt-usa.tempo-fim <= deFim)) NO-ERROR.
    IF AVAIL tt-usa THEN DO:

       ASSIGN deRet = tt-usa.tempo-fim.

       blk-folga:
       REPEAT iRpt1 = 1 TO 3:
          blk-seq:
          REPEAT iRpt2 = 1 TO 3:
             /* Pega todos os usos em sequencia */
```

```
            FIND FIRST bfUsa
               WHERE  bfUsa.gene = cGene
               AND    bfUsa.tempo-ini = deRet NO-ERROR.
            IF AVAIL bfUsa THEN DO:
               ASSIGN deRet = bfUsa.tempo-fim.
            END.
            ELSE LEAVE blk-seq.
         END.
         /* Se houver espa‡o, verifica se cabe */
         FIND FIRST bfUsa
            WHERE  bfUsa.gene = cGene
            AND    bfUsa.tempo-ini > deRet NO-ERROR.
         IF AVAIL bfUsa THEN DO:
            IF (bfUsa.tempo-ini - deRet) < (deFim - deIni) THEN DO:
               ASSIGN deRet = bfUsa.tempo-fim.
            END.
            ELSE LEAVE blk-folga.
         END.
         ELSE LEAVE blk-folga.
      END.

   END.

END PROCEDURE.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME

&ANALYZE-SUSPEND _UIB-CODE-BLOCK _PROCEDURE xunxeira C-Win
PROCEDURE xunxeira :
/*------------------------------------------------------------------------------
  Purpose:
  Parameters:  <none>
  Notes:
------------------------------------------------------------------------------*/

FIND FIRST tt-pedido
   WHERE tt-pedido.termin = 2
   AND   tt-pedido.berco = 1.
ASSIGN tt-pedido.quantidade = 21545.

FIND FIRST tt-pedido
   WHERE tt-pedido.termin = 7
   AND   tt-pedido.berco = 1.
ASSIGN tt-pedido.quantidade = 8000.

FIND FIRST tt-pedido
   WHERE tt-pedido.termin = 8
   AND   tt-pedido.berco = 1.
ASSIGN tt-pedido.quantidade = 10401.

FIND FIRST tt-pedido
   WHERE tt-pedido.termin = 9
   AND   tt-pedido.berco = 1.
ASSIGN tt-pedido.quantidade = 18149.

FIND FIRST tt-pedido
   WHERE tt-pedido.termin = 3
   AND   tt-pedido.berco = 2.
ASSIGN tt-pedido.quantidade = 30000.

FIND FIRST tt-pedido
   WHERE tt-pedido.termin = 5
   AND   tt-pedido.berco = 2.
ASSIGN tt-pedido.quantidade = 4500.

FIND FIRST tt-pedido
   WHERE tt-pedido.termin = 8
   AND   tt-pedido.berco = 2.
ASSIGN tt-pedido.quantidade = 666.

FIND FIRST tt-pedido
   WHERE tt-pedido.termin = 9
   AND   tt-pedido.berco = 2.
ASSIGN tt-pedido.quantidade = 13206.
```

```
FIND FIRST tt-pedido
   WHERE tt-pedido.termin = 3
   AND   tt-pedido.berco = 3.
ASSIGN tt-pedido.quantidade = 40033.

FIND FIRST tt-pedido
   WHERE tt-pedido.termin = 5
   AND   tt-pedido.berco = 3.
ASSIGN tt-pedido.quantidade = 12000.

FIND FIRST tt-pedido
   WHERE tt-pedido.termin = 9
   AND   tt-pedido.berco = 3.
ASSIGN tt-pedido.quantidade = 2121.

END PROCEDURE.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME
```

ANEXO B: TESTE 42

```
RESULTADOS DA EXECUÇÃO DO PROGRAMA ROVINA - SCHEDULING DE PORTOS
================================================================


PARAMETRIZAÇÕES:

Tamanho da População:           50
Número de Gerações:             200
Percentual de Mutação:          10
Percentual de Sobrevivência:    40
Elimina Genótipos Repetidos:    Sim
DADOS DA EXECUÇÃO:

Hora Início:             11:06:44
Hora Fim:                11:09:48
Tempo Total de Execução: 00:03:04


POPULAÇÃO FINAL:

           Fitness Genótipo  Genótipo  Genótipo     Geração É Válido?
----------------- --------- --------- --------- ---------- ---------
            6.3485 GEHBICAFD GBDHFEAIC FAGICDEHB         89 Sim
            6.3485 HFABICDGE ICADBEHGF FAGICBEHD        116 Sim
            6.3485 HFABICEGD IBADFEHGC FAGICDEHB        105 Sim
            6.3485 GHFEBDACI BEADHFGIC BGADCEIFH        105 Sim
            6.3485 AHFBEDGCI HDGIBACFE AHIFCGDBE        105 Sim
            6.3485 GHFEBIACD IEHDFAGCB ICADEFGHB        112 Sim
            6.3485 GHFBCAEID HGAEICBFD ICFAGHEBD         21 Sim
            6.3485 HGABICEFD IBDAFEHGC FAGICDEHB         47 Sim
            6.3485 GHFBEIACD IHEDFACGB ICAHDFGEB        200 Sim
            6.3485 GHFBAEICD HEGIBACFD FABICGDHE        111 Sim
            6.3485 GHEABDCIF ICAHBGFDE FIGCBADEH        186 Sim
            6.3485 GHFBCDAEI EHAIBGFDC AGFICHDEB         47 Sim
            6.3485 GBFAHDEIC IHACBGFDE FHDICAGEB         47 Sim
            6.3485 GHFCBDAEI IEABDHFGC AIFGCHDEB        113 Sim
            6.3485 HBACIDEFG IEHGABCDF FGIDCAEHB         29 Sim
            6.3485 GHFBEDACI HEAIBCGFD AGBICHDFE         76 Sim
           22.2890 GAFBHDEIC IHACBGDFE GHDICAFEB        200 Sim
           22.2890 GHFBEAICD EGHCAIBFD BACIGEFDH        200 Sim
           22.2890 CHFBEDAGI HECIAFGBD AIGCBHDFE        200 Sim
           22.2890 IGFBDAECH EHGIABCFD AIFGCHDEB        200 Sim
           22.2890 IBAGHDEFC IEHGDCABF IGCDAFEHB        200 Sim
           22.2890 HFGEBDCAI CEDGIHAFB FGHDAIBCE        200 Sim
           22.2890 AHFBCDGEI CDBIAHGFE AGIFCHDBE        200 Sim
           22.2890 GHFAEBDCI GIEDBHAFC FABGICDEH        200 Sim
           22.2890 AHFBGEICD HCAIBEGFD AFCGHEDIB        200 Sim
           22.2890 GBIAHDEFC IEHCBGFDA FGDICAEHB        200 Sim
           22.2890 HAFEBDCIG AHDEIBCGF BFIGCEDAH        200 Sim
           22.2890 HGBAICEFD IEABFHGDC FIDAEHGCB        200 Sim
           22.2890 FAHBICGED ABDHGEFIC FAGDCIEHB        200 Sim
           22.2890 GHACEBDFI IECGABDFH BGIACDEFH        200 Sim
           22.2890 EHFBICGAD EHGBFACID FDGHCEABI        200 Sim
           22.2890 ABFHCDEIG IBACDGFEH EHAICDGFB        200 Sim
```

```
22.2890 GDFBCAHIE IBDAFEHGC FAGIHDECB        200 Sim
22.2890 GHAIBDCEF IDABFHEGC AIFGCEDHB        200 Sim
22.2890 GHFEBDCAI CHADIFGBE BGAICEDFH        200 Sim
22.2890 IHFABGEDC IAGCHEFBD IFEHCAGDB        200 Sim
22.2890 GHFICBAED IHAEGCFDB ICFDBAGHE        200 Sim
22.2890 HFEAGDCIB ICAEDHGBF FIGCHEDAB        200 Sim
22.2890 GHFABDEIC EHIABGFDC FDGICEAHB        200 Sim
22.2890 GHFCBAEID AEBDIHGFC AIEFGHDBC        200 Sim
22.2890 GHFABEICD BEGAHICFD CDAEBGIFH        200 Sim
22.2890 HFBADGECI HEAIBCGFD AHBIFCDGE        200 Sim
22.2890 IGFHBDCEA IHDFEABGC DIHFCGBEA        200 Sim
22.2890 CGFBAIDEH EIAHBGDFC AGFICHDEB        200 Sim
22.2890 AHGFCIEBD HGDAIBCFE GCIFABEDH        200 Sim
22.2890 HDFBGAECI EHGIABCFD FDGICBAEH        200 Sim
22.2890 DFAIBCHGE IHABEGDFC ACFGIBDEH        200 Sim
22.2890 GFHEBDACI AEIBCHFGD AFHDCIGEB        200 Sim
22.2890 GHEIFCBDA CAHIBFGDE GAFHCDBEI        200 Sim
22.2890 IHFAEDCBG IDBEHACGF EGIFCHDAB        200 Sim
```

ANEXO C: TESTE 47

```
RESULTADOS DA EXECUÇÃO DO PROGRAMA ROVINA - SCHEDULING DE PORTOS
================================================================


PARAMETRIZAÇÕES:

Tamanho da População:           50
Número de Gerações:             200
Percentual de Mutação:          10
Percentual de Sobrevivência:    30
Elimina Genótipos Repetidos:    Sim
DADOS DA EXECUÇÃO:

Hora Início:             10:47:07
Hora Fim:                10:47:31
Tempo Total de Execução: 00:00:24


POPULAÇÃO FINAL:

            Fitness Genótipo  Genótipo  Genótipo     Geração É Válido?
------------------ --------- --------- --------- ---------- ---------
             0.0000 DHGFEBCIA AEBIHFGDC FIABDCHGE         18 Sim
             2.5000 CEGIDBHAF AFBCHIGED FEABDCHGI         13 Sim
             2.5000 DHGCFAEIB EAHGBCFDI FIHGBAEDC          4 Sim
             2.5000 AHFEDICBG AIHBEFCDG FDEAHBGIC          5 Sim
             5.0000 CEGIABDFH GAFCBHEDI FEBDHIAGC          1 Sim
             5.0000 GDEIACBHF AEHCFIGDB IFHEABDGC          6 Sim
             5.0000 IFABDCGHE EGAIHDCBF FHGEICBAD          9 Sim
             7.5000 FBGDEAICH DEAHCGIFB HAGEFBCID          1 Sim
             7.5000 DHGECBFIA EDBGHAFIC FIHGBAEDC         14 Sim
             7.5000 DHECIABGF GIHEBAFDC ECHBGAIDF          2 Sim
             7.5000 CEGADBIHF EDIAHGFCB GAEBIDHFC         10 Sim
            10.0000 CEGIDBAFH GFBDEHACI FEBAHGDIC          3 Sim
            10.0000 FIEBCAHGD DHAICFBEG GHDEABCIF          4 Sim
            10.0000 BCEGIDHAF EFAHDICGB EABHIFDGC          6 Sim
            10.0000 BCEIGDHAF ABFCDIGHE HAEBDCFGI          6 Sim
            10.0000 HIEGBACDF AEBDIHGFC GEAIBHDFC          7 Sim
            15.0000 ECGDABIFH EDIAGHFCB GFEBAHDIC         18 Sim
            20.0000 BCIGDAHEF AEFHDICGB BAEHIFDCG         18 Sim
            32.5000 FEGDBAICH EDAHCBGFI HGACFBEDI         18 Sim
            37.5000 IAEBDCHFG EFGHBACID EFBHIGDCA         18 Sim
            40.0000 DEGBFAHCI EADCBGFHI FADGBHEIC         18 Sim
            40.0000 HEFIDGBAC EIDGAHBCF FEIHAGBCD         18 Sim
            47.5000 DEGIABCFH GEFAIHCDB FCEDHBAGI         18 Sim
            47.5000 IEDCGBAFH EGABHDCIF FHAEIGBDC         18 Sim
            47.5000 DHECIGBAF AGIEHCFDB DBCEHAGIF         18 Sim
            47.5000 HAECIBFGD HDFBECAIG HFIGBEADC         18 Sim
            50.0000 GAIDCBHEF AFEBHDCGI FAEHIBDCG         18 Sim
            50.0000 DEFCBAIHG EAIDBHFCG FDGEBIACH         18 Sim
            52.5000 HEGIABCDF EDICBHAFG GEIBHDAFC         18 Sim
            52.5000 ICABFDGHE IGEAHDCBF FEGABHICD         18 Sim
            55.0000 CEAGIDHBF EAIGDHFCB GAECIFDBH         18 Sim
            57.5000 CEBAGDIHF EDIFBGACH CAFEGDHIB         18 Sim
            57.5000 BEDAGCIHF EDIFBGACH EAGDFCHIB         18 Sim
```

```
57.5000 GEDABCIHF EIBAGFDCH BAGIEHDFC         18 Sim
60.0000 GEIHDBACF EBGAHDFCI FAEBIGHCD         18 Sim
60.0000 FDGBEHICA DEAHFGCIB HIGEFBCAD         18 Sim
60.0000 ADFEHCIBG AFHIEBCDG FHEABDCIG         18 Sim
60.0000 BHEGIACDF GIHEABFDC AHDBGIECF         18 Sim
62.5000 GEIADCBHF ECAFIBGHD IDHBAEFGC         18 Sim
62.5000 CGIEBDHAF DAGFBHICE FGCHBAEDI         18 Sim
65.0000 FIEDCHAGB DBIAEFHCG HBDEGACIF         18 Sim
65.0000 CGEIADBFH ABFCDIGHE FAEBDHCGI         18 Sim
65.0000 HADEBFCGI HBDCFEAGI FDEGHABCI         18 Sim
65.0000 EGHCIBDAF DHEAICGFB HGEFABDCI         18 Sim
65.0000 HEGADCIFB EFIAHCDBG IFEHGADBC         18 Sim
65.0000 BCEDIGHAF EDBHAFIGC EFAHIBDGC         18 Sim
65.0000 BHEDCIGAF ABECFIDHG DAEIBGFCH         18 Sim
67.5000 CBEIGDHAF IBFCDAGHE HBEADCFGI         18 Sim
67.5000 GECHBAFID AGIDFEBHC EFBIAGDHC         18 Sim
67.5000 DCEGIBAHF AFBEDGIHC FCHBGAEDI         18 Sim
```

ANEXO D: EXEMPLOS DE EMBARQUE

	Term.	22/3/2007			2/4/2007			4/4/2007			7/4/2007			29/4/2007		
A	SV	**212**	**213**	**214**	**212**	**213**	**214**	**212**	**213**	**214**	**212**	**213**	**214**	**212**	**213**	**214**
B	SH				2.780							5.950				
C	CB	21.545						8.000		10.000	5.635					
D	CG		40.033	30.000	3.000		20.426							7.000		41.500
E	CL				8.541	12.000			3.000			3.000	16.800		4.000	
F	CO		12.000	4.500	12.520	21.200	7.690		15.500			14.720		8.000	5.100	5.000
G	CS				2.000	8.600						2.852			1.600	
H	CT	8.000			11.714	18.200			44.500	5.000		28.478			3.300	
I	PA	10.401		666	7.800			41.790			49.709	5.000			1.000	
	Total	18.149	2.121	13.206			28.784	4.000		25.000				6.000		11.500
		58.095	**54.154**	**48.372**	**48.355**	**60.000**	**56.900**	**53.790**	**63.000**	**40.000**	**55.344**	**60.000**	**16.800**	**21.000**	**15.000**	**58.000**

		1/5/2007			22/5/2007			25/5/2007			28/5/2007			30/5/2007		
	Term.	**212**	**213**	**214**	**212**	**213**	**214**	**212**	**213**	**214**	**212**	**213**	**214**	**212**	**213**	**214**
A	SV								5.800	3.000			5.500			
B	SH		7.850				3.100					7.103		22.872		
C	CB			57.750	53.138			48.137			27.300		3.000			
D	CG	2.000					9.944			18.763			649			4.847
E	CL	25.097								26.369			27.660		10.500	30.000
F	CO	17.683								2.726			3.136			
G	CS	6.256										5.000	18.691	8.000		13.000
H	CT	3.000	12.469			14.215	48.349		14.215	4.000				4.578		4.000
I	PA		14.000		7.363			12.362		2.273		4.009		7.243	11.000	11.000
	Total	**54.036**	**34.319**	**57.750**	**60.501**	**14.215**	**61.393**	**60.499**	**20.015**	**57.131**	**27.300**	**16.112**	**58.636**	**42.693**	**21.500**	**62.847**

		17/8/2007			26/8/2007			28/8/2007			16/9/2007			27/9/2007		
	Term.	**212**	**213**	**214**	**212**	**213**	**214**	**212**	**213**	**214**	**212**	**213**	**214**	**212**	**213**	**214**
A	SV			22.540			30.702			19.781						41.175
B	SH	17.700			8.000						26.620			13.730		
C	CB		15.236		9.300	11.340		8.100				22.300	10.000		25.200	3.123
D	CG			14.502			10.123		16.000	18.574	17.783		1.000			
E	CL		14.764	19.441	12.700		7.523			2.000		11.000	5.145		10.000	6.955
F	CO					40.740					2.000			1.940		
G	CS	1.300			19.600			4.000				4.000	24.855	9.500	4.000	
H	CT	10.000	5.000		8.000			34.125			9.000		2.000	6.039		940
I	PA			5.000			10.485			17.646	7.968		25.516	1.694		8.364
	Total	**29.000**	**35.000**	**61.483**	**57.600**	**52.080**	**58.833**	**46.225**	**16.000**	**58.001**	**63.371**	**37.300**	**68.516**	**32.903**	**39.200**	**60.557**

		1/10/2007			24/10/2007			26/10/2007		
	Term.	**212**	**213**	**214**	**212**	**213**	**214**	**212**	**213**	**214**
A	SV									38.462
B	SH				1.954					
C	CB	13.000	17.750	19.817			13.540			
D	CG									6.157
E	CL		18.000	4.957	7.583		7.395			5.000
F	CO			18.612				51.345		
G	CS	8.950	3.000	8.713	11.997					
H	CT	40.072		6.000						
I	PA				3.959		5.983			6.100
	Total	**62.022**	**38.750**	**58.099**	**25.493**	**0**	**26.918**	**51.345**	**0**	**55.719**