JAIRO HUMBERTO CABRERA TOVAR

MODELO DE UM SISTEMA DE PROGRAMAÇÃO DINÂMICA DE TURNOS BASEADO NA TEORIA DOS CONJUNTOS NEBULOSOS

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Engenharia.

São Paulo

2000

JAIRO HUMBERTO CABRERA TOVAR

MODELO DE UM SISTEMA DE PROGRAMAÇÃO DINÂMICA DE TURNOS BASEADO NA TEORIA DOS CONJUNTOS NEBULOSOS

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Engenharia.

Área de Concentração:
Engenharia Naval e Oceânica

Orientador:
Prof. Dr. Oscar Brito Augusto

2000

"La mayoría de las ideas fundamentales de la ciencia son esencialmente sencillas y por regla general pueden ser expresadas en un lenguaje comprensible para todos."

Albert Einstein

"A computação com palavras representa uma clivagem na tradição de ter mais respeito pelos números do que pelas palavras. O modelo de referência da computação por palavras é a mente humana."

Lotfi Zadeh
(criador da "Lógica Nebulosa")

Dedico este trabalho aos meus pais,
Luis Humberto e **Lucía**,
e às minhas irmãs,
Martha e **Sandra**.
Por tudo o que eles representam para mim.

AGRADECIMENTOS

Gostaria de registrar meus agradecimentos às pessoas que contribuíram para que eu concluísse este trabalho e realizasse os cursos no Departamento de Engenharia Naval e Oceânica da EPUSP.

Inicialmente agradeço ao Prof. Dr. Oscar Brito Augusto, meu orientador, pela atenção e orientação desta pesquisa, mas especialmente por ter me mostrado o auxílio e apoio nos momentos de dificuldade. Registro meu reconhecimento e profundo respeito pelo seu exemplo de dedicação acadêmica.

Da mesma forma, não poderia esquecer meu amigo e Prof. Dr. Edínson Aedo, cujas sábias ponderações e cujo incentivo nas horas certas muito contribuíram desde o início até quase a consecução desta obra.

Ao Prof. Dr. Cheng Liang Yee e ao Prof. Dr. Marco Túlio Carvalho, por manterem sua porta sempre aberta e pelos comentários e críticas no percurso deste trabalho.

Desejo manifestar minha especial gratidão aos professores do Depto de Engenharia Naval e Oceânica, Prof. Dr. Célio Taniguchi, Prof. Dr. Marco Antônio Brinatti e Prof. Dr. Rui Carlos Botter pela atenção, sugestões e comentários valiosos no desenvolvimento desta pesquisa.

Agradecimentos sinceros aos meus colegas de pós-graduação do departamento. Desejo manifestar minha especial gratidão à pesquisadora e amiga Giuliana Bonatelli Dario pela constante ajuda no mais que pudesse, pela troca de idéias e contribuições ao longo deste trabalho.

Agradeço à Lânia, secretaria da seção de pós-graduação do Depto de Engenharia Naval e Oceânica, pela sua eficiência e constante apoio nos tramites necessários junto a este departamento. Da mesma forma, agradeço a Sandra, Idamaris, Neuza, César e Adenílson, funcionários do Departamento pelos serviços prestados durante todo esse tempo.

Ao Departamento de Engenharia Naval e Oceânica da Escola Politécnica da Universidade de São Paulo, pela confiança em mim depositada em desenvolver um trabalho de doutorado e por proporcionar a infra-estrutura necessária para conseguir tal propósito.

Ao Brasil e à Universidade de São Paulo, meu apreço por ter-me permitido essa oportunidade. Foi dignificante estar vinculado à USP durante alguns anos, ela é um exemplo do que se pode fazer com grandeza neste maravilhoso país. Foi muito bom mesmo ter tido, com a graça de *Deus*, essa oportunidade em minha vida.

À CAPES – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - pelo apoio financeiro.

A Anderson pela sua acessória com o *LINUX & C++*; e a Cristiano pela revisão do português do texto.

Aos meus sobrinhos Diana, Daniel e Andrés, e demais familiares que, mesmo na distância, sempre estiveram presentes em todos estes momentos longe de casa.

À Marina Yessenia pelo constante estímulo e apoio durante a realização deste trabalho.

A todos que por ventura tenha esquecido de mencionar que direta ou indiretamente me auxiliaram ou inspiraram o desenvolvimento da pesquisa e que permitiram chegar a esta tese e outros trabalhos.

E agradeço a *Deus* e a *Nosso Senhor Jesus Cristo* que me deram forças para realizar esta tarefa e me iluminaram o caminho.

ERRATA

Jairo Humberto Cabrera Tovar
Departamento de Engenharia Naval
Data da Defesa: 11 de Agosto de 2000

- Na página 37, onde lê-se na equação (3.10):

$$S_M(\mu_A(x),b)=\max(\mu_A(x),\mu_B(x))$$

Leia-se:

$$S_M(\mu_A(x),\mu_B(x))=\max(\mu_A(x),\mu_B(x))$$

- Na página 51, onde lê-se na tabela 4.1 S_i^*; leia-se S_j^*.

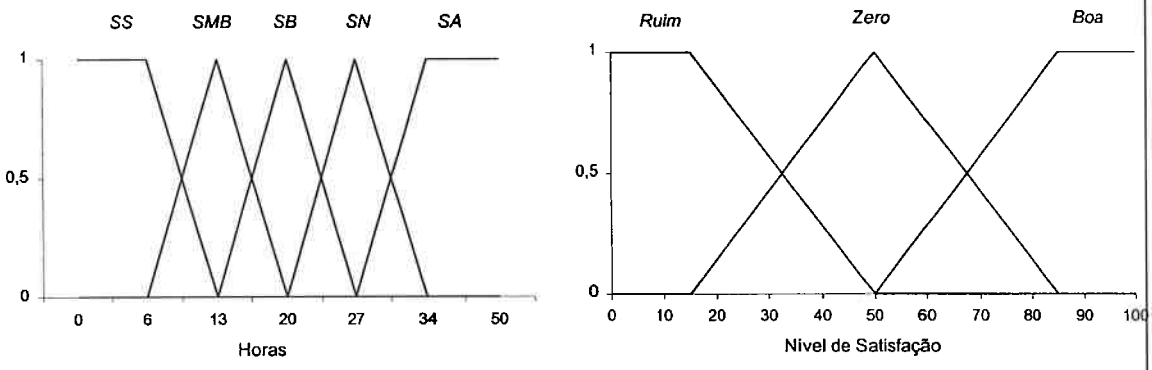
- Na página 54, 1ª Etapa, onde lê-se:

[...] correspondente à demanda de turnos para cada dia da semana e o número de trabalhadores, calculado pelo algoritmo 2.

Leia-se:

[...] correspondente à demanda de turnos para cada dia da semana e o número de trabalhadores, calculado pelo algoritmo 1.

- Na página 67, a representação qualitativa da Restrição *SeqüênciaFolgas* (Figura 4.6) corrigida é a seguinte:



onde SS = *sem_satisfação*
SMB = *satisfação_muito_baixa*
SB = *satisfação_baixa*
SN = *satisfação_normal*
SA = *satisfação_alta*

Figura 4.16 - Representação Qualitativa da Restrição *SeqüênciaFolgas*

- Na página 86, onde lê-se:

Na seqüência, será calculado o número inicial de trabalhadores W', equação (4.19) e a mão de obra inativa t, equação (4.10), como:

Leia-se:

Na seqüência, será calculado o número inicial de trabalhadores W', equação (4.9) e a mão de obra inativa t, equação (4.10), como:

- Na página 123, onde lê-se na equação (A4):

$$\mu_A(x) = 1 - \mu_A(x), \ \forall x$$

Leia-se:

$$\mu_A(x) = 1 - \mu_B(x), \ \forall x$$

- Na página 128, as Funções de Pertinências dos Termos da Variável Lingüística *Idade* (Figura A5) corrigida é a seguinte:

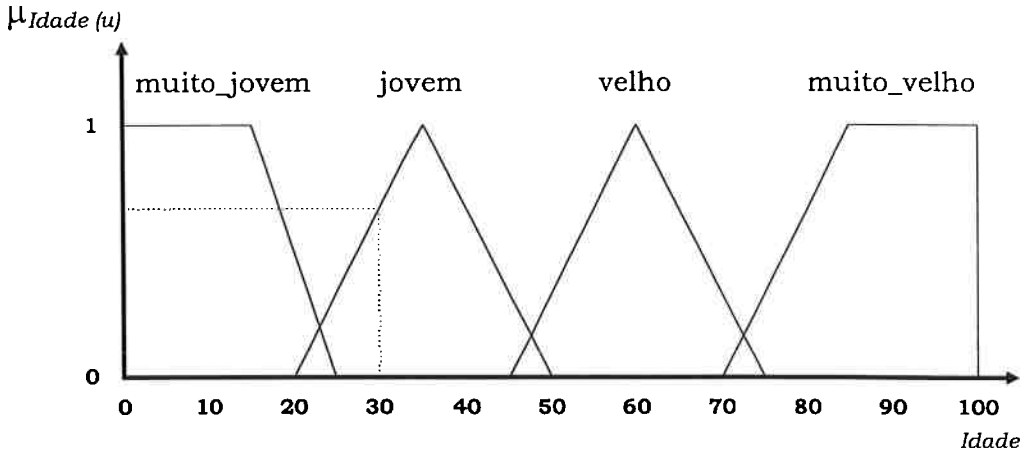


Figura A5 – Funções de Pertinência dos Termos da Variável Lingüística *Idade*

- Na página 132, onde lê-se na equação (A19):

$$R^{(i)} : Se\ x_1\ é \ldots x_k\ é\ E_i^k\ Então\ y_1 = f_1(x_1 \ldots x_k)\ e \ldots\ y_m = (x_1 \ldots x_k)$$

Leia-se:

$$R^{(i)} : Se\ x_1\ é \ldots x_k\ é\ E_i^k\ Então\ y_1 = f_1(x_1 \ldots x_k)\ e \ldots\ y_m = f_m(x_1 \ldots x_k)$$

SUMÁRIO

Capítulo 5

Capítulo 6

Anexo A

Anexo B

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS E ABREVIATURAS

B	Número de grupos de turnos trabalhados consecutivamente em dias diferentes;
c_j	Custo semanal do padrão de dias de folga j;
$d_{[i,(j,j+1)]}$	Diferença de horas trabalhadas por um empregado i na semana j e $j+1$;
D	Conjunto finito de valores para cada variável em V (domínio de um PSR);
G	Regra sintática para gerar os nomes das etiquetas lingüísticas em uma variável lingüística;
J	Conjunto de turnos J = {1, 2, 3, ... , p};
K	Conjunto de categorias ou empregados K = {1, 2, 3, ... , m};
L	Extensão máxima permitida do período de trabalho em dias;
m	Índice correspondente a cada categoria ou empregado;
M	Número de empregados ou de categorias [i];
N	Número de semanas [j] ;
O_{ACOtan}	Operador médio compensatório associativo ;
O_{fuzzy_and}	Operador médio *fuzzy and* ;
O_{fuzzy_or}	Operador médio *fuzzy or* ;
O_{comp_and}	Operador médio *compensatory and*;
Q	Número de pares de turnos consecutivos em dias diferentes;
p	Índice correspondente a cada tipo de turno;
P	Esquema de pesos em um PSRN;
RS	Regra semântica que associa o seu significado a cada etiqueta lingüística;
S	Função *t-conorma* ;
S_i	Alcance da restrição, lista ordenada de variáveis k_i ;
S_j	Conjunto de 3 números não adjacentes, j = 1, ..., 7;
S_j^*	Complemento S_j^*;
$Sat.g$	Função de avaliação global ;
$Sat.p$	Função de avaliação parcial ;
t	Mão de obra inativa;
T	Função *t-norma*;
T_D, S_D	Operadores produto e soma drásticos;
T_P, S_P	Operadores probabilísticos ou produto algébrico;
T_M	Operadores T de Zadeh;
T_G S_G	Operadores *Gupta*;
T_L S_L	Operadores *Lukasiewicz*;
T_H , S_H	Operadores *Hamacher*;
T_W S_W	Operadores *Weber*;
$T(x)$	Conjunto dos termos lingüísticos x de uma variável lingüística;
U	Universo de discurso de uma variável lingüística;
V	Conjunto finito de variáveis em um PSR;
W	Número mínimo do nível de escalação;
τ	Extensão do turno;
γ	Parâmetro do grau de proximidade entre "and" e "or";
$\mu_A(x)$	Função de pertinência de um conjunto nebuloso A;
$\Re_i$	Relação da restrição sobre D de dimensão k_i ;

$\Re_{crisp}$	Restrição *crisp*;
$\Re_{soft}$	Restrição *soft*;
$\Re_i(S_i)$	Par representativo de uma restrição;
$\Re\mu$	Conjunto de restrições nebulosas;
$\odot$	Operador de composição no MPG;
$\Pi(x)$	Distribuição possibilística de x;
AD	Método de Agregação primeiro e Defuzzificação depois;
CG	Centro de Gravidade;
DA	Método de Defuzzificação primeiro e Agregação depois;
MN	Modelo Nebuloso;
MNFD	Modelo Nebuloso com Fuzzificador e Defuzzificador;
MNP	Modelo Nebuloso Puro;
MPG	*Modus Ponens* Generalizado;
MPSRN	Modelo de Programação de Turnos baseado no Problema de Satisfação por Restrições Nebulosas;
PD	Padrão de Demanda;
PSR	Problema de Satisfação de Restrições;
PSRN	Problema de Satisfação por Restrições Nebulosas;
RCI	Regra Composicional de Inferência;
SIPRODT	Sistema de Programação Dinâmica de Turnos;
TD	Turno de Tarde;
TFS4	Turno de Fim-de-Semana de 4 horas;
TFS12	Turno de Fim-de-Semana de 12 horas;
TM	Turno de Manhã;
TN	Turno de Noite;
VMM	Defuzzificador baseado no Valor Médio dos Máximos.

RESUMO

Métodos matemático-analíticos como os usados na abordagem da pesquisa operacional são freqüentemente insuficientes para a modelagem de problemas combinatórios em diferentes áreas de aplicação, incluindo programação, planejamento, projeto, configuração, etc. Isto de deve principalmente a três razões: incertezas no processo estudado, complexidade combinatória do espaço de busca e objetivos conflitantes no processo de otimização. Esta tese apresenta uma metodologia baseada na teoria dos conjuntos nebulosos com o intuito de resolver estes problemas. Técnicas a base de conhecimento, especialmente "raciocínio aproximado" e "relaxamento", próprias desta teoria, cobrem as incertezas inerentes ao problema, além de suportarem o relaxamento dinâmico das restrições em conflito (restrições antagônicas), que, combinadas com determinadas heurísticas, podem chegar a soluções chamadas de sub-ótimas; a pesar disto, estas soluções são aceitáveis pelo fato de ser, as vezes, impossível encontrar uma melhor solução para um dado problema.

Esta metodologia serviu de base para o projeto de um sistema de programação dinâmica de turnos que permite encontrar soluções, em um tempo relativamente reduzido, para um problema de ordem combinatória como é a programação de turnos direcionada a um processo de estiva em um terminal marítimo de contêineres. O sistema calcula inicialmente o número mínimo de trabalhadores/categorias (nível de escalação) necessários para uma definida demanda e, através de um processo iterativo, baseado em métodos por reparos, que pressupõe a geração de uma solução inicial e seu "reparo", fundamentado este último em um problema de satisfação por restrições nebulosas, PSRN, chega-se finalmente à programação de turnos do recurso humano. A requisição de um número mínimo de trabalhadores necessários e o exercício de sua alocação nos turnos são tarefas de alta complexidade devido, entre outras coisas, à quantidade de restrições que deve ser considerada. Estas são questões das mais críticas na determinação dos níveis de produtividade em um terminal. Provisões legais e regras internas da administração do terminal fornecem diferentes restrições em relação a atividades portuárias, as quais devem ser satisfeitas quando os trabalhadores são designados para os turnos de trabalho (número permitido de horas semanais de trabalho, extensão mínima de ciclos contínuos de trabalho, folgas entre turnos, etc.). O sistema proporciona a flexibilidade necessária a fim de atingir a variabilidade freqüente e imprevisível própria da demanda de prestação de serviços portuários assim como de acontecimentos eventuais dos empregados, tais como faltas, licenças, saídas, etc.

Também é analisado o processo de otimização iterativa por reparos, método que trata das restrições violadas como passos de reparos especificamente definidos para a aplicação em estudo. Estes passos de reparos são usados pelo algoritmo de otimização como instruções para modificações a fim de encontrar melhores soluções. Efetuam-se comparações de desempenho entre duas técnicas de reparos, uma de natureza determinística, reparo em profundidade, e outra estocástica, reparo por modificações aleatórias.

Devido ao estágio atual do processo de modernização dos portos nacionais, em que a organização do trabalho nos terminais passa por uma profunda reestruturação, o que dificulta a apreensão da realidade corrente devido à sua complexidade, propõe-se aqui um processo de estiva ideal em um terminal de contêineres, que servirá também de base para ilustrar as técnicas usadas neste trabalho.

ABSTRACT

Mathematical-analytical methods as used in operational research approach are often insufficient for modeling combinatorial problems in different areas of application, including scheduling, planning, design, configuring and alike. This is due to three reasons: uncertainties in the studied process, combinatorial complexity of the search space, and conflicting objectives in the optimization process. This dissertation presents a methodology based on the theory of fuzzy sets with the intention to solve combinatorial problems. Knowledge-based techniques, especially "approximate reasoning" and "constraint relaxation", proper of this theory, cover the inherent uncertainties to the problem, besides supporting the dynamic relaxation of conflict constraints (antagonistic constraints), which, combined with determined heuristics, can achieve the so-called sub-optimal solutions. In spite of that, these solutions are acceptable sometimes for the fact of being impossible to find a better solution for the given problem.

This methodology served as base for the design a system of dynamic programming with shifts that allows finding the solutions, in a relatively reduced time, for a problem of combinatorial order as it is in the shift scheduling directed to the container stevedoring process in a maritime terminal. Initially, the system calculates the minimum number of workers/categories (manning scales) that is necessary for a defined demand and, through an iterative process, based on methods of repairs, that estimate the generation of an initial solution and its "repair", which is based on a problem of satisfaction for fuzzy constraints, leads, finally, to the shift scheduling of the human resources. The requirement of a minimum number of necessary workers and the exercise of its allocation in shifts are tasks of high complexity due to, among other things, the amount of constraints that must be considered. These are questions of the most critical in the determination of the levels of productivity in a terminal. Legal provisions and internal rules of the terminal management supply different constraint in relation to the port activities, which must be satisfied when the workers are assigned for the shift work (allowed number of weekly hours of work, minimum extension of continuous cycles of work, time off between shifts, etc.) The system provides the necessary flexibility in order to reach the frequent and unexpected variability that is intrinsic to the demand of port service as well as of occasional events of the employees, such as absences, leaves, sick-leaves, etc.

It is also analyzed the process of iterative optimization for repairs, which is a method that deals with the violated constraints as steps of specifically defined repairs for the application being studied. These repairs-steps are used by the algorithm of optimization as instructions for modifications in order to find better solutions. Comparison of performance is made between two techniques of repairs, one of deterministic nature, repair deeping, and another random one, random repair.

Due to the current process of modernization of the national ports, in which the organization of the work in the terminals is being completely restructured and, therefore, makes it difficult to apprehend the current reality, what is proposed in this work is an ideal container stevedoring process, that it will also serve as base to illustrate the techniques used in this dissertation.

Capítulo 1

INTRODUÇÃO

1.1 Descrição do Problema

O trabalho em regime de turnos nos terminais marítimos de contêineres é uma atividade que vem surgindo em resposta à crescente demanda dos clientes, ou seja, do aumento da movimentação de carga, o que leva à necessidade de trabalho ininterrupto, dia e noite, até completar as tarefas de carregamento e descarregamento. Esta demanda reflete-se substancialmente nos custos diários da permanência dos navios no berço. Um sistema de programação de lista de turnos *(Rostering Systems)* é um mecanismo que fornece mão de obra, quando requerido, para operar terminais de contêineres sobre uma base contínua, isto é, 24 horas por dia, sete dias por semana. Este sistema deve proporcionar a flexibilidade suficiente para atingir uma variabilidade freqüente e imprevisibilidade da demanda pela prestação de serviços portuários, típicos dos portos latino-americanos.

A requisição de um número mínimo de trabalhadores necessários e o exercício de sua alocação nos turnos são tarefas de alta complexidade devido, entre outras coisas, à quantidade de restrições que deve ser considerada. Estas são questões das mais críticas na determinação dos níveis de produtividade num terminal.

Para cada período cíclico, o responsável pelo terminal de contêineres elabora uma programação sobre o período e o turno para os quais os trabalhadores devem ser designados para as operações diárias, incluindo também a possibilidade de horas extras. Esta programação deverá indicar, inicialmente, o número total de empregados a serem alocados, permitindo determinar a utilização e os custos totais dessa mão de obra. Tal resultado dependerá, em grande parte, da escolha feita pelo responsável entre diferentes alternativas de programações. Provisões legais e regras internas da administração do terminal fornecem diferentes restrições em relação a atividades portuárias, as quais devem ser satisfeitas quando os trabalhadores são designados para os turnos de trabalho (número permitido de horas semanais de trabalho, extensão mínima de ciclos contínuos de trabalho, folgas entre turnos, etc.).

Esta é uma enorme e complexa tarefa para os programadores humanos. O esforço pode levar um enorme tempo para produzir manualmente uma programação deste tipo, devido não só ao grande número de restrições, mas também aos cálculos das estatísticas de programações anteriores, as quais devem ser consideradas. Restrições conflituosas com alto grau de incerteza fazem parte deste tipo de aplicação (preferências individuais dos operadores, regulamentos legais e operacionais, etc.)

Além disso, depois de estabelecida a programação produzida manualmente, esta não pode ser facilmente modificada e ajustada. Desafortunadamente, o trabalho no terminal de contêineres não pode ser prognosticado para um longo período de tempo. De fato, é difícil predizer as operações com um mês de antecedência. A programação deverá ser continuamente aprimorada e ajustada para atender a natureza do trabalho do

terminal como mudanças fortuitas na demanda dos serviços e de acontecimentos eventuais dos empregados, tais como faltas, licença por doenças, saídas, etc.

A determinação do número mínimo de empregados, a elaboração de programações de turnos e o gerenciamento de listas de tarefas são, portanto, críticas em um terminal de contêineres que pretenda manter-se competitivo e expandir seus negócios.

1.2 Motivação

O autor não encontrou na literatura, entre o vasto número de pesquisas publicadas, nenhum trabalho acerca do problema da alocação do número necessário de trabalhadores e da programação de turnos em um terminal marítimo de contêineres. Grande número destas aplicações encontra-se focalizada especialmente na programação de tarefas de enfermeiras em hospitais (MILLER:1976), (WARNER:1976), (SMITH:1977), (LAZARO:1995), (MEYER:1998), empresas de serviços de comidas rápidas, vendas no varejo, bancos (BLOSSOM:1993) e programadores de sala de aula.

Mas, sem dúvida, a maior parte das pesquisas na área de programações focaliza-se em problemas de nível industrial, conhecidos como *job-scheduling*. Trata-se da alocação de máquinas, ferramentas, pedidos, etc., deixando-se de lado, de alguma forma, o fator humano, um dos mais críticos, senão o mais custoso nas operações de qualquer sistema. Programações deste tipo têm sido intensamente estudadas há muito tempo.

A motivação da escolha do tema para o presente trabalho provém de dois caminhos:

1) do preenchimento de um vazio em relação às diferentes aplicações que têm sido dadas ao problema de programações de mão de obra; e
2) da observação de grandes dificuldades nesta área quando são aplicados métodos convencionais de otimização como, por exemplo, os métodos *simplex, backtracking* e *branch&bound* (DORN:1994). Isto me levou a tratar da questão como um problema de satisfação parcial por restrições nebulosas e a propor um sistema de alocação dinâmica de turnos para a elaboração destas programações.

De acordo com (BERRY:1993) e (KERR:1994), um alto grau de incerteza e imprecisões pode ser atribuído a muitos requisitos encontrados nos problemas de programação, o que os torna apropriados para serem trabalhados junto à teoria dos conjuntos nebulosos. A imprecisão origina-se a partir das definições nebulosas das restrições dos sistemas. No caso da programação de tarefas, estas definições imprecisas encontram-se nas datas de entrega e de vencimento dos prazos, nas restrições especificando o tempo aproximado decorrido entre operações sucessivas, etc.; no caso das programações de turnos, pode-se encontrá-las na definição da duração dos turnos, no tempo aproximado entre os turnos, etc. Tais restrições são freqüentemente expressas por programadores humanos usando variáveis lingüísticas, definíveis operacionalmente, ao se mapear a faixa preferível dos parâmetros sobre os valores correspondentes das funções de pertinência dos conjuntos nebulosos, representando essas restrições. O mapeamento pode representar as preferências subjetivas do programador humano, ou

pode ser derivada a partir de um sistema do gerenciamento hierárquico de restrições que reflete as preferências correntes entre possíveis objetivos conflitivos da estratégia organizacional.

Espera-se com este trabalho desenvolver uma metodologia favorável que permita encontrar uma solução o mais próxima possível da ótima para um problema de ordem combinatória (como é o exercício da programação), direcionando-o para uma aplicação real de uma programação de turnos para o processo de estiva em um terminal marítimo de contêineres. Dentro dessa nova metodologia, surge a necessidade de uma ferramenta flexível que permita a definição das variáveis e das restrições nebulosas do problema em estudo, assim como dos apropriados operadores de agregação. Ferramenta que deverá ser superior quando comparada aos métodos clássicos de otimização, os quais mostram, freqüentemente, fraqueza de representação. Pretende-se também analisar o processo de otimização iterativa por reparos, método que trata as restrições violadas como passos de reparo especificamente definidos para dada aplicação, que, segundo (DORN:1994), parecem ser mais promissores do que as abordagens clássicas já amplamente utilizadas.

1.3 Definição dos Termos Básicos

A seguir, serão fornecidas importantes definições com a finalidade de uma maior compreensão dos leitores. Concentra-se nos termos-chave que serão usados em todo o trabalho. Para facilitar o entendimento das definições, serão utilizadas algumas ilustrações:

Estivador

É o empregado encarregado do processo de estiva dos contêineres, que inclui gerenciamento, supervisão e serviços operacionais. Para maior clareza, no anexo B, há maiores descrições sobre este assunto.

Horizonte de Tempo

O *horizonte de tempo* do processo de programação de turnos consiste normalmente de três níveis:

- nível diário: é definida a posição e estrutura dos turnos.
- nível semanal: fixa-se a força de trabalho nominal por dia no período de uma semana.
- nível anual: é feito um balanço entre a força de trabalho nominal e a atual. A diferença entre estas duas forças de trabalho pode ser causada por vários motivos, dentre os quais, picos nos níveis de serviço, cursos e treinamentos, férias, licenças por doenças, etc. Esta diferença pode ser compensada principalmente com tempos extras ou com pessoal de reserva (incluindo uma categoria na programação dos turnos).

Programação de Turnos

Uma programação de turnos define quais empregados têm de trabalhar, sobre que turno e em que dia específico. Os requisitos das operações são as diretrizes para a elaboração da lista de tarefas. Se é conhecido o número de empregados requeridos sobre quais dias da semana e quantas vezes ao dia, é possível designar um turno apropriado e uma categoria nesse tempo. Por exemplo, na segunda-feira, dois grupos do turno da manhã deverão estar disponíveis de modo a assegurar um nível suficiente de operação.

A tabela 1.1 é um exemplo de uma programação de turnos com uma extensão de 50 dias com 5 categorias. Ela designa os serviços em conjuntos de 6 turnos e 4 dias de folga usando um padrão rotativo.

Tabela 1.1 – Programação de Turnos para uma Extensão de 50 dias
(TM = Turno de Manhã, TD = Turno da Tarde, TN = Turno da Noite)

Dia	Segunda 1	Terça 2	Quarta 3	Quinta 4	Sexta 5	Sábado 6	Domingo 7	Segunda 8	Terça 9	Quarta 10
Categoria 1	TM	TM	TD	TD	TN	TN				
Categoria 2			TM	TM	TD	TD	TN	TN		
Categoria 3					TM	TM	TD	TD	TN	TN
Categoria 4	TN	TN					TM	TM	TD	TD
Categoria 5	TD	TD	TN	TN					TM	TM

Geralmente, esperam-se programações de turnos de natureza mais complexa que a apresentada acima. A descrição das seqüências dos turnos não pode ser tão simples como um ciclo rotativo do exemplo acima. Além disso, o aumento do número de categorias introduz um grau adicional de complexidade. Dependendo da natureza do problema em questão, está sendo comum e algumas vezes necessário definir programações de turnos de longas durações.

Os sistemas de programações de turnos são cruciais para o desempenho das operações de carregamento e descarregamento nos terminais de contêineres. Diferentes tipos de turnos, seja de 4, 8 ou 12 horas são usados pelos estivadores para atingir uma demanda de natureza variável. No processo de alocação dos recursos humanos, estivadores enfrentam um grande número de restrições, e reduzir parcialmente o impacto destas facilita o projeto de extensas programações de turnos. O trabalho por turnos em um terminal de contêineres é a resposta em relação ao tempo da demanda dos clientes até que a tarefa de carregamento e descarregamento do navio tenha sido completada. Esta demanda se reflete nos custos diários associados à permanência do navio no berço.

Nível de Operários

O nível de operários refere-se ao total de trabalhadores empregados em um terminal. Isto varia de terminal para terminal e geralmente inclui os empregados permanentes e empregados suplementários ou avulsos.

Nível de Escalação

O nível de escalação especifica o número de trabalhadores ou categorias requeridos por terno para realizar tarefas definidas de um processo específico em um

terminal, tais como: operação de equipamento pesado (por exemplo, dois operadores por porteiner), operação de veículos ou desempenho de outras tarefas como apeação da carga etc.

Processo de Estiva

Define-se como atividade de carregamento e descarregamento de mercadoria de um navio. O processo de estiva para um terminal típico de contêineres é descrito na seção 1.5

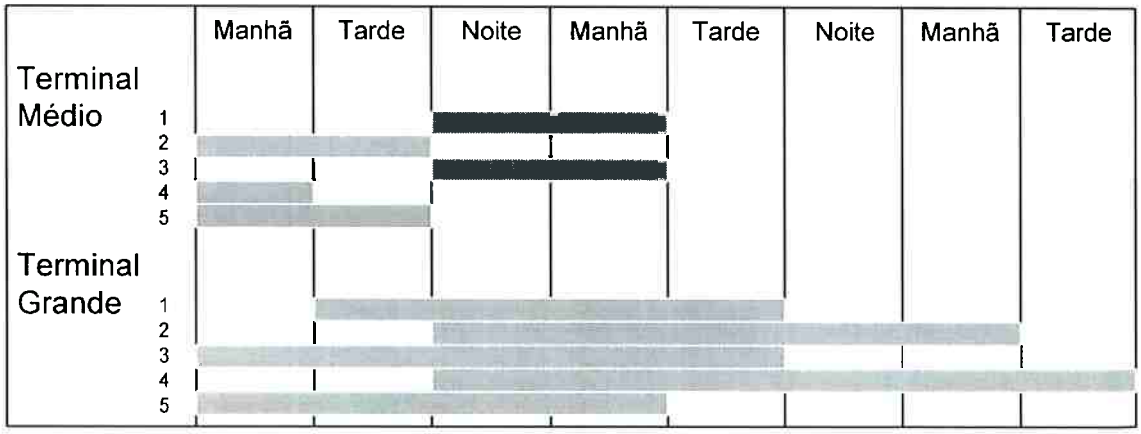
Ternos

É muito comum, nos terminais marítimos assim como nas plantas de produção, agregar os empregados em unidades funcionais e/ou organizacionais. Estas unidades são chamadas de ternos. Estes apresentam certa hierarquia de agrupamento chamada de grupos ou categorias. No anexo B, são apresentados maiores detalhes sobre os arranjos típicos de mão de obra em terminais de contêineres, bem como o número de empregados necessários para formar um respectivo grupo e/ou terno e sua escalação.

Turno

Um turno é um certo intervalo de tempo definido em um ou vários dias específicos da semana em que é requerida uma dada operação, ou seja, ele define uma certa designação de serviços, por exemplo, um turno diurno de 8 a.m. a 3 p.m. É possível também ter diferentes turnos para diferentes dias da semana. Por exemplo, muitos problemas de programação de turnos requerem turnos particulares em uma semana. Quando se toma uma decisão sobre a estrutura de um turno, temos que considerar vários aspectos: ciclo do tempo de operação, interrupções, reciclagem, etc. Hoje em dia, e considerando sua movimentação, a maioria dos terminais marítimos de contêineres operam 7 dias por semana, e 24 horas ao dia e geralmente em 3 ou 4 turnos diários de aproximadamente 8 e 6 horas, respectivamente. A tabela 1.2 mostra um exemplo do regime de trabalho por turnos, para cada navio, em um terminal de médio e grande porte, as áreas sombreadas indicam quantos turnos trabalharam por navio.

Tabela 1.2 – Exemplo de Regime de Trabalho por Turnos e por Navio, em um Terminal de Médio e Grande Porte



Em resumo, o sistema de programação de turnos é um mecanismo que fornece mão de obra, quando requerida, para operar terminais de contêineres sobre uma base contínua, isto é, 24 horas por dia, sete dias por semana. Este sistema precisa fornecer flexibilidade suficiente para atingir uma carga de trabalho freqüente e imprevisível. A tarefa do estivador é atingir uma carga variável de trabalho com disponibilidade de mão de obra e equipamento.

Contudo, para fornecer uma quantidade apropriada de mão de obra sob certas condições, defronta-se com várias restrições no processo de alocação dos empregados para os diferentes turnos.

1.4 Objetivos de uma Programação de Turnos

A elaboração da programação de turnos é um exercício de ordem administrativa em qualquer empresa, seja de produção de mercadorias ou de serviço. A mão de obra é um recurso limitado e indispensável que deverá ser administrado efetivamente, de modo que uma firma, seja qual for sua natureza, atinja seus objetivos. Uma programação efetiva é o principal componente do gerenciamento do trabalho, na medida em que exerce um impacto forte sobre a rentabilidade, sobre o serviço dos clientes e sobre a moral dos empregados. As decisões a serem tomadas, em uma específica programação de mão de obra, deverão ser consideradas dependendo do meio-ambiente da empresa e do recurso humano. Este trabalho focaliza a programação de recursos humanos em um terminal marítimo de contêineres.

Incontestavelmente, a minimização de custos é um objetivo no sistema terminal marítimo, mas seus objetivos principais podem ser freqüentemente diferentes devido à interação direta com os clientes. Os objetivos para a programação de turnos em um terminal de contêineres podem ser resumidos como segue:

- Minimizar o tempo de resposta do início do atendimento a navios;
- Minimizar o tempo médio de espera dos navios;
- Minimizar o tempo médio de atendimento a navios;
- Minimizar o número médio de empregados;
- Melhorar e manter a moral dos empregados através de boas programações.

1.5 Descrição de Processo de Estiva de Contêineres

A estiva dos contêineres é um processo mecanizado que envolve o uso de equipamentos especializados tal como guindastes e veículos transportadores de contêineres em terminais dedicados ao manuseio de contêineres. A infra-estrutura inclui pátios para estoque de curta duração e sistemas de computação para planejar e processar o deslocamento de contêineres através do terminal (figura 1.1)

Apesar da mecanização das tarefas de carregamento-descarregamento e movimentação, há, ainda, tarefas manuais, as quais devem ser realizadas pelos estivadores.

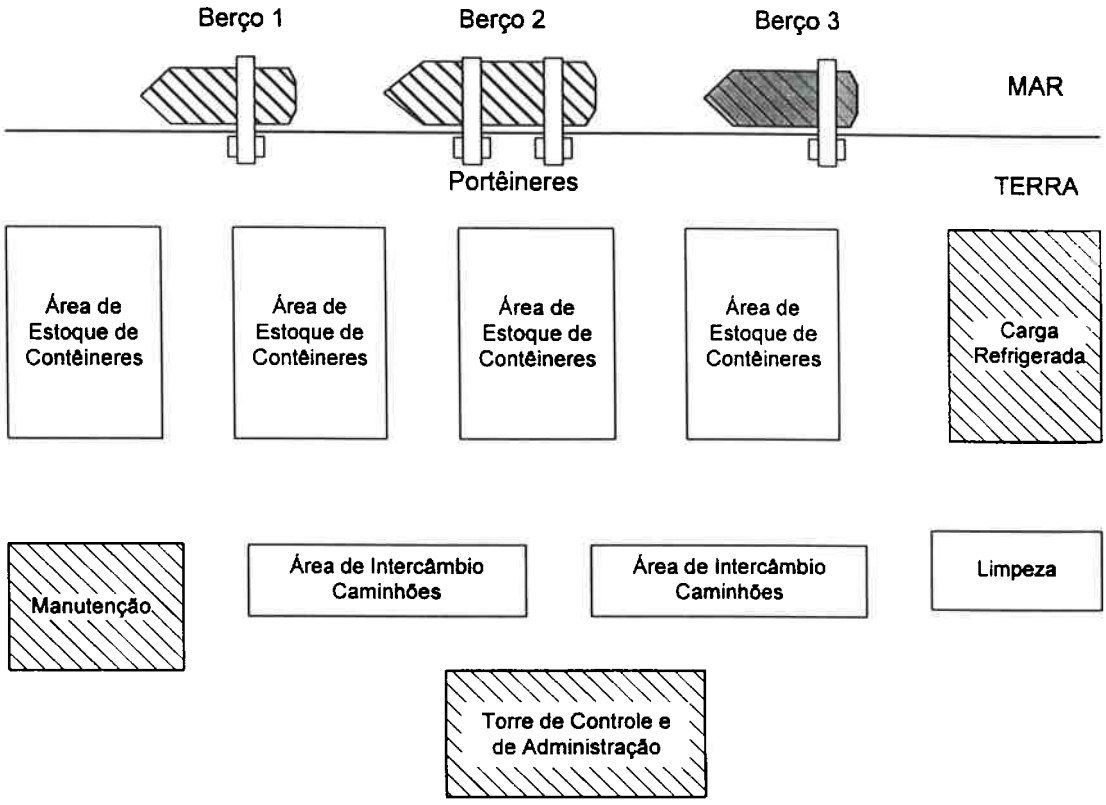


Figura 1.1 –*Lay-out* ideal de um Terminal de Contêineres

Há principalmente duas técnicas usadas para o carregamento e descarregamento de contêineres. A primeira trata de navios especializados no transporte de contêineres que dependem de guindastes situados no terminal (portêineres); este processo é usado para movimentar a maioria dos contêineres que chegam aos portos. A segunda trata de navios *roll-on, roll-off* (*ro-ro*), os quais são carregados e descarregados por equipamentos especializados e levados a bordo do navio usando uma rampa. Geralmente, navios *ro-ro* não são especializados no transporte de contêineres, levando outros tipos de carga fragmentada.

Diferentes tipos de equipamento são usados em um terminal ideal de contêineres. A escolha está parcialmente relacionada com as características das operações, tais como a área do pátio e a movimentação. Apesar de todos os tipos de equipamentos usados em um terminal particular, um conteiner se desloca através de uma série similar de etapas nas operações de carregamento e descarregamento como se mostra na figura 1.2

Durante o transporte da terra para o navio, um conteiner é geralmente manuseado separadamente pelo menos em duas ocasiões. Para carga de exportação, a primeira etapa envolve um movimento entre a rodovia ou ferrovia e as pilhas no pátio. Contêineres são geralmente estocados em dois e três de altura, em um intervalo de tempo curto antes de serem transferidos para o navio. Dependendo do terminal, vários tipos de equipamento, incluindo *straddle carries,* transtêineres e veículos de transferência interna, tais como caminhões e trailers, são usados nesta etapa. Empilhadeiras não são geralmente usadas nos terminais para deslocar contêineres para ou das pilhas. Sistemas de planejamento são projetados para minimizar a re-estivagem

da carga, mas contêineres podem ser movidos entre as pilhas antes de serem deslocados para os portêineres.

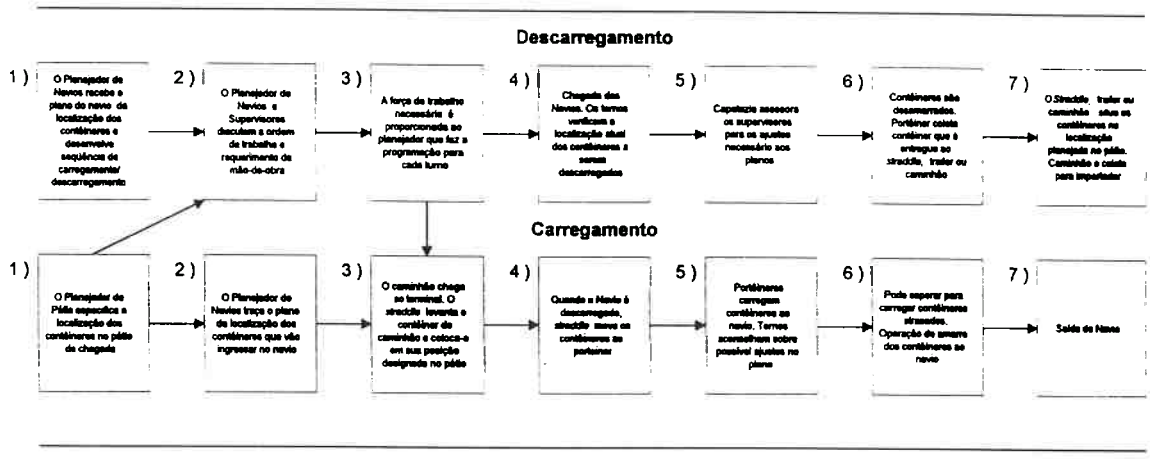


Figura 1.2 – Explicação simplificada de como os Navios Porta-Contêineres são Descarregados e Carregados, incluindo Recebimento e Entrega

Equipamento similar é envolvido na próxima etapa de movimento de contêineres entre as pilhas e o porteiner localizado próximo ao navio. O porteiner então coloca o conteiner dentro do lugar designado no navio. Sistemas de planejamento computadorizado procuram minimizar o número de movimentos por contêineres desde quando este é carregado no navio até seu descarregamento em seu destino final. Contudo, quando um navio leva contêineres para diferentes destinos, um conteiner pode ser deslocado várias vezes se isto é necessário para acessar os contêineres embaixo daqueles estivados para o mesmo porto, ou para outros em rota do destino final do conteiner.

O processo de contêineres de importação é inverso. Alguns contêineres requerem diferentes técnicas de manuseio. Há requisitos especiais, por exemplo, para carga perigosa (tais como químicos) que afeta a estiva de contêineres a bordo do navio e nos pátios dos terminais.

A introdução de novas tecnologias, particularmente, conteinerização e computadorização, tem requerido considerável investimento em novos equipamentos e processos. A produtividade ressaltada pela tecnologia tem conduzido a mudanças nos padrões da organização do trabalho e reduzido significativamente a requisição de mão de obra. Da mesma maneira, o uso do planejamento assistido por computador, cada vez maior, para as operações de carregamento e descarregamento de navio, tem mudado o papel e reduzido o número de empregados de escritório.

1.6 Contribuição desta Tese

- Propor um novo método para resolver o problema de alocação de recursos humanos em terminais marítimos. As aplicações encontradas em outros campos, desenvolvidas no contexto da pesquisa operacional, limitam-se a métodos matemático–analíticos, os quais são considerados teoricamente confiáveis. De um modo geral, estes métodos chegam a fornecer soluções ótimas, mas possuem

limitações no campo da modelagem de aplicações da vida prática; no caso da programação de turnos, uma aplicação real, dificulta a representação ou formulação da modelagem de suas restrições.

- Fornecer um procedimento novo para a modelagem e otimização de problemas de programações de turnos utilizando a teoria dos conjuntos nebulosos. Em termos gerais e como já foi dito, os métodos matemático–analíticos são freqüentemente insuficientes para aplicações da vida real. Isto se deve a três razões: 1) a informações imprecisas; 2) à complexidade combinatória do espaço de busca 3) aos objetivos conflitantes entre si na otimização. A combinação de várias técnicas, com base no conhecimento, é a forma mais promissora para tratar estes problemas. O procedimento proposto poderá ser utilizado também como um "solucionador" geral de problemas de natureza combinatórias.

- Delinear um procedimento com base no conhecimento do sistema, que permita representar explicitamente todo tipo de restrições de problemas de programações de turnos, mesmo que algumas delas sejam vagamente conhecidas, o que permitirá ainda modelagem para um vasto número de aplicações.

- Prover um método que resolva, a partir de uma nova combinação entre restrições formuladas por meio da teoria dos conjuntos nebulosos e heurísticas com base em reparos, o problema da programação de turnos para atender um processo de estiva de um terminal de contêineres. A formulação das restrições por números nebulosos auxilia a modelagem de problemas de programações de uma forma melhor que os modelos prévios (tais como *simplex, backtracking* e *branch&bound*), devido ao fato de que eles permitem uma melhor representação das restrições através do uso de variáveis lingüísticas. Heurísticas com base em reparos, tem apresentado uma eficiência melhor para resolver grandes problemas, típicos de programação, quando comparados com os algoritmos construtivos e enumerativos.

- Resolver o problema de programações de turnos por meio de um sistema de programação dinâmica de turnos, o qual permita aprimorar e ajustar as mudanças nas demandas de serviços e de acontecimentos fortuitos relacionados à mão de obra.

1.7 Organização da Tese

O ponto central desta tese é a relação entre o problema de satisfação de restrições e a teoria dos conjuntos nebulosos, com o objetivo de aplicá-los ao problema de programação de turnos no processo de estiva em um terminal de contêineres. A lista a seguir contém uma breve descrição de como estão organizados os capítulos.

- O capítulo introdutório expõe o problema e o motivo da escolha do tema, além de introduzir algumas definições importantes, incluindo a descrição do funcionamento de um processo ideal de estiva de contêineres.

- O capítulo 2 discute os diferentes tratamentos que se tem dado, na literatura especializada, ao problema de programações de turnos. Por ser se tratar de problema

combinatório, existe um universo de técnicas para sua solução, desde as tradicionais ferramentas previstas pela pesquisa operacional até as mais recentes abordagens, conseqüências de pesquisas nas áreas de inteligência artificial junto à de pesquisa operacional.

- O capítulo 3 trata da formulação do problema propriamente dito. Neste capítulo, será explicado, na teoria e em exemplos, como é possível modelar problemas de programações de turnos por meio de restrições nebulosas. Para isso, será feito inicialmente uma breve introdução dos métodos clássicos de satisfação de restrições PSR's e de satisfação parcial de restrições PSPR's, abordando também algumas técnicas de solução; dando uma ênfase especial às heurísticas baseadas nos métodos de reparos.

- No capítulo 4, propõe-se um sistema para a programação dinâmica de turnos no processo de estiva em um terminal de contêineres (SIPRODT), o qual, além de elaborar a programação de turnos com um número mínimo e suficiente de empregados – considerando as restrições pertinentes do sistema –, permita realizar os ajustes necessários na programação conforme as mudanças nas flutuações da demanda de serviço.

- No capítulo 5, apresentam-se aplicações de SIPRODT através de dois procedimentos; inicialmente será demonstrado passo-a-passo como deve ser elaborado um novo projeto de programação de turnos, gera-se assim uma base de dados que será utilizada no segundo procedimento, no exercício da programação dinâmica de turnos, mostrando-se várias aplicações com diferentes exemplos de demanda.

- O capítulo 6 apresenta as conclusões e as recomendações para futuros trabalhos.

- O anexo A descreve uma introdução à teoria dos conjuntos nebulosos que inclui definições importantes que serão usadas no decorrer de toda esta tese.

- No anexo B, serão apresentadas algumas generalidades em relação à classificação, tarefas e responsabilidades da mão de obra no processo de estiva em um terminal marítimo de contêineres ideal.

Capítulo 2

REVISÃO BIBLIOGRÁFICA

2.1 Generalidades

Muitas aplicações da vida real tais como planejamento e programação (problemas conhecidos na literatura com a palavra inglesa *"scheduling"*) podem ser classificadas como problemas combinatórios de otimização, caracterizados por:

- espaço de busca que cresce exponencialmente com o tamanho do problema;
- restrições de alta complexidade;
- número alto de possíveis soluções, entre as quais, a necessidade de se escolher a melhor, seguindo algum critério de otimização.

Por muito tempo, foi usada somente a abordagem da pesquisa operacional para resolver problemas combinatórios de otimização. Esta abordagem é baseada na representação matemática dos problemas, os quais são tipicamente modelados por meio da programação linear inteira (NEMHAUSER:1988). Recentes pesquisas nas áreas de Inteligência Artificial e Pesquisa Operacional têm indicado interessantes revisões às tradicionais abordagens vistas até alguns anos atrás.

Dentro deste contexto, pode-se dividir a pesquisa sobre programação de turnos em aplicações gerais como um problema de otimização combinatório em dois grandes campos de atuação. Inicialmente, utilizando as inúmeras ferramentas previstas na tradicional pesquisa operacional; e utilizando, mais recentemente, as novas abordagens, produto de pesquisas nas áreas de inteligência artificial junto à pesquisa operacional. A seguir, apresentar-se-á um resumo dos principais trabalhos em cada uma destas áreas.

Inicialmente, é conveniente introduzir notação simples para representar diferentes classes de problemas de programação de turnos para recursos humanos, tratados na literatura. A representação $\tau/p/m$/PD/L será usada para este fim, onde:

τ = Duração ou extensão do turno, por exemplo 4, 8 ou 10 horas;
p = |J|, sendo J = {1, 2, 3, ... , p}: o conjunto de turnos;
m = |K|, sendo K = {1, 2, 3, ... , m}: o conjunto de categorias ou de empregados;
PD = Padrão de demanda que pode variar dependendo do problema em tratamento;
L = Extensão máxima permitida do período de trabalho de dias consecutivos.

As origens da formulação matemática de problemas de programação de turnos podem ser atribuídas a Tibrewala et al. (TIBREWALA:1972), os quais forneceram um simples algoritmo iterativo para determinar a força de trabalho necessária sob as condições de que cada empregado trabalhe cinco dias da semana e tenha dois dias consecutivos de folga. Baker (BAKER:1974) descreveu um algoritmo de duas etapas

para a solução do mesmo problema. A vantagem deste algoritmo em relação ao primeiro fundamenta-se em que o tamanho do problema não é susceptível ao esforço computacional requerido para a determinação de uma alocação ótima. Assumindo relaxar os dias consecutivos de folga de modo que os empregados possam ter estes dois dias em qualquer momento, é desejável minimizar o número de empregados que são designados em dias de folga não consecutivos, neste contexto, Rothstein (ROTHSTEIN:1972) propôs um modelo de programação linear.

Miller et al. (MILLER:1976) formularam a programação de turnos para enfermeiras como um problema geral de programação matemática. Neste problema, foram usados dois tipos de restrições. A primeira consiste na viabilidade de um conjunto de restrições tais como o trabalho de 10 dias num período de programação de 14 dias e fronteiras máximas e mínimas na duração do período de trabalho; e, a segunda, consiste em restrições independentes cujas violações incorrem em custos que penalizam a função objetivo. Um exemplo deste tipo de restrição é que nenhum padrão deve conter divisões nos finais de semana. O objetivo é minimizar a soma dos custos penalizados pelas violações destas restrições e pelo número mínimo de requisição de pessoal.

Warner (WARNER:1976) atribuiu a decisão da programação a um problema de programação de múltiplos critérios, cuja função objetivo quantifica as preferências de cada empregado em relação à duração do período de trabalho e à necessidade de dias de folga. A demanda para cada categoria de empregados e para cada turno são impostas como restrições.

Smith et al. (SMITH:1977) desenvolveram um procedimento heurístico, o qual considera um conjunto de restrições e preferências individuais de várias categorias de pessoal. Em vez de examinar o problema como uma alternativa à programação numérica para cada categoria e selecionar a melhor combinação a partir de um conjunto de alternativas, são geradas programações individuais dinamicamente de semana em semana. As preferências individuais são acomodadas dentro dos limites das atuais restrições.

Barthholdi et al. (BARTHOLDI:1980) sugeriram dois modelos. O primeiro modelo se refere a uma rede de fluxos, o segundo, a uma programação linear com procedimento especial de finalização para atingir uma solução ótima. Estes procedimentos de solução são apropriados para uma classe restrita de problemas e não podem manusear restrições relativas à freqüência de folgas durante os finais de semana.

Com o uso de métodos combinatórios, tem-se chegado a soluções ótimas para alguns problemas de programações de turnos simples e categorias com diferentes combinações de restrições impostas pela administração e sindicatos. Brownell et al. (BROWNELL:1976) analisaram uma situação em que a requisição dos empregados para os dias de semana era maior ou igual aos de final de semana. Os empregados tinham dois dias consecutivos de folga e, de cada dois finais de semana, um de descanso. Lowerre (LOWERRE:1977) estendeu este trabalho a outros dois contextos: cada empregado tem dois dias de folga, incluindo, pelos menos de cada três finais de semana, um de descanso; e cada empregado recebe todos os outros finais de semana de folga, em um total de quatro dias livres em duas semanas e um período de trabalho de pelo menos cinco dias. Baker e Magazine (BAKER:1977) estenderam ainda estes resultados, relaxando a condição em que a demanda para empregados durante os dias de semana seja maior ou igual aos de finais de semana, reconhecendo explicitamente o período máximo de trabalho como um critério secundário e permitindo a exclusão prévia de alguns padrões de trabalho. Burns (BURNS:1978) generalizou para uma situação de uma demanda variável, em que cada empregado trabalha 10 dias em 14; tem

livre pelo menos cada segundo final de semana e um período de semana de pelo menos 6 dias. Baker et al. (BAKER:1979) forneceram uma abordagem modular para resolver a situação de demanda uniforme, na qual cada empregado tem o direito a, pelo menos, uma fração de folga no final de semana e a permanecer entre os limites máximos e mínimos do período de trabalho.

Burns et al. (BURNS:1985) conduziram uma situação ainda mais generalizada, em que as requisições dos empregados variam para cada dia da semana. Cada empregado teria exatamente 2 dias de folga numa semana, incluindo uma fração A livre de cada final de semana B; e, cada empregado deveria ter uma duração do ciclo de trabalho máximo de 6 dias, (8/1/1/PD[1]/6). É importante aqui diferenciar os algoritmos de programação do tipo iterativos dos algoritmos de programação cíclicos. A abordagem de programação cíclica estabelece primeiro uma programação praticável para um período inicial (por exemplo, 3 ou 4 semanas) e, logo então, este padrão é repetido período após período. Na abordagem iterativa, a programação prossegue da semana i à semana $i+1$. Desta maneira, a abordagem iterativa supera a inflexibilidade dos algoritmos de programação cíclicos. Embora os problemas como turnos simples e categorias simples sejam de aplicação limitada, eles servem como blocos projetados nos algoritmos, como um meio de encontrar soluções ótimas para programações de múltiplos turnos e com hierarquia na força de trabalho.

Burns et al. (BURNS:1987) forneceram uma solução modular para soluções ótimas nos problemas de programação da força de trabalho caracterizados por turnos múltiplos e categorias simples de empregados (8/p/1/PD[1]/6). A demanda dos empregados durante os dias de semana é maior ou igual aos de final de semana. Cada empregado requer uma média de dois dias de folga por semana, incluindo pelo menos uma fração A livre dos finais de semana B, uma extensão do ciclo de trabalho máximo de 6 dias e pelo menos um dia livre entre dois diferentes turnos. Na abordagem modular, o algoritmo constrói uma programação, combinando mini-programações chamadas de módulos. O programador pode usar um conjunto de módulos pré-construídos ou pode construir um conjunto maior de módulos que forneçam uma maior flexibilidade.

Emmons et al. (EMMONS:1991) discutiram um algoritmo cíclico do tipo (8/1/m/PD[2]/6) para problemas de turnos simples e hierarquia na força de trabalho. Os empregados formam uma hierarquia de níveis de especialização, baseada na capacidade de cada empregado realizar sua tarefa e de substituí-lo, caso a função seja de menor especialização, mas não vice-versa. Existe uma requisição uniforme de empregados n_k para cada categoria k (k = 1, 2, 3,..., m) por dia, 7 dias da semana. As regras de trabalho pregam que cada empregado tenha 2 dias livres por semana, incluindo exatamente uma fração A de folga por B finais de semana; que cada trabalhador tenha uma duração do ciclo de trabalho de no máximo 6 dias. No entanto, o algoritmo não é ótimo no sentido de que, no pior caso, o tamanho da força de trabalho desvia-se do ótimo por uma margem de 3%. As programações não são completamente praticáveis devido ao fato de que o limite da duração do ciclo excede 6 dias.

Programações de turnos simples e força de trabalho com hierarquias são relativamente recentes. Hung (HUNG:1994) apresentou um algoritmo ótimo para a situação de uma demanda variável durante a semana sob certas condições, nas quais se

[1] Padrão de demanda no qual as demandas dos dias da semana e final de semana para os empregados são fixas e a primeira é maior ou igual à segunda.

[2] Padrão de demanda em que o número de empregados requeridos permanece o mesmo para todos os dias da semana.

considera a hierarquia da força de trabalho em que um empregado mais qualificado pode substituir outro de qualificação menor, mas não vice-versa. Esta estrutura de trabalho existe em muitas situações práticas, como é o caso nos quadros de enfermeiras e dos militares. Billionnet (BILLIONNET:1999) utilizou uma abordagem de programação inteira para resolver o mesmo problema apresentado por (HUNG:1994).

Beaumont (BEAUMONT:1997) descreveu a solução de um problema de programação de recursos humanos para atingir uma demanda variável, cujos objetivos giraram em torno da determinação do número de empregados e os instantes de tempo em que os turnos devem começar.

Hesham K. (HESHAM:1998) desenvolveu um procedimento em duas etapas na solução de problemas de programações cíclicas de recursos humanos, considerando dois dias consecutivos de folga na semana. Inicialmente, é desenvolvida uma expressão simples, que, logo então, é usada para calcular o número mínimo da força de trabalho necessária. Este mínimo é logo incluído como uma restrição no modelo de programação linear do problema, cuja solução produzirá os valores inteiros ótimos. Este novo procedimento resolve o caso em que o custo dos diferentes padrões semanais de trabalho não sejam necessariamente iguais; isto resultou sendo computacionalmente mais eficiente do que os outros dois métodos disponíveis, desenvolvidos por (BARTHOLDI:1980)

Outra linha de pesquisa focalizou-se na variável da duração dos turnos. Poor (POOR:1972) descreveu a vantagem e o contexto para o uso dos turnos de 10 horas. Hung (HUNG:1991) forneceu um algoritmo para problemas de categorias simples e turnos de 10 horas, mas ignorando a restrição do período de trabalho. Ele discutiu ainda programações no contexto de 3 e 4 dias de semana, no qual, em qualquer duas semanas consecutivas, cada empregado deve trabalhar 3 dias numa semana e 4 dias na outra.

Em virtude de muitos problemas da vida real tais como *time-tabling, scheduling, planning, etc.* poderem ser formulados como Problemas de Satisfação de Restrições (*Constraint Satisfaction Problems*) PSR, diversas pesquisas têm desenvolvido eficientes métodos para tratar PSR's, incluindo técnicas especializadas (PIRLOT:1996) ou gerais (NONOBE:1998). Descrições de aplicações práticas podem ser encontradas em (FREUDER:1982), (DECHER:1988), (FREUDER:1990), (MINTON:1992) (TSANG:1993) e (RODOSÊK:1997).

Os problemas de satisfação de restrições, estudados principalmente no campo da inteligência artificial, podem ser informalmente definidos como problemas que consistem em um conjunto de variáveis, cada uma delas associadas a um conjunto finito de valores, e um conjunto de restrições sob estas variáveis. O objetivo é encontrar os valores das variáveis que não viole as restrições.

O sucesso destes métodos se deve a diversos fatores, especialmente a suas inovadoras técnicas de solução. Um destes fatores trata das restrições, que podem ser representadas de uma maneira natural e compacta em vez das técnicas de solução alternativas como por exemplo, programação linear, em que as restrições têm de ser igualdades ou desigualdades lineares. Outra razão refere-se diretamente a alguns mecanismos de otimização que são produto da própria natureza, como no caso de *Simulated Annealing* (BLOSSOM:1993) e (PIRLOT:1996) e de Algoritmos Genéticos (DORN:1994), (PIRLOT:1996), cuja aplicabilidade geral das suas abordagens, flexibilidade para levar em consideração restrições específicas em casos reais, boa relação entre qualidade na solução, facilidade de implementação e tempo de computação, fazem cada vez maior o seu uso. Muitos destes procedimentos vêm de princípios de busca gerais, guiados por estratégias específicas, como é o caso da busca

tabu (GLOVER:1989), (GLOVER:1990), (PIRLOT:1996), (NONOBE:1998) (BOYCE:1995), (GENDREAU:1998), (BARBAROSOGLU:1999) e dos algoritmos de busca de profundidade iterativos (vide seção 3.4.2), estas últimas técnicas heurísticas são descritas também como métodos ou meta-algoritmos (PIRLOT:1996).

A programação com base em restrições tem-se convertido recentemente numa abordagem popular que combina técnicas de pesquisa operacional e inteligência artificial para o tratamento de problemas de gerenciamento de recursos humanos e lista de tarefas, conhecidos na literatura como de *Crew Rostering Problem* (CRP) e *job scheduling* respectivamente. O incremento nesta linha de pesquisa é devido ao fato de que estes problemas apresentam, de um lado, uma formulação natural com base em restrições e, de outro, a disponibilidade de programas comerciais aceitáveis que suportam os conjuntos de variáveis e de restrições (CHIP:1994) (ILOG:1996). Segue abaixo algumas linhas de pesquisas sobre o problema de programação de turnos encontradas na literatura e formuladas por meio de PSR, conhecidos também como problemas de programações com base nas restrições.

Meyer et al. (MEYER:1998) introduziram certas adaptações do algoritmo *Branch&Bound* com *Forward Checking* e *Backmarking,* métodos construtivos, para aplicar técnicas de PSR a PSPR (Satisfação Parcial de Restrições) na representação do problema de programação de turnos de enfermeiras. Meyer usou um modelo de 6 a 9 turnos em vez de 3 turnos, como na maioria dos hospitais, e considerou requisitos de diferentes importâncias, criando deste modo uma ordem hierárquica dentro das restrições.

Seguindo a mesma linha de pesquisa, Hoong Lau et al. (LAU:1997) tentaram outra abordagem com base nas restrições, formulando-a também por meio de um PSR, mas resolvendo um CRP de múltiplas hierarquias, usando um conjunto de três restrições básicas, permitindo o manuseio de preferências entre estas. A designação dos trabalhadores em turnos é governada pelas seguintes restrições: cada trabalhador não deverá trabalhar mais do que um número determinado de dias consecutivos (antes de um dia de folga), cada trabalhador deverá ter dois dias de descanso por semana e um empregado, trabalhando no período noturno de um dia j, não deverá ser designado no turno da manhã do dia $j+1$, de modo que consiga descansar o suficiente entre os turnos. Neste trabalho, demonstra-se teoricamente que o espaço de busca é reduzido por um fator exponencial comparado com as abordagens convencionais.

Como se tem demonstrado, muitos problemas podem ser representados como um conjunto de restrições sob um conjunto de variáveis, por exemplo, o problema de satisfação de restrições. Desafortunadamente, problemas da vida real tendem a ser inconsistentes e, portanto, os problemas correspondentes de satisfação de restrições não têm solução. Uma maneira de driblar esta inconsistência é usando a teoria dos conjuntos nebulosos. A idéia é associar valores nebulosos aos elementos das restrições e combinar estes valores de uma modo razoável que corresponda diretamente à forma como são manuseados os problemas com base nas restrições *crisp*[3].

Dorn et al. (DORN:1992) aplicaram o conceito de "relaxamento"[4] baseados na teoria dos conjuntos nebulosos ao domínio na programação de um processo de produção, em que as tarefas impõem restrições de diferentes graus de importância sobre as precedentes e as seguintes. Os valores das variáveis lingüísticas são usados para

[3] Também chamadas de booleanas ou restrições *hard*. São restrições que podem ser ou não satisfeitas, sem estados intermediários; mais detalhes nos capítulos 3 e 4.

[4] O *relaxamento* de uma restrição modifica as relações definidas por uma restrição. O termo *relaxamento* dá a entender que permite modificações em uma ampla gama de relações.

designar a importância das tarefas individuais e a compatibilidade de cada par de tarefas. A atenção da programação enfoca inicialmente as tarefas mais importantes. Algumas destas podem não ser programadas sem violar restrições e, algumas outras, adjacentes, terão uma baixa compatibilidade. Encontra-se inicialmente uma solução que deve ser melhorada, e as tarefas associadas às restrições são removidas ou trocadas por outras tarefas, permitindo assim uma melhoria. Uma função de avaliação é definida baseada nos valores nebulosos das restrições e a busca é guiada por esta função de avaliação.

2.2 Análise da Revisão Bibliográfica

Os problemas de programação têm sido estudados isoladamente há muitos anos, porém, em virtude de pesquisas recentes nas áreas de Inteligência Artificial e Pesquisa Operacional, têm surgido revisões interessantes em detrimento das abordagens tradicionais vistas há alguns anos. Além disso, estes problemas estão sendo tratados em um contexto mais geral, no qual o trabalho já começa a fechar o ciclo usando sistemas de programação no campo operacional. Contudo, como foi visto na revisão da literatura, a programação de turnos é um problema complexo. Um pesquisador ou praticante tem de se acomodar a múltiplos objetivos, múltiplas restrições e um sem número de variáveis. A principal fonte de dificuldade na construção de boas programações é produto da natureza conflitante de seus objetivos.

Assim como na maioria das decisões na vida real, quando se resolve construir uma nova programação, não é sempre possível satisfazer perfeitamente todos os objetivos. Isto se aplica a todos os diferentes tipos de programações, tanto em processos industriais quanto em empresas prestadoras de serviços, desde programadores de salas de aula nas escolas e universidades até programadores de alocações de recursos humanos.

As pesquisas prévias têm demonstrado que houve um crescente desenvolvimento de heurísticas específicas para cada aplicação, sem gerar, no entanto, um método único que pudesse ser aplicado a todo tipo de programação. Nesse sentido, esta tese apresenta um procedimento geral destinado ao projeto de programações para múltiplas aplicações.

2.3 Oportunidades para Pesquisa : Aplicação de Novos Algoritmos baseados na Teoria dos Conjuntos Nebulosos

Em situações da vida real, resulta ser mais realista quando se consideram os compromissos entre os objetivos. Para muitos problemas, faz sentido satisfazer parcialmente os objetivos. O grau de satisfação pode ser usado para avaliar o alcance destes compromissos. Além disso, objetivos reais são freqüentemente priorizados, portanto é necessário ponderar esta satisfação com fatores de importância. Uma maneira simples e especial de atingir estes dois aspectos - satisfazer as restrições a um certo grau e levar em consideração importâncias relativas dos problemas de programação - é a modelagem destas restrições através da teoria dos conjuntos nebulosos.

Embora o tratamento por restrições nebulosas seja tão poderoso quanto os outros modelos de satisfação parcial de restrições, ele é particularmente adaptável a

modelagens, pelo fato de as restrições poderem ser escritas em um formato facilmente compreensível pelo especialista humano, uma vez que eles apresentam um comportamento robusto que quase não precisa de afinação para se obter um controle razoável. Além disso, a avaliação do grau de satisfação pode ser muito eficiente quando é usada para orientar métodos heurísticos de busca a fim de encontrar *boas* soluções, enquanto, ao mesmo tempo, conseguem-se grandes simplificações dentro da complexidade típica dos problemas de programação.

Descrições do mundo real contêm formulações vagas sobre as relações entre os critérios devido ao desconhecimento de muitos detalhes, dificultando a obtenção de melhores resultados; isto poderia levar, na modelagem, à desconsideração dos dados incertos. Tal situação não ocorreria fazendo-se uso da teoria dos conjuntos nebulosos; neste caso, estes valores de dados incertos poderão ser melhor modelados por meio de distribuições possibilísticas, que são interpretações especiais que fazem parte desta teoria.

Pretende-se com este estudo elaborar um procedimento para o projeto de programações dinâmicas de turnos para um terminal de contêineres; para tanto usar-se-á, inicialmente, a modelagem das restrições do sistema através da teoria dos conjuntos nebulosos, baseando-se no uso de técnicas heurísticas de reparo para sua solução.

Capítulo 3

FORMULAÇÃO DO PROBLEMA

3.1 Introdução

Problemas de otimização combinatória do mundo real, tais como a programação de turnos, são complexos de resolver com métodos exatos. Além do mais, possuem restrições vagamente especificadas e de diferentes importâncias, sem falar na disponibilidade de dados que pode ser também incerta e ter relações antagônicas entre os seus objetivos. Para grande parte destes problemas, faz sentido satisfazer parcialmente os objetivos. O grau de satisfação pode ser usado para avaliar o alcance desses compromissos. Adicionalmente, objetivos reais são priorizados, portanto é necessário ponderar suas satisfações com fatores de importância. Uma maneira especial e simples de atingir estes dois aspectos dos problemas de programações, satisfazer as restrições a um certo grau e levar em consideração importâncias relativas, é modelar estas restrições através da teoria dos conjuntos nebulosos, problemas conhecidos na literatura como *Problema de Satisfação por Restrições Nebulosas* - PSRN (*Fuzzy Constraints Satisfaction Problem*) .

Neste capítulo será explicado, na teoria e com alguns exemplos, como se poderia modelar problemas de programações de turnos por meio de restrições nebulosas. Para isso, far-se-á inicialmente uma breve introdução dos métodos clássicos de satisfação de restrições PSR's e de satisfação parcial de restrições PSPR's, abordando-se também algumas técnicas de solução; com ênfase especial nas heurísticas baseadas nos métodos de reparos. Para um melhor entendimento da modelagem, agregação e grau de importância das restrições nebulosas, recomenda-se uma leitura no anexo A desta tese.

3.2 Problemas Clássicos de Satisfação de Restrições - PSR's

3.2.1 Introdução

A maioria dos problemas de Inteligência Artificial, tais como planejamento, configuração, programação, etc., podem ser modelados como Problemas de Satisfação de Restrições – PSR's (*Constraints Satisfaction Problems*).

Um problema de satisfação de restrições é uma maneira de expressar requisitos simultâneos para valores das variáveis (PEARSON:1997).

O estudo de problemas de satisfação de restrições foi iniciado por Montanari (MONTANARI:1974), quando descreveu certos problemas combinatórios surgidos no processamento de imagens. O trabalho teve tanto sucesso que a mesma estrutura foi aplicada a um ampla faixa de problemas; desde então, tem sido intensamente estudada.

Para maiores detalhes sobre PSR's, Mackworth apresenta uma boa introdução em (MACKWORTH:1992).

Um PSR é basicamente um problema de busca, cuja solução requer poderosas heurísticas e meios adequados de representação. De um lado, as técnicas de satisfação de restrições podem ser vistas como simples estratégias de busca em um dado espaço; de outro lado, pode-se interpretar estas técnicas como um paradigma para a solução de problemas muito complexos, dentro dos quais é necessário definir: os objetos, suas variáveis com seu respectivo valor do domínio, e a relação entre estes objetos. A tarefa do mecanismo de inferência é a de encontrar a solução deste problema, que é feito procurando uma instância consistente de todos os objetos envolvidos. Métodos de satisfação de restrições têm a vantagem de possibilitar uma representação declarativa restrita de um dado problema. Além disso, as estratégias aplicadas para encontrar uma solução são independentes do ambiente do problema. Deste modo, tem-se uma distinção restrita entre a declaração de um problema e seu mecanismo de inferência. Na seção seguinte, introduzir-se-ão algumas definições sobre PSR's clássicos.

Problemas clássicos de satisfação de restrições são usualmente compostos de restrições *crisp*, também chamadas de booleanas ou restrições *hard*, as quais ou são satisfeitas completamente ou não, sem estados intermediários. A solução deverá satisfazer todas as restrições do problema. Se o problema tem mais de uma solução, é chamado de um problema *underconstrained*. Para alcançar uma solução, uma ou mais restrições adicionais têm de ser consideradas. Se o problema não tem solução para todas as restrições, é chamado de *overconstrained*, caso em que algumas restrições devem ser relaxadas a fim de encontrar soluções aceitáveis. Em relação aos problemas *underconstrained*, especificar preferências no domínio válido das variáveis com a finalidade de classificar soluções que, de outra maneira, poderiam ser todas iguais. Quanto aos problemas *overconstrained*, especificam-se margens nas restrições, nas quais elas podem ser relaxadas, enquanto não conduzem a resultados aceitáveis.

3.2.2 Definição de Clássicos PSR's

Técnicas de satisfação de restrições estão bem desenvolvidas para definir e resolver certos problemas no campo da Inteligência Artificial e em aplicações técnicas tais como problemas de planejamento, de configuração, de programação (*scheduling*), etc. Uma propriedade importante do PSR's é a representação declarativa e distinção restrita entre a definição de um problema e o mecanismo de inferência para encontrar uma solução viável.

Mesmo que exista uma grande variedade de definições de PSR's, os conceitos fundamentais são relativamente os mesmos. Introduzir-se-ão algumas definições básicas a partir de Pearson e Jeavon (PEARSON:1997).

Definição 3.1:

Um problema de satisfação de restrições, PSR, é especificado por uma tupla,

$$PSR = (V, D, \Re_1(S_{k1}), \cdots, \Re_n(S_{kn})) \qquad i=1...n \qquad (3.1)$$

onde,

- V é um conjunto finito de variáveis;
- D é um conjunto finito de valores (isto é chamado de *domínio de PSR*);
- Cada $\Re_i(S_{ki})$ é uma restrição,

  Em cada restrição, $\Re_i(S_{ki})$,
  - S_{ki} é uma lista ordenada de variáveis k_i, chamada de alcance da restrição;
  - $\Re_i$ é uma relação[5] sobre D de dimensão k_i, chamada de relação da restrição.

Definição 3.2:

A solução de $PSR = (V, D, \Re_1(S_{k1}), \cdots, \Re_n(S_{kn}))$ é uma designação de valores a partir de D para cada variável em V, que satisfaça todas as restrições simultaneamente.

Define-se a função h que mapeia a tupla S_{ki}, onde a cada elemento de n-tuplas de S_{ki} corresponde um elemento do conjunto D, isto quer dizer que para cada $V_i \in S_{ki}$, $h_i(V_i) \in D$, assim:

h_i: $V \rightarrow D$

$V_i \rightarrow h(V_i) \in D$,

Em outras palavras, se $S_i = \langle v_1, \ldots, v_k \rangle$, então $h(S_{ki}) = \langle h(v_1), \ldots, h(v_k) \rangle$.

O conjunto de todas as soluções para um problema PSR deverá ser denotado como Sol.(PSR). Dois problemas com o mesmo conjunto de soluções serão ditos equivalentes.

Exemplo 3.1:

Construa-se um problema simples de satisfação de restrições com variáveis $V=\{x,y,z\}$ e domínio $D = \{1,2, ..., 6\}$ (por exemplo, os números naturais de 1 a 6).

Pretende-se expressar, supostamente, os requisitos de forma que a soma de x e y deva ser igual 6, e o produto de y e z deva ser pelos menos 20. Isto pode ser realizado a partir das seguintes restrições: $\Re_1(\langle x, y \rangle)$ e $\Re_2(\langle y, z \rangle)$, onde:

- $\Re_1 = \{\langle 1,5 \rangle, \langle 2,4 \rangle, \langle 3,3 \rangle, \langle 4,2 \rangle, \langle 5,1 \rangle\}$
- $\Re_2 = \{\langle 4,5 \rangle, \langle 4,6 \rangle, \langle 5,4 \rangle, \langle 5,5 \rangle, \langle 5,6 \rangle, \langle 6,4 \rangle, \langle 6,5 \rangle, \langle 6,6 \rangle\}$

As soluções deste problema podem ser calculadas manualmente, por exemplo, o mapeamento:

[5] Uma *relação* é simplesmente um conjunto de tuples de alguma extensão fixa. A extensão das tuples é chamada de dimensão da relação.

$f(x) = 1;$
$f(y) = 5;$
$f(z) = 4.$

é uma solução, já que $f(\langle x, y \rangle) = \langle 1, 5 \rangle$, está em $\Re_1$ e $f(\langle y, z \rangle) = \langle 5, 4 \rangle$, pertence a $\Re_2$. O conjunto completo de soluções é: $\{\langle 1,5,4 \rangle, \langle 1,5,5 \rangle, \langle 1,5,6 \rangle, \langle 2,4,5 \rangle, \langle 2,4,6 \rangle\}$.

Definição 3.3: Restrição

Restrições são objetos matemáticos usados para tornar explícita a lógica por detrás de um problema. Satisfação de restrições refere-se aos algoritmos que podem ser usados para resolver tais problemas. Uma *n-ésima* restrição consiste de variáveis $V_1,...,V_n$ sobre domínios $D_1,...,D_n$ e uma relação de decisão $\Re \subseteq D_1 \times ... \times D_n$. O domínio das variáveis são valores de conjuntos finitos.

Definição 3.4: Rede de Restrições

Uma rede de restrições sobre variáveis $V_1,...,V_m$ consiste de restrições $\Re_1,...,\Re_n$, cada uma das variáveis sendo um subconjunto de $V_1,...,V_m$.

Definição 3.5: Satisfação

A tupla $(d_1,...,d_m)$ satisfaz uma restrição *m-ésima* $\Re_i$ se a subseqüência $(d_1,...,d_m)$ que corresponde às variáveis de $\Re_i$ é um elemento da relação $\Re_i$.

Definição 3.6: Solução

Uma solução de uma rede de restrições que é dada pelas restrições $\Re_1,...,\Re_n$ sobre variáveis $V_1,...,V_m$ no domínio $D_1,...,D_m$ é uma tupla $(d_1,...,d_m) \in D_1 \times ... \times D_n$ que satisfaz as restrições. A solução é uma instância completa de todas as variáveis $V_1,...,V_m$.

A tarefa de encontrar uma, algumas ou cada solução de uma rede de restrições é chamada de problema de satisfação de restrições. Em geral, PSR's são *NP-complete* e portanto difíceis de resolver. Desta maneira, têm-se desenvolvido algoritmos que não computam uma solução, mas que restringem o espaço de busca para tal solução, limitando o domínio destas variáveis e cumprindo com certos requerimentos adicionais de nível local. Estes requerimentos podem ser expressados em termos de consistência, *k*-consistência, e firme *k*–consistência, que são definidas a seguir:

Definição 3.7: Consistência

Uma rede de restrições sob variáveis $V_1,...,V_m$ no domínio $D_1,...,D_m$ é consistente se, para cada $j \in \{1,...,m\}$: cada tupla $(d_1,...,d_{j-1}, d_{j+1},...,d_m)$, $d_i \in D_i$ para todo $i \in \{1,...,m\} \setminus \{j\}$, que satisfaz as restrições da rede pode ser estendida a uma tupla *m-ésima* $(d_1,...,d_m)$, $d_j \in D_j$, que satisfaz as restrições.

Definição 3.8: *k*-Consistência

Uma rede de restrições é *k-consistente* se cada variável k da sub-rede é consistente.

Definição 3.9: Firme *k*-Consistência

Uma rede de restrições é firme *k-consistente* se esta é *i-consistente* para toda $i \leq k$.

3.2.3 Restrições em PSR's

Restrições são um método geral para formular dependências e relações entre diferentes objetos (variáveis). A formulação de restrições dá mais independência para solucionar um problema em comparação aos métodos funcionais. Um termo funcional do tipo:

$$z = x + y\,; \tag{3.2}$$

pode somente ser avaliado se os valores de x e y são conhecidos. Uma restrição da forma:

$$\Re_{xyz} : z = x + y \tag{3.3}$$

conduz à inferência de diferentes relações, que podem ser:

- ***Verificação de Consistência:*** Todas as três variáveis são instanciadas. Pode-se verificar se $\Re_{xyz}$ é satisfeita ou não.
- ***Computação:*** Duas das três variáveis são instanciadas. Computa-se o valor da terceira variável.
- ***Filtragem:*** Uma ou nenhuma das três variáveis é instanciada. Filtra-se o domínio das variáveis restantes e reduzir o espaço de busca.

Restrições podem ser formuladas *intencionadas* ou *extensionadas*. As restrições *intencionadas* são formuladas como relações compactas, por exemplo, $x<y$, que tem de ser melhoradas por métodos de procedimento. Restrições *extensionadas* são conjuntos finitos de possíveis tuplas de valores, por exemplo, {(6,9),(3,8),...}.

Dependendo do tipo de restrição pode-se diferenciar entre:

- ***Restrições Booleanas*** : O mecanismo de inferência verifica todas as variáveis de uma solução sobre a validade de seus valores e retorna verdadeiro ou falso sobre cada verificação. Restrições booleanas são usadas especialmente em algoritmos tradicionais de busca tais como *backtracking, Branch&Bound,* etc.

- ***Restrições Numéricas*** *:* Problemas que são descritos por relações numéricas entre as variáveis e que precisa de técnicas que encontre instâncias das variáveis a fim de satisfazer um número máximo de restrições. Tais sistemas são chamados de *problemas numéricos de otimização.*

3.3 Problemas de Satisfação Parcial de Restrições – PSPR's

O método acima pode ser utilizado na formulação para a solução de problemas clássicos de satisfação de restrições. Restrições parcialmente satisfeitas, relaxa o conceito de tal forma que somente um subconjunto de variáveis é instanciado, satisfazendo um subconjunto das restrições. Algumas restrições podem ser violadas a fim de permitir instâncias mais válidas.

Um problema de satisfação de restrições é solúvel se existe uma tupla $(d_1,...,d_m) \in D_1 \times ... \times D_n$ que satisfaça cada restrição. Um problema de satisfação de restrições sem solução é chamado de *overconstrained.* Uma *solução parcial* de uma rede de restrições $\Re_1,...,\ \Re_n$, sobre variáveis $V_1,...,V_m$ no domínio $D_1,...,D_n$ é uma tupla $(d_1,...,d_m) \in D_1 \times ... \times D_n$ que viola algumas restrições. Portanto, restrições parcialmente satisfeitas são um *relaxamento* de restrições satisfeitas. Uma solução parcial com o menor número de restrições violadas é usualmente declarada solução de um problema de restrições parcialmente satisfeitas.

3.4 Resolução de Problemas de Satisfação de Restrições

O espaço de busca de um problema de satisfação de restrições é geralmente muito amplo, e seu tamanho pode depender exponencialmente do número de variáveis relacionadas e do tamanho de seus respectivos domínios. Na maioria dos casos, as soluções válidas são espalhadas esparsamente sobre o espaço de todas as possíveis designações dos valores das variáveis. Distinguem-se, aqui, dois métodos básicos para resolver um problema de satisfação de restrições. Uma classe é a abordagem *construtiva*, em que a solução é gerada sistematicamente por extensão da designação do valor a uma variável válida parcialmente até que uma solução completa seja alcançada.

A segunda maneira é gerar uma primeira instância de todas as variáveis, com base aleatória, é "reparar" todos os valores designados inconscientes e selecionar outras designações a partir dos domínios das respetivas variáveis. Para determinar o grau de satisfação de uma solução, ambos algoritmos, o construtivo e o de reparo, devem fornecer uma função de avaliação, que esteja capacitada para agregar um grau de satisfação total para um conjunto de restrições. Em problemas PSR's clássicos, as restrições são admitidas como binárias e finitas. Neste caso, a função de avaliação é equivalente a um teste de consistência para uma solução. Considerando a abordagem baseada em reparos e satisfação parcial de restrições, precisar-se-á de uma função de avaliação para estabelecer uma categoria ou ordem de instâncias ou soluções, a fim de classificá-las ou decidir entre possíveis passos de reparo.

3.4.1 Método Construtivo

Algoritmos construtivos geram designações subseqüentes aos valores das variáveis até que a solução seja encontrada. Um algoritmo construtivo fornece um comportamento determinístico e garante que uma solução ótima, caso ela exista, seja encontrada. Estes algoritmos oferecem um melhor desempenho para pequenos números de variáveis e problemas caracterizados por situações simples. O progresso da designação de valores é geralmente manifestado por um algoritmo de árvore de busca.

3.4.1.1 O Algoritmo *Backtracking*

O algoritmo *backtraking* (HARALICK:1980) (TSANG:1993) tenta expandir uma solução parcial de um problema a uma solução completa, designando valores subsequentes às variáveis de um problema de satisfação de restrições. Em sua versão padrão, o algoritmo *backtraking* atravessa as variáveis numa ordem predeterminada, designa provisoriamente valores consistentes a uma subsequência $(V_1,...,V_i)$ das variáveis e tenta anexá-los a uma nova instância de V_{i+1} tal que o conjunto completo seja consistente.

Este algoritmo começa selecionando um valor a partir das primeiras variáveis do domínio. Em seguida, é selecionado um valor para a próxima variável do domínio. Cada nova seleção é verificada com as designações prévias que estão relacionadas à designação atual por restrições. Se uma designação *n-ésima* é inconsistente e viola uma restrição, o algoritmo tenta encontrar um novo valor, a partir do domínio, da variável V_n. Se nenhum valor do domínio das variáveis satisfizer a relação entre as restrições, ocorre uma falha local, impondo a necessidade de se observar as designações prévias. Este caso é chamado de situação de "beco sem saída". O algoritmo de *backtracking* então volta à variável $V_{(n-1)}$ e assume este valor sendo incompatível com a variável V_n. Um novo valor para a variável $V_{(n-1)}$ é selecionado e o algoritmo tenta de novo uma instância da variável V_n, com um valor apropriado do domínio. Quando mudanças no nível (n-1) não conduzem a uma designação correta sobre o nível *n-ésimo*, então a designação (n-2) tem de ser considerada incompatível com sua sucessora, e assim continuadamente.

O algoritmo é bem sucedido quando são designados valores a todas as variáveis sem violar as restrições. Um PSR não tem solução completa se todas as combinações forem verificadas e não forem encontradas instâncias válidas das variáveis.

Estudos prévios, (KUMAR:1992), têm demonstrado que há duas razões principais para o desempenho pobre do algoritmo *backtracking:* o chamado *trashing* e a verificação redundante das restrições. O primeiro ocorre quando, encontrada uma violação, o algoritmo tenta realizar alocações no nível atual, enquanto a razão para a violação situa-se em outro lugar da árvore de busca. Têm sido desenvolvidos alguns algoritmos refinados a partir do *backtracking* para se evitar estes problemas, tais como o *backjumping* e o *backmarking* (NADEL:1989), entre outros, nos quais o número de nós visitados na árvore de busca pode ser reduzido, resultando igualmente numa redução do número de restrições verificadas.

3.4.1.2 O Algoritmo *Branch&Bound*

O *Branch&Bound* é uma extensão do *backtracking*, utilizado especialmente para problemas de satisfações parciais de restrições. Este algoritmo leva em consideração a maior pontuação de avaliação até o momento presente e tenta encontrar a melhor solução. Quando uma solução parcial não pode conduzir a uma melhor solução do que a encontrada até então, a designação para as variáveis restantes poderão ser descartadas. Isto ocorre quando as variáveis na trajetória atual de busca violam as restrições numa margem maior que a solução encontrada no momento presente. O menor grau de violação encontrado até aqui é chamado de "limite necessário". Uma nova instância com um grau menor de violação atualiza o limite necessário. O algoritmo tem sucesso quando a solução atinge um dado "limite suficiente", ou quando tiverem sido testadas todas as combinações das variáveis. Nas aplicações de tempo real, o "tempo limite" é fornecido freqüentemente, após o qual, o algoritmo entrega a melhor solução até este momento, característica conhecida como "em qualquer tempo".

3.4.1.3 Métodos Prospectivos e Retrospectivos

Devido ao fraco desempenho dos algoritmos clássicos na solução de PSR's, foram propostas algumas melhorias. Existem duas abordagem principais para reduzir o *trashing* e a verificação redundante das restrições vistas no *backtracking*, a abordagem prospectiva e retrospectiva.

O algoritmo prospectivo testa um novo valor entre o domínio das variáveis, que não está contido ainda na trajetória atual de busca, a fim de detectar inconsistências para variáveis futuras, antes que esta seja considerada. Os mais importantes métodos prospectivos são basicamente *Partial Look Ahead* e *Forward Cheking* (HARALICK:1980) (LIU:1998)

Métodos retrospectivos, como o *Backchecking*, (HARALICK:1980) *Backmarking* e *Backjumping*, estes dois últimos desenvolvidos por Gaschnig (NADEL:1989), testam um novo valor entre os valores contidos na trajetória atual de busca das variáveis.

3.4.2 Método por Reparos

Heurísticas com base em reparos, conhecidas também na literatura como "técnicas de melhorias iterativas", têm sido aplicadas com sucesso para resolver problemas que pareciam previamente intratáveis, alcançando uma pontuação melhor do que a abordagem construtiva. Esta pontuação começa a partir de uma instância vazia e constrói sucessivamente mais e mais instâncias parciais, até atingir uma instância completa. Minton et al. (MINTON:1992) analisaram a solução do problema de programação do telescópio espacial Hubble com tarefas complexas para cada medida. Eles estudaram o desempenho de diferentes heurísticas, testando-as na solução dos problemas de *n-queens* e de *graph-coloring*. Estes problemas têm auxiliado os pesquisadores (por muito tempo tidas como clássicas referências — *benchmark*) a

estudarem a eficiência de novas heurísticas e algoritmos para a solução de problemas de satisfação de restrições. O algoritmo baseado em reparos proposto por Minton et al. (MINTON:1992) apresentou os melhores resultados para o problema de *million-queens*, encontrando soluções em menos de 4 minutos sobre uma *Workstation Sparc* 1, enquanto a melhor abordagem construtiva analisada (*backtracking*) se converte intratável para n>1000. A figura 3.1 resume a média das estatísticas coletadas sobre 100 experimentos para diferentes n. Observa-se na figura que o método baseado em reparos exibe um tempo lineal e uma complexidade espacial para altos n.

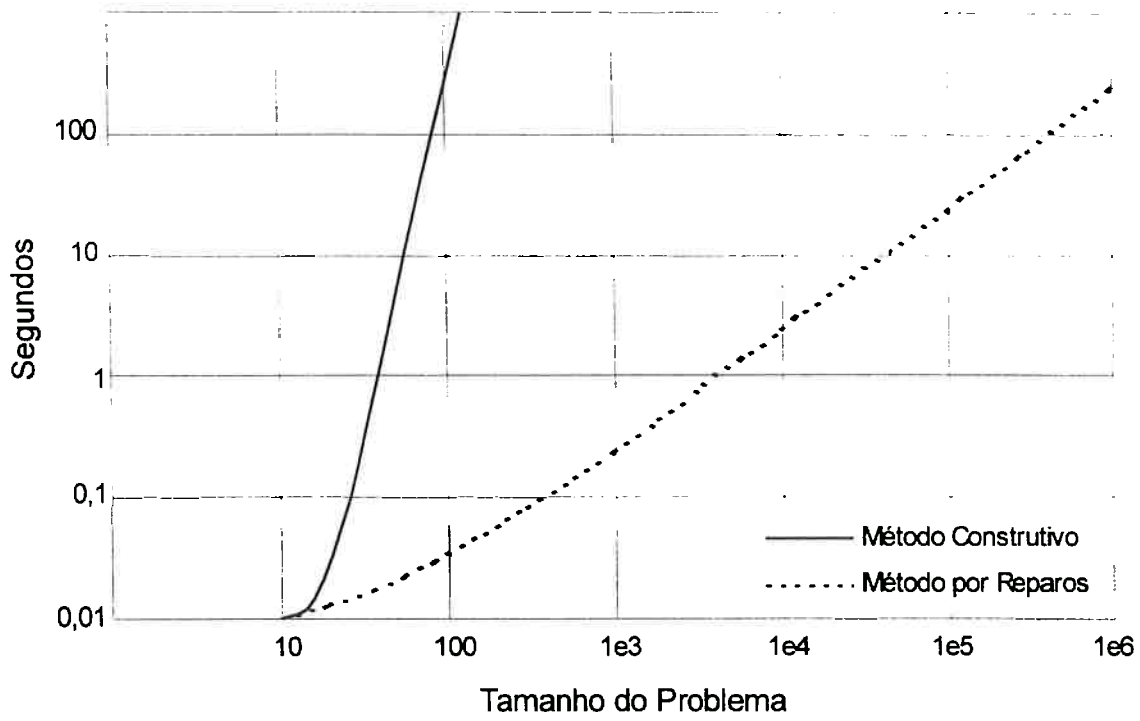


Figura 3.1 – Tempo médio de solução para o problema de *n-queens* sobre 100 experimentos para diferentes n utilizando um Método por Reparos *versus* um Método Construtivo (Resultados tomados de Minton et al. (MINTON:1992))

Métodos com base em reparos possuem a propriedade inerente de fornecer "em qualquer tempo" uma boa solução desde o inicio do processo, podendo no entanto continuar a busca, tanto quanto o tempo o permita, até encontrar uma solução melhor.

Ao contrário dos métodos construtivos, a abordagem com base em reparos começa a partir de uma designação completa dos valores das variáveis. Estes valores são freqüentemente selecionados a partir de gerações aleatórias dos respectivos domínios. Na maioria dos casos, esta seleção inicial pode ser subótima e errônea do ponto de vista do cumprimento das restrições. Métodos com base em reparos tentam melhorar a consistência das designações, aplicando heurísticas locais de reparo, até que seja atingida uma instância consistente para todas as variáveis.

O algoritmo geral do método por reparos é mostrado na figura 3.2 e é uma adaptação do apresentado por Dorn et al. (DORN:1994) que consiste de um ciclo principal fechado, onde a cada iteração a programação é melhorada até atingir-se um critério de parada, que pode ser tempo consumido. A procura, a partir de uma melhor

programação intermediária, por uma próxima programação, convertendo-a em uma melhor programação, é descrita aqui como uma iteração anilhada. Chamar-se-á de *passo reparo* à modificação da atual programação à uma melhor programação. Um passo reparo pode consistir em vários modificadores simples da programação. A função booleana "aceitável" do diagrama da figura 3.2 é a principal diferença entre as duas técnicas que serão apresentadas no seguinte capítulo, reparo em profundidade e reparo por modificações aleatórias. Estas heurísticas escolhidas para a aplicação em estudo divergem basicamente no método utilizado para escapar dos ótimos locais.

Outros algoritmos que podem utilizar também as técnicas por reparos, como a busca *tabu* (*tabu search*), os algoritmos genéticos, e *simulated annealing,* não serão tratados nesta tese, ficando como proposta de seu desempenho para futuras pesquisas.

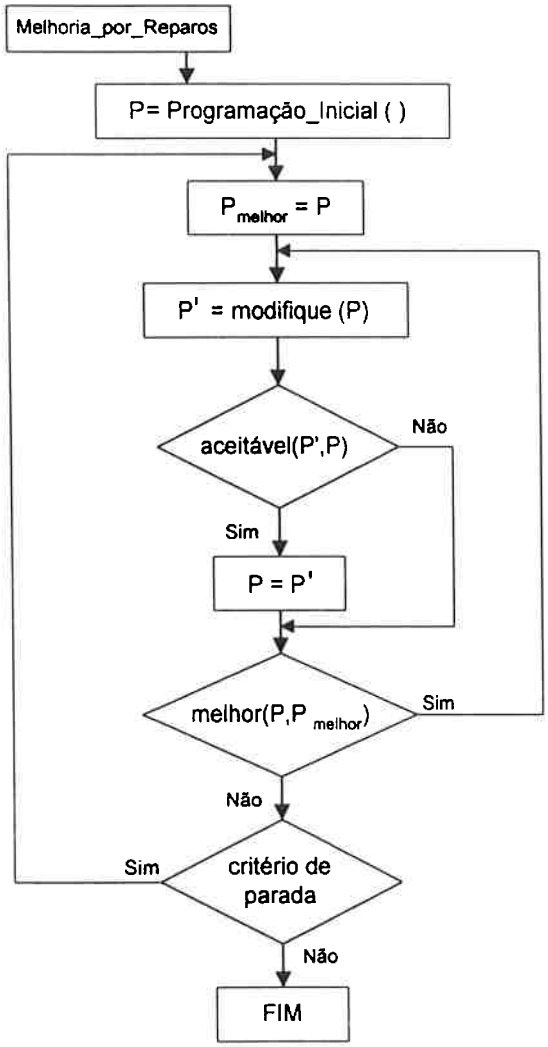


Figura 3.2 - Algoritmo Padrão do Método por Reparos para Programações (Algoritmo Iterativo de Melhoria)

3.4.2.1 Reparo em Profundidade

Jürgen Dorn et al. apresentaram em seu trabalho (DORN:1994) comparações entre diferentes técnicas de reparo para a otimização na produção em uma planta de aço; o método de reparo em profundidade demonstrou ser a melhor heurística para procurar deterministicamente por passos de reparos.

O algoritmo de reparo em profundidade foi inspirado na abordagem de Korf (KORF:1987) e D.S. Nau (NAU:1993); foi desenvolvido para avaliar se eram necessárias técnicas estocásticas na procura de boas soluções. Este é uma combinação entre *depth-first backtracking search* e *breadth-first search* e tem a vantagem de exigir somente espaço linear, garantindo encontrar o trajeto mais curto para uma melhoria.

O algoritmo padrão mostrado na figura 3.3 funciona como segue; numa primeira tentativa realiza-se uma busca à profundidade 1. Isto significa que é executada uma modificação começando-se a partir da programação inicial. A execução de uma modificação implica que todas as possíveis modificações desse tipo sejam tratadas sobre a posição violada. Se nenhuma melhora é encontrada, realiza-se, em uma segunda tentativa, em uma busca *depth-firts backtracking* a uma profundidade 2. Isto significa que o operador de modificação é aplicado duas vezes, permitindo também aceitar uma programação intermediária pior em termos de avaliação, contanto que a programação final seja a melhor da aplicação seqüencial dos operadores de modificação. Se ainda não foram encontradas melhorias, a profundidade é aumentada iterativamente até que se atinja um passo de reparo com sucesso ou se alcance um limite causando o término do algoritmo. A figura 3.4 mostra como são visitados os nós pelo algoritmo da árvore de busca.

Uma característica conveniente da técnica de melhoria iterativa é a característica "em qualquer tempo"; por exemplo, o algoritmo pode ser detido em qualquer momento e sempre fornecer a melhor solução alcançada até esse instante.

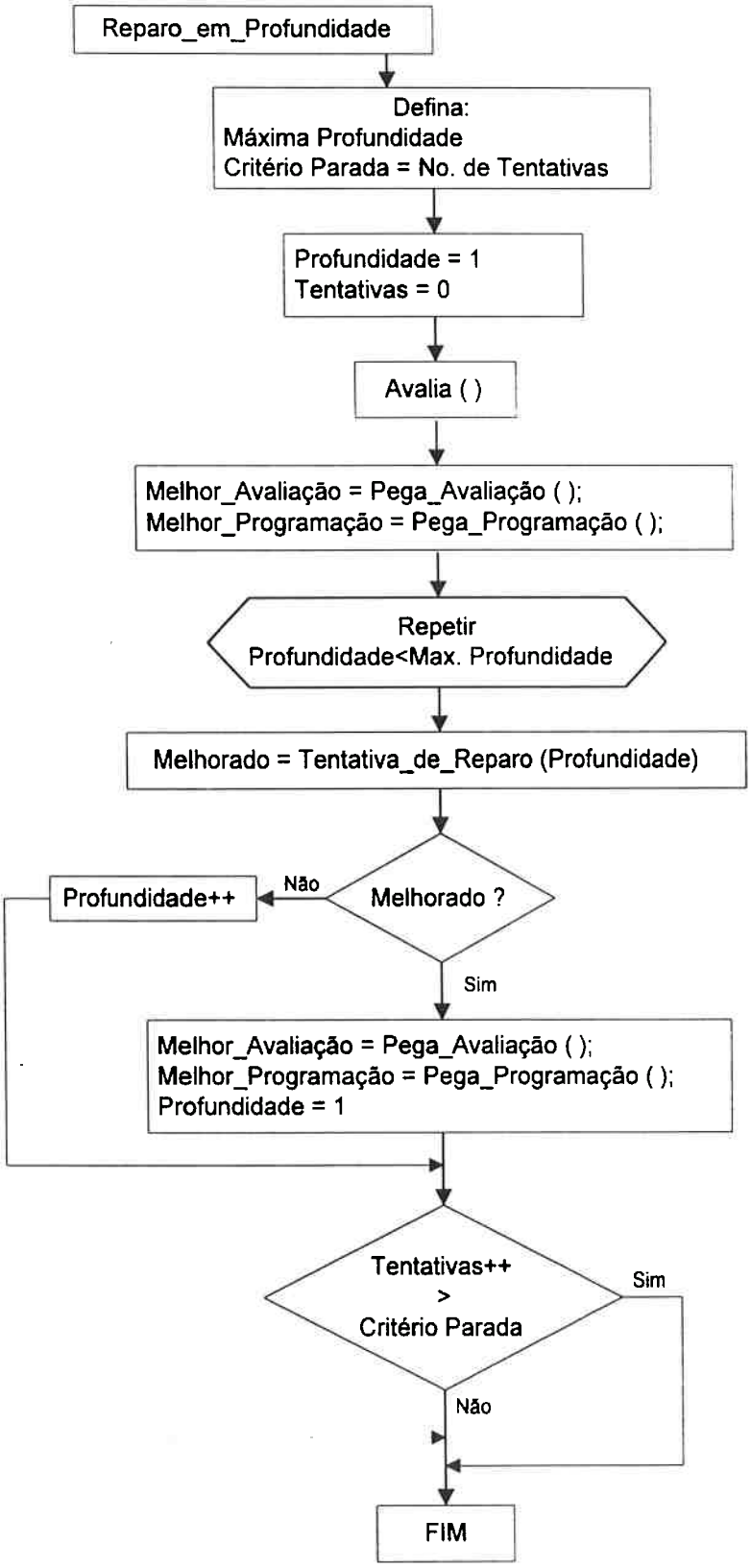
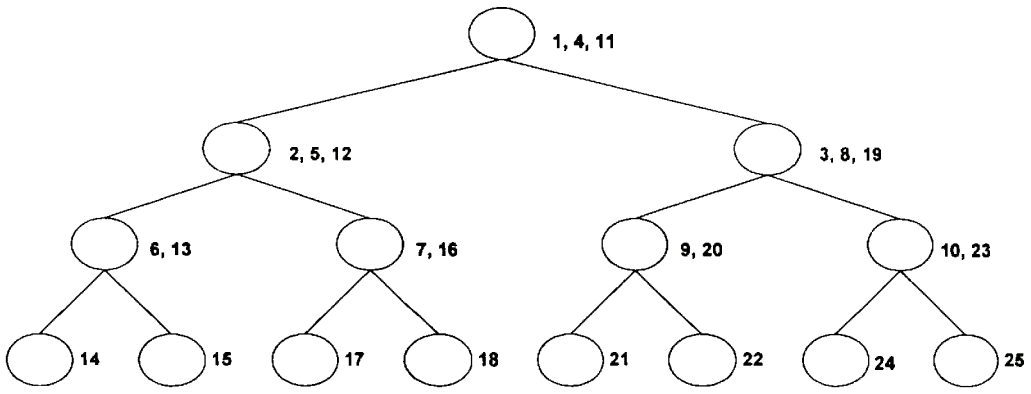


Figura 3.3 – Algoritmo de Otimização por Reparos em Profundidade



Profundidade 1: 1, 2, 3
Profundidade 2: 4, 5, 6, 7, 8, 9, 10
Profundidade 3: 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25

Figura 3.4 – Como o Algoritmo Iterativo de Profundidade atravessa a Árvore de Busca

3.4.2.2 Reparo por Modificações Aleatórias

A abordagem de reparo em profundidade testa, deterministicamente, todos os nós da árvore de busca; isto garante teoricamente um método exaustivo se o tempo o permite. Contudo, isto pode consumir muito tempo, caso muitos níveis da árvore de busca devam ser examinados ou se o espaço de busca for muito amplo.

Para esta aplicação, tem-se implementado um algoritmo de melhoria iterativa a partir do algoritmo padrão, figura 3.2, que decide aleatoriamente que turnos e que operadores devam ser aplicados. A nova programação gerada é avaliada e se nenhuma melhoria é encontrada, outro modificador é gerado aleatoriamente. O algoritmo é apresentado na figura 3.5, sendo *tentativa_de_reparo* uma função recursiva que testa um determinado passo de reparo aplicando um operador de modificação *profundidade* vezes. A vantagem frente ao algoritmo de reparo em profundidade deve-se ao fato de que nem todos os sucessores da programação devem ser avaliados.

Neste estudo, experimenta-se com estes dois últimos procedimentos (reparo em profundidade e reparo por modificações aleatórias para a aplicação em estudo), estabelecendo o ganho em eficiência e tempo de computação.

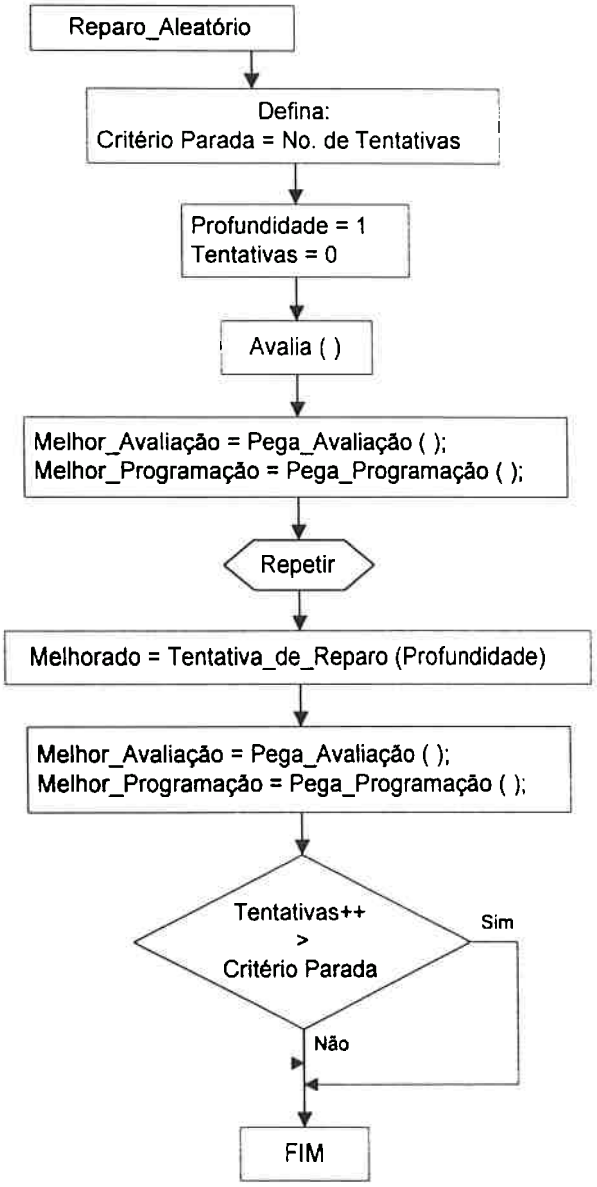


Figura 3.5 – Algoritmo de Otimização por Modificações Aleatórias

3.5 Problemas de Satisfação por Restrições Nebulosas – PSRN's

3.5.1 Introdução

No anexo A, apresenta-se uma introdução aos conjuntos nebulosos, indispensável para a compreensão deste capítulo. O anexo contém definições importantes que serão usadas em todo o trabalho, o que permitirá servir de base para a formulação do problema de programação de turnos, por meio da modelagem das restrições utilizando a teoria dos conjuntos nebulosos (restrições nebulosas).

3.5.2 Por que Restrições Nebulosas ?

As ferramentas tradicionais para a modelagem formal do raciocínio e da computação são determinísticas ou *crisp,* mas em muitas situações de tomada de decisão da vida real não é possível satisfazer perfeitamente nem todos os objetivos nem todas as condições, ao tentar modelar um ambiente que tem que ser otimizado por meio de um problema de satisfação de restrições. Diferentes motivos nos conduzem a introduzir o conceito de restrições nebulosas para a modelagem deste tipo de problema, de modo a conseguir uma representação mais apropriada para situações de tomada de decisão do mundo real.

De um lado, nem sempre é possível formular certas condições do mundo real de um modo numérico preciso, por meio de restrições *crisp*, devido à falta de um critério estrito para a validade de todas as variáveis do sistema. A maioria das aplicações do mundo real não é rígida, e as margens de satisfações são do tipo flexível. Considerando o problema de programação como um especial PSR, encontramo-nos frente a diferentes tipos de imprecisões e incertezas. Pode-se imaginar, por exemplo, que uma operação da programação pode começar "um pouco" mais cedo e que podem ser aceitos "pequenos" desvios dos valores ótimos. No caso de utilizar-se métodos clássicos de PSR's, poder-se-ia rejeitar uma programação eficiente, mesmo quando as violações são insignificantes com relação à precisão dos limites realísticos de previsibilidade. Neste sentido, é desejável estabelecer não só funcionalidade e flexibilidade adicionais na modelagem de PSR's como também técnicas de solução para o manuseio de dados imprecisos e vagos na modelagem de restrições com diferentes graus de importância, critérios antagônicos e relaxamento das restrições, a fim de resolver um PSR *overconstrained.*

Por outro lado, o conhecimento e o julgamento humanos concedem valores às variáveis imprecisas, como por exemplo, para "velocidade": *muito baixa, baixa, rápida, muito rápida,* etc. Como se vê, poderia ser conveniente expressar tais variáveis, usando *variáveis lingüísticas* (ZADEH:1975) (vide anexo A), fazendo um mapeamento da gama de preferência dos parâmetros sobre valores correspondentes da função de pertinência da representação do conjunto nebuloso desta variável. Variáveis lingüísticas são variáveis cujos valores são palavras ou frases em uma linguagem natural ou artificial. Por exemplo, "velocidade" é uma variável lingüística, se seus valores são lingüisticos em vez de numéricos assim como viu-se no exemplo anterior.

3.5.3 Definição do Problema de Satisfação por Restrições Nebulosas

Freuder et al. (FREUDER:1992) observaram que o universo no domínio de PSRN não é somente preto e branco, consistente ou inconsistente, mas permitindo também "sombras cinzentas". Bonissone (BONISONE:1992) descreveu e comparou vários métodos formais para manusear tais "sombras cinzentas". Em particular, métodos pertencentes ao amplo campo da teoria dos conjuntos nebulosos têm sido aplicados para PSRN, por exemplo, pelo grupo de pesquisas de Dubois, Prade e Fargier nos trabalhos (DUBOIS:1986) e (DUBOIS:1994), e por Fargier, Lang e Schiex em (FARGIER:1993) e, ainda, por W. Gesguen em (GUESGUEN:1994). Para o problema de programações

têm-se aplicado métodos similares inspirados na lógica nebulosa em (DORN:1995) e (SLANY:1996), todos tratando o mesmo problema de *"job-scheduling"* em uma planta de produção de aço. Esta mesma aplicação inspirou o desenvolvimento de um conjunto de bibliotecas em *C++* chamadas de *FLIP, Santa et al. (SANTA:1996) e Reichel et al. (REICHEL:1997), que através de sua reutilização permite a modelagem de problemas combinatórios em diferentes áreas de aplicação incluindo programação, projeto, configuração, planejamento e classificação, todas estas formulando o problema como um PSRN.

Definição 3.10: Problemas de Satisfação por Restrições Nebulosas (PSRN's)

Um PSRN consiste de:

$$PSRN\langle V, \Re\mu, P, T, D\rangle \tag{3.4}$$

onde V é o conjunto de variáveis; D é o conjunto de universos (domínios) para cada variável em V; $\Re\mu$ é um conjunto de restrições – em que cada restrição é uma função de pertinência μ de valores designados entre [0,1], tendo um peso associado p_r, representando sua importância ou prioridade –; mais um esquema de pesos P – por exemplo, uma função que combina um grau de satisfação da restrição $\mu(r)$ com p para produzir um grau de satisfação da restrição ponderada $\mu^P(r)$ –; e, T é uma função de agregação – por exemplo, uma *t-norma* (seção 3.5.5.1 para mais detalhes sobre normas triangulares) que, dado $\Re\mu$, produz uma simples ordem parcial sobre os valores designados. Uma instância é uma solução de PSRN, se este é o máximo elemento da ordem parcial $T(\Re\mu)$.

A diferença entre PSRN's e os PSR's clássicos é o fato de suas restrições poderem ser violadas; por exemplo, o grau de satisfação de zero significa que a instância correspondente tem que ser rejeitada. A partir da definição de μ e P, é possível definir a restrição que nunca será rejeitada, por exemplo, contribuindo sempre com uma pontuação agregada do PSRN.

3.5.4 Restrições Nebulosas

Algumas restrições do domínio de certas aplicações são freqüentemente especificadas vagamente, sendo assim candidatas a serem formuladas como restrições nebulosas. Adicionalmente, restrições nebulosas são melhor apropriadas para serem aplicadas em algoritmos de satisfação de restrições com base em reparos. Combinando-as com heurísticas, é possível chegar a soluções chamadas de sub-ótimas, mesmo assim estas soluções acabam por ser aceitáveis, uma vez que possa ser impossível encontrar uma solução geral melhor para um dado problema.

A modelagem e representação das restrições por lógica nebulosa será mostrada a seguir:

Definição 3.11: Restrições Nebulosas

Uma restrição *crisp k-ésima*, $\Re_{crisp}$, de variáveis $x_1,...,x_k \in D_1 \times ... \times D_k$, pode ser formalizada como uma relação $\Re_{crisp}$ com sua função característica 1_{crisp}

$$\begin{array}{lll} 1_{crisp}: & D_1 \times ... \times D_k & \{0,1\} \\ & (x_1,...,x_k) & 1_{crisp}(x_1,...,x_k) \end{array} \quad (3.5)$$

Obviamente, só aquelas *k-tuplas* que designadas como 1, a partir do $\Re_{crisp}$, são restrições satisfeitas.

Uma restrição *soft k-ésima*, $\Re_{soft}$, de variáveis $x_1,...,x_k \in D_1 \times ... \times D_k$, pode ser formalizada como uma relação $\Re_{soft}$ com sua função característica μ_{soft}

$$\begin{array}{lll} \mu_{soft}: & D_1 \times ... \times D_k & [0,1] \\ & (x_1,...,x_k) & \mu_{soft}(x_1,...,x_k) \end{array} \quad (3.6)$$

R_{soft} designa um valor de pertinência nebuloso $\mu_{soft}(x_1,...,x_k)$ no intervalo [0,1] para cada elemento k $(x_1,...,x_k)$, onde μ_{soft} representa o nível de preferência entre diferentes tuplas. O valor de 1 significa que todas as restrições são perfeitamente satisfeitas, e o valor de 0 indica que as instância de $(x_1,...,x_k)$ não são satisfeitas completamente.

A modelagem de restrições *soft* por intermédio de conjuntos nebulosos proporciona uma variedade de características. Segue abaixo algumas delas:

- a definição de restrições *soft* nos dá a possibilidade de medir o grau de satisfação das restrições.
- no caso de problemas "*underconstrained*", especificam-se preferências no domínio válido das variáveis, a fim de classificar soluções que de outro modo poderiam ser todas iguais.
- no caso de problemas "*overconstrained*", pode-se especificar para as restrições margens, nas quais elas podem ser relaxadas enquanto não conduzem a resultados aceitáveis.
- podem ser modeladas restrições nebulosas de níveis de importâncias diferentes, especificando prioridades entre as restrições.
- é possível encontrar soluções sub-ótimas e boas, usando combinações heurísticas. Estas soluções são freqüentemente aceitáveis até que seja possível encontrar uma solução global melhor para um dado problema.
- possibilita a definição de valores incertos por meio de distribuições possibilísticas (vide anexo A).

Exemplo 3.2: Seleção de Candidatos

Um candidato em um concurso deverá demonstrar os seguintes requisitos: ser jovem, inteligente e assistir freqüentemente à TV. Um sistema especialista com base nas

regras nebulosas deverá selecionar um candidato entre um conjunto de competidores. Uma regra típica e simplificada é apresentada a seguir:

SE	*candidato.QI*	**É**	*ALTO*	**E**
	candidato.IDADE	**É**	*JOVEM*	**E**
	candidato.TV	**É**	*FREQÜENTEMENTE*	
ENTÃO	*candidato*	**É**	*selecionado*	

Aqui vê-se novamente o uso de termos lingüísticos tais como *ALTO, JOVEM* e *FREQÜENTEMENTE.* Estes termos, correspondentes ao coeficiente intelectual, idade real e freqüência de assistir TV, são descritos por funções apropriadas de pertinência (vide anexo A). As restrições para selecionar um candidato podem ser formuladas como restrições nebulosas, já que os termos jovem, alto, etc., são de definição indistinta. Assume-se que dois candidatos foram selecionados pelo sistema, chamados de:

Candidato_1:	*IDADE*	=	21 *anos*
	QI	=	112
	TV	=	6 *horas*
Candidato_2:	*IDADE*	=	29 *anos*
	QI	=	128
	TV	=	2 *horas*

Ambos os candidatos parecem atingir os requisitos. Mas, de fato, somente um candidato deve ser selecionado, levando-nos à seguinte questão: qual deles é o melhor? Para medir a satisfação total das três restrições é necessário:

- *agregar* as restrições adequadamente;
- introduzir *prioridades* entre as restrições.

3.5.5 Agregação de Restrições Nebulosas

O próximo passo para resolver um problema geral de satisfação de restrições é satisfazer todas as restrições ao mesmo tempo, isto se consegue, substituindo-as por valores numéricos. Por exemplo, uma instância de todas as variáveis satisfazendo todas as restrições simultaneamente. Em problemas clássicos de satisfação de restrições, a solução de um problema de otimização de múltiplos critérios é obtida, tomando a interseção de todas as relações, por exemplo, calculando a conjunção de todas as restrições:

$$1_{\Re_1 \cap \ldots \cap \Re_n}(x_{j_1}, \ldots, x_{jm}) = \bigcap_{i=1}^{n} 1_{\Re_i}(x_{i_1}, \ldots, x_{ik}) \tag{3.7}$$

com $\{x_{j_1}, \ldots, x_{jm}\} = \bigcap_{i=1}^{n} \{x_{i_1}, \ldots, x_{ik}\}$ e $m = \left| \bigcap_{i=1}^{n} \{x_{i_1}, \ldots, x_{ik}\} \right|$.

Qualquer restrição violada resulta em uma rejeição total da instância. No entanto, em aplicações da vida real, restrições não podem ter o mesmo nível de importância. Para restrições *soft* (como definido na equação 3.6) é adaptado o mesmo conceito que em PSR clássicos (equação 3.7), escolhendo um operador apropriado de conjunção nebulosa. Zadeh (ZADEH:1965) propôs o uso do operador *mínimo* (MIN), mas existe na literatura uma grande variedade de outros operadores, como será visto a partir da próxima seção.

Particularmente, é interessante o uso de *t-normas* (normas triangulares) como conjunção devido à simplicidade na sua estrutura algébrica. Mas, segundo Slany (SLANY:1996) o uso de agregações tipo *t-normas* impede a compensação e, nem sempre, consegue uma boa representação dos diferentes graus de satisfação das restrições. Isto nos leva a testar outros operadores encontrados na literatura, apresentados por (YAGER:1988), (ZIMMERMANN:1996) e (CÓRDON:1997), que junto às *t-normas* e *t-conormas* serão descritos a seguir.

3.5.5.1 Operadores *t-normas* e *t-conormas*

Os operadores convencionais T de L. A. Zadeh (ZADEH:1965) e (ZADEH:1973), MIN e MAX, têm sido usados em quase todos os projetos de controladores de lógica nebulosa e mesmo na modelagem de alguns processos de tomada de decisão. Contudo, alguns estudos teóricos e experimentais indicam que outros tipos de operadores T podem trabalhar melhor em algumas situações, especialmente no contexto de tomada de decisões (GUPTA1991A). Por exemplo, segundo Dubois et al. (DUBOIS:1986), o operador produto pode ser preferível a operador MIN.

Por outro lado, quando escolhe-se um conjunto de operadores T para um dado processo, há de se considerar suas propriedades, precisão do modelo, simplicidade, implementação computacional e *hardware*, etc. Por esta e outras razões, é de interesse usar outros conjuntos de operadores T na modelagem de processo de tomada de decisão, de modo que opções múltiplas, mais apropriadas, estejam disponíveis para a seleção dos operadores T (GUPTA:1991A).

Os operadores do tipo *t-normas*, *t-conormas* e *funções de negação* são usados para calcular os valores de pertinência da interseção, união e complemento dos conjuntos nebulosos, respectivamente. A definição dos operadores T tem sido tratada por muitos pesquisadores. Nesta seção, porém, dar-se-á atenção a alguns operadores T, que serão usados posteriormente em testes experimentais, de modo a demonstrar a sua aplicabilidade no projeto de modelos nebulosos na otimização de programações.

Definição 3.12: Operadores *t-normas*

Uma função T:[0,1]×[0,1]→[0,1] é uma *t-norma,* se para toda $\mu_A(x)$, $\mu_B(x)$, $\mu_C(x)$, $\mu_D(x) \in [0,1]$, verificam-se as seguintes propriedades (MIZUMOTO:1989), (GUPTA:1991A) (GUPTA:1991B):

(1.1) $T(0,0)=0$; $T(\mu_A(x),1) = T(1,\mu_A(x)) = \mu_A(x), x \in X$ *(elemento neutro)*

(1.2) Se $\mu_A(x) \leq \mu_B(x)$ e $\mu_C(x) \leq \mu_D(x)$
então $T(\mu_A(x), \mu_C(x)) \leq T(\mu_B(x), \mu_D(x))$ *(monotonicidade)*

(1.3) $T(\mu_A(x), \mu_B(x)) = T(\mu_B(x), \mu_A(x))$ *(comutatividade)*

(1.4) $T(T(\mu_A(x), \mu_B(x)), \mu_C(x)) = T(\mu_A(x), T(\mu_B(x), \mu_C(x)))$ *(associativa)*

O que os operadores *T, t-normas,* fazem para a interseção, os operadores *S, t-conormas,* fazem para a união, correspondendo à disjunção na lógica clássica.

Definição 3.13: Operadores *t-conormas*

Uma função $S:[0,1]\times[0,1]\rightarrow[0,1]$ é uma *t-conorma,* se para toda $\mu_A(x)$, $\mu_B(x)$, $\mu_C(x)$, $\mu_D(x) \in [0,1]$; verificam-se as seguintes (MIZUMOTO:1989), (GUPTA:1991A) (GUPTA:1991B):

(1.5) $T(1,1)=1; S(\mu_A(x),0) = S(0,\mu_A(x)) = \mu_A(x), x \in X$ *(elemento neutro)*

(1.6) Se $\mu_A(x) \leq \mu_B(x)$ e $\mu_C(x) \leq \mu_D(x)$
então $S(\mu_A(x), \mu_C(x)) \leq S(\mu_B(x), \mu_D(x))$ *(monotonicidade)*

(1.7) $S(\mu_A(x), \mu_B(x)) = S(\mu_B(x), \mu_A(x))$ *(comutatividade)*

(1.8) $S(S(\mu_A(x), \mu_B(x)), \mu_C(x)) = S(\mu_A(x), S(\mu_B(x), \mu_C(x)))$ *(associativa)*

Qualquer *t-conorma* pode ser derivada a partir de sua associada *t-norma* como segue:

$$S(\mu_A(x), \mu_B(x)) = 1 - T(1-\mu_A(x), 1-\mu_B(x)) \quad (3.8)$$

A seguir, serão definidas algumas *t-normas* e *t-conormas* propostas na literatura e usadas no processo de inferência nebulosa. Elas serão utilizados ainda nos testes experimentais, de modo que fique, assim, demonstrada a sua aplicabilidade no projeto dos modelos nebulosos para a otimização de programações.

Os operadores *T* de Zadeh (ZADEH:1973), os mais populares na literatura, são definidos como:

$$T_M(\mu_A(x), \mu_B(x)) = \min(\mu_A(x), \mu_B(x)) \quad (3.9)$$

$$S_M(\mu_A(x), b) = \max(\mu_A(x), \mu_B(x)) \quad (3.10)$$

Goguem (WEBER:1983), Bandler et al. (BANDLER:1980) etc. propuseram o estudo de um conjunto de operadores *T* que foram chamados de probabilísticos, ou produto algébrico, definidos como:

$$T_P(\mu_A(x), \mu_B(x)) = \mu_A(x) . \mu_B(x) \quad (3.11)$$

$$S_P(\mu_A(x), \mu_B(x)) = \mu_A(x) + \mu_B(x) - \mu_A(x)\ \mu_B(x) \quad (3.12)$$

(MIZUMOTO:1989) e (ZIMMERMANN:1996) estudaram um conjunto de operadores chamados de produto e soma drásticos, definidos como :

$$T_D(\mu_A(x),\mu_B(x)) = \begin{cases} \min(\mu_A(x),\mu_B(x)) \text{ se } \max(\mu_A(x),\mu_B(x)) = 1, \\ 0, \text{ caso contrário.} \end{cases} \quad (3.13)$$

$$S_D(\mu_A(x),\mu_B(x)) = \begin{cases} \max(\mu_A(x),\mu_B(x)) \text{ se } \min(\mu_A(x),\mu_B(x)) = 0, \\ 1, \text{ caso contrário.} \end{cases} \quad (3.14)$$

Há também os operadores chamados de *Lukasiewicz*, estudados por Giles (GILES:1980) e outros, definidos como:

$$T_L(\mu_A(x),\mu_B(x)) = \max(0,\ \mu_A(x)+\mu_B(x)-1) \quad (3.15)$$

$$S_L(\mu_A(x),\mu_B(x)) = \min(1,\ \mu_A(x)+\mu_B(x)) \quad (3.16)$$

Gupta et al. (GUPTA:1991A), (GUPTA:1991B) e Zimmermann (ZIMMERMANN:1996) definiram os operadores *Hamacher* apresentados a seguir:

$$T_G(\mu_A(x),\mu_B(x)) = \frac{\mu_A(x)\cdot\mu_B(x)}{(\mu_A(x)+\mu_B(x)-\mu_A(x)\cdot\mu_B(x))} \quad (3.17)$$

$$S_G(\mu_A(x),\mu_B(x)) = \frac{(\mu_A(x)+\mu_B(x)-2\mu_A(x)\cdot\mu_B(x))}{(1-\mu_A(x)\cdot\mu_B(x))} \quad (3.18)$$

e Weber (WEBER:1983) e outros estudaram um conjunto de operadores T que são dados por:

$$T_W(\mu_A(x),\mu_B(x)) = \begin{cases} \mu_A(x) \text{ se } \mu_B(x) = 1, \\ \mu_B(x) \text{ se } \mu_A(x) = 1, \\ 0, \text{ caso contrário.} \end{cases} \quad (3.19)$$

$$S_W(\mu_A(x),\mu_B(x)) = \begin{cases} \mu_A(x) \text{ se } \mu_B(x) = 0, \\ \mu_B(x) \text{ se } \mu_A(x) = 0, \\ 0, \text{ caso contrário.} \end{cases} \quad (3.20)$$

sendo *Tw* e *Sw* os únicos que não são contínuos.

Hamacher (GUPTA:1991A) propôs um conjunto de operadores que são definidos como:

$$T_H(\mu_A(x),\mu_B(x)) = \frac{\lambda\ \mu_A(x)\cdot\mu_B(x)}{1-(1-\lambda)(\mu_A(x)+\mu_B(x)-\mu_A(x)\cdot\mu_B(x))} \quad (3.21)$$

$$S_H(\mu_A(x),\mu_B(x)) = \frac{\lambda\ (\mu_A(x)+\mu_B(x))+\mu_A(x)\cdot\mu_B(x)(1-2\lambda)}{\lambda+\mu_A(x)\cdot\mu_B(x)(1-\lambda)} \quad (3.22)$$

onde $\lambda \geq 0$;

Nota-se a relação entre todos os operadores anteriores:

Para $\lambda=0$, $T_H \rightarrow T_W$, e $S_H \rightarrow S_W$;
$\lambda=1$, $T_H = T_P$, e $S_H = S_P$; e
$\lambda \rightarrow \infty$, $T_H \rightarrow T_G$, e $S_H \rightarrow S_G$

que podem ser ordenados como segue:

$$T_D \leq T_L \leq T_P \leq T_G \leq T_M;$$
$$S_M \leq S_G \leq S_P \leq S_L \leq S_D;$$

Esta ordem implica qualquer conjunto nebuloso A e B em X, com valores de pertinência entre 0 e 1; se o operador de interseção for uma *t-norma,* ele ficará limitado pelo operador mínimo T_M e pelo produto drástico T_D. Caso seja uma *t-conorma*, vai estar limitado por um operador máximo S_M e por um soma drástico S_D.

Mesmo que as *t-normas* e *t-conormas* sejam definidas como operadores binários, podem ser estendidas, por suas propriedades associativas, a operações múltiplas, por exemplo, um operador de agregação tipo "and" para múltiplas restrições. A função de avaliação E para uma instância $d=(d_1,...,d_m)$ e restrições ($\Re_1$,..., $\Re_n$) pode ser escrita como:

$$E(d) = T(\mu_{\Re 1}(d),..., \mu_{\Re n}(d)), \tag{3.23}$$

sendo $\mu_{\Re i}(d)$ a função de pertinência da *i-ésima* restrição. Em (YAGER:1988) demonstrou-se que:

$$T(\mu_A(x), \mu_B(x)) \leq min(\mu_A(x), \mu_B(x)) \tag{3.24}$$

é válida para qualquer *t-norma* T e $\mu_A(x), \mu_B(x) \in [0,1]$. A partir disso e da lei associativa para as *t-normas,* para tomada de decisão de critérios múltiplas, o uso de *t-normas* com o operador "and" não permite compensação para uma satisfação fraca:

$$T(\mu_{A1}(x),..., \mu_{An}(x)) \leq min(\mu_{A1}(x),..., \mu_{An}(x)) \tag{3.25}$$

Para *t-conormas*, a seguinte função de agregação de estilo "or" $E'(d)$ é análoga às funções da *t-norma* "and",

$$E'(d) = T(\mu_{\Re 1}(d),..., \mu_{\Re n}(d)), \tag{3.26}$$

Similarmente:

$$S(\mu_A(x), \mu_B(x)) \geq min(\mu_A(x), \mu_B(x)) \tag{3.27}$$
$$S(\mu_{A1}(x),..., \mu_{An}(x)) \geq min(\mu_{A1}(x),..., \mu_{An}(x)), \tag{3.28}$$

são válidas para quaisquer *t-conorma* S e $\mu_A(x), \mu_B(x) \in [0,1]$, que ressalta a característica da agregação do tipo "or" das *t-conormas:* diferença de $E(d)$, que não permite a compensação de uma satisfação individual fraca, $E'(d)$ não permite a compensação de uma satisfação forte.

Resumindo, a representação da agregação de conjuntos nebulosos em tomada de decisão deve ser equivalente à agregação feita pelos humanos. Devido às regras representarem a visão humana do problema, a combinação de tal conhecimento deve ser realizada de uma forma comparável ao pensamento humano. O entendimento humano de operadores como "and" ou "or" não é estritamente lógico, mas situa-se entre estes dois extremos. Tais agregações deverão ser feitas com operadores compensatórios, alguns deles descritos logo em seguida.

3.5.5.2 Operadores Médios

Uma abordagem simples para agregar conjuntos nebulosos no contexto da tomada de decisões, poderia ser usando procedimentos de agregação, utilizados freqüentemente na teoria de decisões multi-critérios. Estes procedimentos são utilizados em relações conflitantes entre objetivos, quando é permitida, então, a compensação, cujo resultado estará entre o limite inferior mais otimista e o limite superior mais pessimista. Os operadores médios mapeiam entre o grau de pertinência mínimo e máximo dos conjuntos agregados, portanto, são modelos adequados para processos de agregação humana em ambientes de decisões e por terem apresentado também bons desempenhos empíricos (ZIMMERMANN:1983).

A partir da combinação dos operadores mínimo e máximo, mais a média aritmética, foram gerarados por Werners (ZIMMERMANN:1996) os operadores *fuzzy and* e *fuzzy or*, os quais permitem resultados muito bons comparados aos dados empíricos de Zimmermann et al. (ZIMMERMANN:1983).

Definição 3.14: Operadores *fuzzy and* e *fuzzy or*

Os operadores *fuzzy and* (O_{fuzzy_and}) e *fuzzy or* (O_{fuzzy_or}) são definidos a seguir (ZIMMERMANN:1996):

$$O_{fuzzy\_and}(\mu_A(x),\mu_B(x)) = \gamma \cdot \min(\mu_A(x),\mu_B(x)) + \frac{(1-\gamma)(\mu_A(x)+\mu_B(x))}{2} \quad (3.29)$$

$$O_{fuzzy\_or}(\mu_A(x),\mu_B(x)) = \gamma \cdot \max(\mu_A(x),\mu_B(x)) + \frac{(1-\gamma)(\mu_A(x)+\mu_B(x))}{2} \quad (3.30)$$

onde $\gamma \in [0,1]$

O parâmetro γ indica o grau de proximidade do significado lógico de "and" e "or", respectivamente. Para $\gamma = 1$, reduz-se ao operador máximo; para $\gamma = 0$, propicia a ambos a média aritmética.

Os operadores *fuzzy and* e *fuzzy or* indicam uma compensação fixa entre a lógica "and" e a lógica "or". A fim de descrever uma variedade de fenômenos em situações de decisão, Zimmermann et al. (ZIMMERMANN:1983) testaram um operador mais geral no sentido de que a compensação entre a interseção e a união fosse expressada por um parâmetro γ. Foi chamado de *compensatory and*, e é definido como segue:

Definição 3.15: Operadores *compensatory and*

O operador *compensatory and* são definidos, segundo (ZIMMERMANN:1983), (TURKSEN:1992) e (ZIMMERMANN:1996), como:

$$O_{comp\_and}(x) = \left(\prod_{i=1}^{m} \mu_i(x)\right)^{(1-\gamma)} \left(1 - \prod_{i=1}^{m} (1-\mu_i(x))\right)^{\gamma}, x \in X,\ 0 \le \gamma \le 1 \qquad (3.31)$$

Este operador é uma combinação entre o *produto algébrico* para a modelagem da lógica "and" e a *soma algébrica* para a modelagem do "or", para conjuntos nebulosos m, com valores de pertinência μ_i, i=1,2, ... , m.

Definição 3.16: Operadores Compensatórios Associativos (O_{ACOtan})

Klement E. P. et al. projetaram em (KLEMENT:1994) um operador especial com os requisitos necessários para a agregação de regras, combinando mais de uma variável a uma outra de um nível superior na hierarquia. Segundo os autores, suas propriedades associativas e sua agregação similar à usada pelo homem fazem deste operador um bom candidato para ser usado na agregação de conjuntos nebulosos nos ambientes de tomada de decisão. Os operadores são baseados na função tangente e definidos como segue:

$$O_{ACO\tan}(\mu_A(x), \mu_B(x)) = \frac{\arctan(\tan(\mu_A(x)\,\pi/2)\tan(\mu_B(x)\,\pi/2))}{2\pi} \qquad (3.32)$$

3.5.6 Restrições Nebulosas de Importâncias Diferentes

O exemplo 3.2 tem mostrado que, em aplicações da vida real, nem todos os critérios tem a mesma importância. Porém, é necessário introduzir restrições com prioridades relativas, quando são avaliadas e comparadas as instâncias. Informações adicionais sobre as preferências entre as restrições podem ser obtidas, considerando vetores de pesos que descrevem a importância relativa das restrições.

Yager e Fargier et al. (SLANY:1996) argumentaram que categorizar ou colocar pesos nos objetivos, pode ser possível, por ordenamento linear, ou intervalos, ou razão relativa, ou por um valor absoluto. Eles propuseram ordenar as restrições, fornecendo um grau de prioridade. Um coeficiente $w \in [0,1]$ é anexado a cada restrição, um alto valor de w indicaria uma comparação de maior importância. Estas propriedades são transformadas em graus de satisfação de restrições, da seguinte maneira,

$$\mu_{soft,w}(x_1,\ldots,x_k) = \max(1\text{-}w,\ \mu_{soft,w}(x_1,\ldots,x_k)) \qquad (3.33)$$

Isto verifica que *w* representa o grau pelo qual a restrição deve ser satisfeita. O número (1-*w*) mede até que ponto é possível violar uma restrição.

A abordagem de Yager é levemente diferente. Ele não usa números, mas um conjunto finito de símbolos ordenados para representar os pesos dos critérios e define a semântica do operador "¬" de uma maneira óbvia (por exemplo, ¬high = low, etc.). Tal ordem pode ser por exemplo:

$$\text{perfeito} \geq \text{muito alto} \geq \text{alto} \geq \text{medio} \geq \text{baixo} \geq \text{muito baixo} \geq \text{bem baixo}$$

A diferença entre os métodos de Yager e Fargier et al. é mais de natureza de terminológica do que semântica, como segue:

$$w_{FARGIER} = \begin{pmatrix} 0.3 \\ 0.2 \\ 1 \\ 0.8 \\ 0.1 \end{pmatrix} \qquad w_{YAGER} = \begin{pmatrix} Llow \\ very\_low \\ perfect \\ high \\ Lowest \end{pmatrix}$$

Uma descrição detalhada pode ser encontrada em (SLANY:1996) e (FARGIER:1993).

3.5.7 Critérios para a Seleção dos Parâmetros dos Modelos Nebulosos

A seleção dos parâmetros têm uma influência determinante na qualidade do modelo nebuloso (MN) (vide anexo A) no ambiente onde está sendo usado (PARK:1992) (SLIWINNSKA:1982) (NAKANISHI:1993) (CÓRDON:1997) como na sua implementação em *hardware* (COSTA:1995). Segundo nosso conhecimento, não existe um método formal que permita escolher a melhor combinação de parâmetros de um MN para uma determinada aplicação, a não ser a experimentação. Muitas vezes, os projetistas de sistemas de controle usam a composição "Max–Min" por causa de suas boas bases teóricas. Os teóricos da lógica nebulosa, por sua vez, consideram-na apropriada pelo fato de ela ser necessária nas aplicações (GUTIÉRREZ:1995).

Finalmente ressaltou-se que, para determinar os parâmetros, os procedimentos comumente empregados, segundo (WANG:1994), são:

- ***Seleção empírica:*** escolher os parâmetros que apresentam um comportamento melhor de acordo com o ambiente onde vão ser empregados.
- ***Base teórica:*** escolher aqueles operadores que satisfaçam um maior número de axiomas, levando em consideração a qualidade no seu desempenho.
- ***Eficiência computacional:*** escolher a estrutura do MN mais simples do ponto de vista computacional.
- ***Facilidade de adaptabilidade:*** uma forma de estabelecer os parâmetros externos do MN é usar técnicas de treinamento derivadas das redes neurais ou técnicas de otimização tais como algoritmos genéticos, *simulated annealing*, etc.

3.5.8 Descrição dos Modelos Nebulosos utilizados na Fase Experimental

Segundo este capítulo e anexo A, existe uma grande variedade de operadores e estruturas de MN. A fim de analisar o comportamento dos operadores nebulosos selecionados para a fase experimental, serão projetados diferentes MN. Neste trabalho, far-se-á uma adaptação de (CÓRDON:1997) onde um modelo MN pode denotar como uma MN(i,j,k,l) quádrupla, sendo i o operador T_i usado na interpretação do conectivo "and" nos antecedentes; j denota a interpretação da inferência I_j; k representa o conectivo "also" (caso k=0, usa-se o modo DA como método de defuzzificação, e sendo $k\neq0$ usa-se o modo AD, se k=1, utiliza-se uma *t-norma* "mínimo" como operador de agregação, se k=2, o conectivo "also" é representado por uma *t-conorma* "máximo" e se k=3 e k=4, utilizam-se operadores médios dos estudados na seção 3.5.5.2). Finalmente, o índice l refere-se ao método de defuzzificação usado pelo MN (vide anexo A, item A6.5, para descrição dos métodos de defuzzificação).

No capítulo 5, escolher-se-ão os MN's a serem projetados e definir-se-á a metodologia de comparação.

Capítulo 4

UM SISTEMA PARA A PROGRAMAÇÃO DINÂMICA DE TURNOS - SIPRODT

4.1 Introdução

Se variáveis atuais como flutuações ou mudanças fortuitas da demanda, produtividade, taxas contínuas de licenças ou permissões por doenças, diferem significativamente dos níveis esperados para um horizonte de tempo contínuo, deve-se, então, recalcular as requisições de força de trabalho para a alocação dos turnos e possivelmente o nível de operários do terminal. A programação dos turnos necessitará, então, de um contínuo ajuste para atender esta natureza de trabalho imprevisível, própria de um terminal, de modo a obter, para um número mínimo e suficiente de empregados, a satisfação de todas as restrições do sistema, com uma distribuição apropriada dos turnos e sem deixar de considerar as estatísticas coletadas até o momento do novo cálculo. Por exemplo, se a produtividade é muito mais baixa que a esperada, então resultará na falta de empregados, já que mais pessoas deveriam ser requesitadas nos turnos do que as fornecidas pela programação.

Da mesma forma, a disponibilidade de empregados poderia ser insuficiente, se as taxas contínuas de permissões e doenças aumentassem substancialmente. Absenteísmo dos empregados também afeta a distribuição da mão de obra ou desempenho do terminal.

Se tais mudanças ocorrem constantemente, ajustes necessários deverão ser realizados na programação dos turnos e no número de empregados do terminal; conseqüentemente mantém-se o nível de qualidade no serviço de atendimento do terminal.

Neste capítulo, é apresentada a proposta de um modelo de sistema de programações dinâmicas de turnos, SIPRODT, no processo de estiva em um terminal de contêineres, que, além de elaborar uma programação dos turnos, com um número mínimo de mão de obra e considerar as restrições pertinentes do sistema, permite realizar os ajustes necessários na programação, conforme haja flutuações da demanda de serviço.

4.2 Projeto de um Modelo de um Sistema para a Programação Dinâmica de Turnos (SIPRODT)

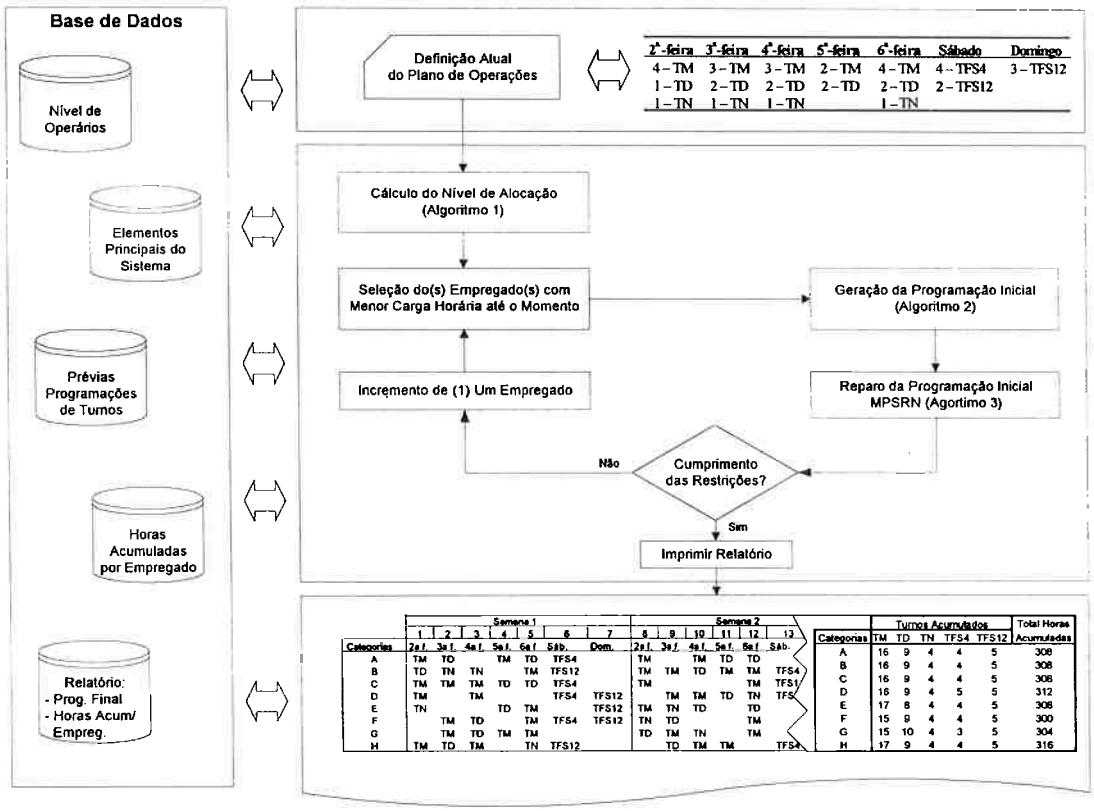


Figura 4.1 - SIPRODT: Sistema para a Programação Dinâmica de Turnos

A figura 4.1 apresenta a arquitetura de SIPRODT, um modelo de um sistema para a programação dinâmica de turnos que controla os ajustes necessários, devido à variabilidade que a demanda possa apresentar na programação dos turnos, regulando o número mínimo de empregados ou de categorias requeridas para a elaboração da programação e o número acumulado de horas trabalhadas por empregado, simultaneamente.

O programador humano define o plano de operações correspondente à semana a ser programada. O sistema carrega os diferentes tipos de dados de entrada necessários a partir de um conjunto de base de dados predefinidos para o terminal em estudo e determina o nível de alocação para o correspondente plano de operações (algoritmo 1). O algoritmo de otimização utilizado para a elaboração das programações é baseado nas técnicas de melhorias iterativas ou métodos por reparos, o que pressupõe a geração de uma programação inicial (algoritmo 2) e seu "reparo" (algoritmo 3), fundamentado em um problema de satisfação por restrições nebulosas, PSRN.

A base de dados é utilizada para armazenar e gerenciar os dados do sistema e dos empregados. A seguir, é feita uma descrição da mesma e na seqüência é apresentado passo a passo o procedimento de SIPRODT.

Nível de Operários

Armazena-se aqui o número total de empregados disponíveis de um terminal; isto varia de terminal para terminal e geralmente inclui os empregados permanentes e os empregados avulsos. O nível de escalação corresponde a um subconjunto do nível de operários e refere-se ao número mínimo de empregados necessários para servir um processo particular, no caso específico, atender o plano de operações da estiva de contêineres, deslocando os restantes empregados para outros processos ou tarefas do terminal.

Elementos Principais do Sistema

Definem-se como os elementos principais do sistema e, igualmente característico de cada terminal, tipos e extensão dos turnos, duração do ciclo e tipo de cobertura, seja de dias de semana e/ou finais de semana (vide seção 4.3). Estes dados são armazenados nesta base de dados e utilizados cada vez que o sistema o solicite.

Prévias Programações dos Turnos

As programações prévias dos turnos são relevantes no sistema, a fim de manter a continuidade das programações dos turnos e, com isto, gerenciar e distribuir uniformemente os diferentes tipos de turnos e, por conseguinte, os períodos de descanso, devido ao custo de possíveis acidentes, evitando com isso a fadiga do empregado. Armazena-se aqui as prévias programações dos turnos.

Horas Acumuladas por Empregado

Armazena-se aqui, para cada cálculo de uma programação final dos turnos, o número total acumulado por tipo de turno por empregado. Isto permite levar o registro histórico de horas trabalhadas por empregado, o que facilmente estabelecerá uma remuneração mais justa. Esta poderia ser diferenciada seja pelo o tipo de turno (caso de um incremento pelos turnos noturnos) seja por turnos cobrindo os finais de semana.

Relatório dos Resultados

Guardam-se aqui os relatórios dos resultados para cada plano de operação definido pelo programador humano, este inclui basicamente o número mínimo necessário de empregados, a sua programação final dos turnos preservando todas as restrições do sistema e as estatísticas dos tipos de turnos alocados por empregado de horas trabalhadas.

Estabelecidas as primeiras duas bases de dados nomeadas acima, que correspondem aos dados de entrada de um novo sistema (*Nível de Operários e Elementos Principais do Sistema)* para um determinado terminal, o programador

humano limita-se unicamente a definir, segundo as expectativas da demanda, o plano de operações em função de trabalhadores-turnos por dia para um horizonte de tempo correspondente a uma semana. Este prazo foi definido de acordo com a experiência em campo, até onde é possível, para um terminal, prever a movimentação de contêineres com uma semana de antecipação. As bases de dados restantes (*Prévias Programações dos Turnos, Horas Acumuladas por Empregado* e *Relatório dos Resultados*) são geradas e utilizadas pelo sistema.

Procedimento de SIPRODT

1ª Etapa: Inicialmente, definem-se os dados correspondentes ao nível de operários e elementos principais do sistema, dados esses, característicos de cada terminal (vide seção 4.3).

2ª Etapa: Estima-se o plano de operações do processo em estudo do terminal, definindo-se a requisição diária de força de trabalho, considerando o prognóstico da demanda de serviço para o período de uma semana (vide seção 4.4).

3ª Etapa: Definido o plano de operações na etapa anterior, estabelece-se aqui o nível de alocação ou número mínimo de trabalhadores para supri-la. O algoritmo 1 - definição da quantidade mínima dos níveis de escalação (vide seção 4.5) - responsabiliza-se por esta função.

4ª Etapa: A partir da base de dados *Nível de Operários* e de *Horas Acumuladas por Empregados,* seleciona-se o nível de escalação (calculado na etapa anterior) com menores horas de trabalho acumuladas até o momento. Este procedimento repete-se toda vez que for necessário efetuar-se uma seleção de empregados a partir do *Nível de Operários*.

5ª Etapa: Com os dados do plano de operações, dos elementos principais do sistema e do número mínimo da força de trabalho necessária para atender a demanda, o algoritmo 2 (vide seção 4.6) gera, construtivamente, uma programação inicial de turnos.

6ª Etapa: Nesta etapa a solução previamente construída é melhorada até uma solução ótima ou até que seja atingido um critério de parada definido. Para isto, o algoritmo 3 – Modelo de Programação de Turnos baseado no Problema de Satisfação por Restrições Nebulosas, MPSRN- (vide seção 4.7), "repara" a programação inicial de turnos, considerando desta vez as restrições características do sistema, tais como: horas de trabalho semanais permitidas; extensão mínima de ciclos contínuos de trabalho; folgas entre os diferentes tipos de turnos; restrições relacionadas com o trabalho dos finais de semana; distribuição uniforme dos turnos entre os empregados.

7ª Etapa: Caso seja encontrado uma programação que satisfaça todas as restrições do sistema, gera-se e armazena-se um arquivo com o relatório final dos resultados, incluindo a programação final e as estatísticas referentes ao registro de turnos e horas trabalhadas por empregado.

8ª Etapa: Caso não seja encontrada uma solução dentro dos critérios definidos acima, procede-se com o incremento de um empregado ao processo de "reparo" da programação de turnos. Este novo empregado será selecionado a partir da base de dados do *Nível de Operários* e conseqüentemente o sistema gerará uma nova programação inicial e em seguida a melhoria desta, como indicam respectivamente as etapas 5ª e 6ª. A 8ª etapa repete-se até que seja encontrada uma programação que satisfaça todas as restrições do sistema.

9ª Etapa: Se as variáveis atuais do sistema (produtividade, taxas contínuas de licenças, permissões por doenças e especialmente flutuações na demanda) diferirem significativamente dos níveis atuais para uma nova semana, dever-se-á ajustar as requisições da força de trabalho para uma nova programação dos turnos. Para tal, insere-se no sistema somente os valores da recente demanda através de um novo plano de operações, repetindo-se todo o procedimento do algoritmo a partir da 3ª etapa.

Definidas as etapas correspondentes ao modelo de um sistema de programação dinâmica de turnos (SIPRODT, figura 4.1), segue uma explicação mais abrangente de cada um de seus principais componentes.

4.3 Definição do Nível de Operários e dos Elementos Principais do Sistema

Um dos primeiros passos no projeto de uma programação de turnos é decidir sobre o total de trabalhadores disponíveis no terminal (empregados permanentes e suplementares ou avulsos) e quais serão os principais elementos do sistema, definidos neste caso como:

- Tipos de turnos (tais como turnos durante o dia, à tarde, à noite, etc.);
- Duração dos turnos (por exemplo, turnos de quatro, oito ou dez horas);
- Duração do ciclo da lista de tarefas (isto é, o número de semanas antes que seja repetida a lista);
- Tipo de cobertura, dias da semana e finais de semana (por exemplo, fazendo uma lista de turnos para o trabalho de final de semana, ou para turnos extras).

4.4 Definição do Plano de Operações

Pressupõe-se aqui já realizada a requisição diária de força de trabalho em termos de empregados-turnos para o terminal e processo em estudo. Para a definição desta requisição são necessárias as seguintes informações:

- O nível esperado de demanda por serviços de estiva, isto é, a movimentação de contêineres;
- O número e a duração dos turnos por dia;

- A produtividade média esperada dos guindastes, ou seja, o número de contêineres que eles movimentam por turno;
- A quantidade e o tipo de equipamento, isto é, o número de portêineres, *straddle carries*, *trailers*, empilhadeiras, transtêineres, etc.;
- número de trabalhadores por equipamento.

4.5 Definição dos Níveis de Escalação (Algoritmo 1)

Devido à dinâmica do problema, nem sempre é necessária a utilização do mesmo número de empregados para atender à demanda de serviço do processo em estudo. A fim de penalizar a ociosidade e contribuir para uma melhor distribuição e aproveitamento da mão de obra nos diferentes processos do terminal, nesta seção pretende-se encontrar o número mínimo de trabalhadores nos níveis de escalação dos turnos, para uma específica demanda de serviço representada por um definido plano de operações. Contudo, o objetivo geral do modelo proposto continua sendo o de determinar a programação completa dos turnos, com um número mínimo de empregados e cumprindo com as restrições do sistema. A quantidade mínima da força de trabalho é uma restrição que deve ser incluída no sistema principal proposto, para garantir custos mínimos.

A formulação usada para determinar a quantidade mínima dos níveis de escalação W, ou seja, os empregados necessários para uma demanda definida, é o resultado de uma adaptação dos trabalhos de Baker K. (BAKER:1974), Vohra R. (VOHRA:1987) e Alfares H. (ALFARES:1994).

Baker K. (BAKER:1974) formulou este problema por meio da programação linear inteira, em função de que cada empregado deveria trabalhar 5 dias por semana, e era permitido só pares consecutivos de folga, problema denominado na literatura de (5/7). A formulação é dada a seguir:

$$minimize\ Z = \sum_{j}^{7} c_j x_j \tag{4.1}$$

sujeito à seguinte restrição:

$$\left(\sum_{j=1}^{7}(x_j)\right) - x_j - x_{j-1} \geq r_j, \quad j = 1,\ldots,7 \quad onde\ x_{-1} = x_7 \tag{4.2}$$

ou

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \geq \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \\ r_7 \end{bmatrix} \tag{4.3}$$

onde $x_j \geq 0$, $(x_j \in \mathbb{N})$, $j = 1, \ldots, 7$

sendo x_j= número de empregados designados a um padrão de dias de folga j; c_j= custo semanal do padrão de dias de folga j por empregado, r_j = número de trabalhadores requeridos no dia j, e W = tamanho da força de trabalho = $x_1 + x_2 + ... + x_7$.

Porém, quando se assume c_j=1, i=1,...,7, o objetivo, equação (4.1), do modelo definido pelas equações (4.1) e (4.2), converte-se em minimização da força de trabalho W. O dual deste modelo é dado por:

$$maximize\, W = \sum_{i=1}^{7} r_i y_i \tag{4.4}$$

sujeito à seguinte restrição:

$$\left(\sum_{i=1}^{7} - y_i\right) - y_j - y_{j-1} \leq 1, \quad j = 1,\ldots,7 \tag{4.5}$$

onde,

y_i = variável dual, $y_i \geq 0$, $i = 1, 2, ..., 7$;

Uma solução ótima do problema referido anteriormente é resposta viável ao problema original de programação de turnos. Assim, o valor ótimo da função objetivo W é o mesmo para os dois problemas. Para resolver o problema dual, designou-se o recurso unitário (lado direito = 1) entre as variáveis, a fim de maximizar o objetivo dual W, que é uma combinação linear da demanda de trabalho. Têm-se aqui três possibilidades de soluções, dependendo da demanda de serviço dada r_1, ..., r_7:

1) A solução mais óbvia é designar o recurso unitário completo à máxima demanda, por exemplo, se r_k = max $\{r_1, ... r_7\}$, colocam-se y_k = 1 e todas as outras varáveis zero. Neste caso, $W = r_k = r_{max}$. Esta solução é melhor que a média, que poderia resultar quando designou-se um valor de $1/m$ a qualquer m variável, onde m = 2,...,7. No entanto, a estrutura do problema permite alocar um valor de $1/m$ a mais que m variáveis. Somente há apenas dois casos onde isto é possível, que serão discutidos na seqüência.

2) Devido ao fato de que a restrição da equação (4.5) contém apenas cinco variáveis duais, é possível dividir o lado direito de cada restrição por estas cinco variáveis. Desta maneira, é viável designar um valor de 1/5 a todas as sete variáveis. Neste caso, o tamanho da força de trabalho $W = \frac{1}{5}\sum_{d=1}^{7} r_d$ ou $W = \frac{7}{5}\bar{r}$, onde $\bar{r}$ = demanda média de serviço.

3) Uma situação similar existe para os sete conjuntos das quatro variáveis duais, cujos sobrescritos, denotados por S_i^*, são mostrados na tabela 4.1. Devido ao fato de que S_i^*=mod7$\{i, i+1, i+3, i+5\}$, i = 1, ..., 7, pelo menos uma de quaisquer duas variáveis sucessivas deve pertencer ao conjunto i. Portanto, pelo menos uma das variáveis sucessivas y_j e y_{j+1}, ausente na restrição (4.5), deve pertencer ao conjunto i. Por exemplo, se as variáveis j=i então y_j e y_{j+1} estão ausentes da restrição j; se j=i+1, então y_{j+1} está ausente; se j=i+2, então y_{j+3} está ausente; e assim por diante. Deste modo, o máximo das três variáveis a partir de cada conjunto estão presentes em cada restrição, tornando-se possível designar o valor de 1/3 para todas as quatro

variáveis em um dado conjunto. Neste caso, poder-se-ia escolher o conjunto tendo o máximo da demanda total; então, $W = \frac{1}{3}\max_i \left\{\sum_{j \in S_i^*} r_j\right\}$ ou $W = \frac{4}{3}\bar{r}_i$, onde $\bar{r}$ =demanda média entre os quatro dias pertencentes ao conjunto i.

Tabela 4.1 – Conjunto de 3 Números não adjacentes S_j e seu Complemento S_j^*, j =1,...,7

j	S_j	S_i^*
1	{3,5,7}	{1,2,4,6}
2	{1,4,6}	{2,3,5,7}
3	{2,5,7}	{1,3,4,6}
4	{1,3,6}	{2,4,5,7}
5	{2,4,7}	{1,3,5,6}
6	{1,3,5}	{2,4,6,7}
7	{2,4,6}	{1,3,5,7}

Para determinar o número de empregados, tem de se escolher o valor máximo de W, obtido a partir dos três casos descritos acima e aproximá-lo a um número inteiro, caso não seja. Finalmente, chegou-se à seguinte expressão para o W mínimo, coincidindo com o resultado encontrado por Vohra (VOHRA:1987):

$$W = \sum_{j=1}^{7} x_j = \max\left\{r_{\max}, \left[\frac{1}{5}\sum_{i=1}^{7} r_i\right], \left[\frac{R_{\max}}{3}\right]\right\} \tag{4.6}$$

onde,

$$R_i = \sum_{j \in S_i^*} x_j, \quad i = 1, 2, \ldots, 7 \tag{4.7}$$

Segundo Baker (BAKER:1985), a força mínima de trabalho W é dada por:

$$W \geq W' + \max\left\{0, \left[\left(-x_{\min} - \frac{t}{2}\right)\Big/ 3\right]\right\} \tag{4.8}$$

onde,

$$W' = \max\left\{r_{\max}, \left[\frac{1}{5}\sum_{i=1}^{7} r_i\right]\right\} \tag{4.9}$$

$$t = 5W' - \sum_{j=1}^{7} r_j \tag{4.10}$$

$$x_i = W' - \sum_{j \in S_i} b'_j, \quad i = 1, 2, \ldots, 7 \tag{4.11}$$

S_i= complemento de S_i^*, mostrado na tabela 4.1, i= 1, 2, ... ,7.

De fato, a fórmula correta para o limite inferior do tamanho da força de trabalho W, demonstrado por Alfares (ALFARES:1994), é:

$$W \geq W' + \max\{0, [(-x_{\min} - t)/3]\} \tag{4.12}$$

Os argumentos da equação (4.9) são os dois mesmos primeiros argumentos da equação (4.6). O que indica de forma clara que a formulação apresentada na equação (4.12) corresponde ao terceiro argumento da equação (4.6). Para facilitar a comparação, a equação (4.12) pode ser escrita como:

$$W \geq \max\{W', W' + [(-x_{\min} - t)/3]\} \tag{4.13}$$

ou

$$W \geq \max\left\{r_{\max}, \left[\frac{1}{5}\sum_{i=1}^{7} r_i\right], [W' + (-x_{\min} - t)/3]\right\} \tag{4.14}$$

Sabendo que $b_i = W - r_i$ e usando a equação (4.11) têm-se :

$$x_i = W' - \sum_{j \in S_i}(W' - r_j) = W' - 3W' + \sum_{j \in S_1} r_j = \sum_{j \in S_1} r_j - 2W', \quad i = 1, 2, \ldots, 7 \tag{4.15}$$

Deste modo,

$$x_{\min} = \min\left\{\sum_{j \in S_1} r_j - 2W'\right\} = \min\left\{\sum_{j \in S_1} r_j\right\} - 2W', \quad i = 1, 2, \ldots, 7 \tag{4.16}$$

Substituindo os valores de t e $x_{\min}$ nas equações (4.10) e (4.16), o terceiro argumento da equação (4.14) converter-se-á :

$$\begin{aligned}
[W' + (-x_{\min} - t)/3] &= \left[W' + \frac{1}{3}\left(-\min\left\{\sum_{j \in S_1} r_j\right\} + 2W' - 5W' + \sum_{j=1}^{7} r_j\right)\right] \\
&= \left[W' + \frac{1}{3}\left(-\max\left\{\sum_{j \in S_i^*} r_j\right\} - 3W'\right)\right] \\
&= \left[\frac{1}{3}\max\left\{\sum_{j \in S_i^*} r_j\right\} = \frac{1}{3} R_{\max}\right]
\end{aligned} \tag{4.17}$$

Posto que isto é igual ao terceiro argumento da equação (4.6), o limite inferior obtido pelo algoritmo de Baker (BAKER:1974) é igual ao número mínimo da força de trabalho W.

Procedimento do Algoritmo 1

De acordo com a literatura, concluiu-se que há dois grandes métodos disponíveis para resolver o problema do número mínimo necessário de trabalhadores, em que a demanda é variável durante os dias de semana. O primeiro método é a partir da programação linear inteira, geralmente ineficiente. O segundo método é a solução por meio de um modelo de programação linear proposta por Bartholdi et al. (BARTHOLDI:1980), em que é necessário, com freqüência, resolver três modelos de programação linear contínuos para cada problema. Todavia, a solução proposta por Baker e Vohra, usando a equação (4.6), fornece um métodos simples para encontrar W. Todas as etapas para o cálculo do número mínimo de empregados para uma definida demanda estão representados na figura 4.2. No capítulo 5 será descrito um exemplo de aplicação deste algoritmo correspondente a um processo de estiva ideal em um terminal de contêineres.

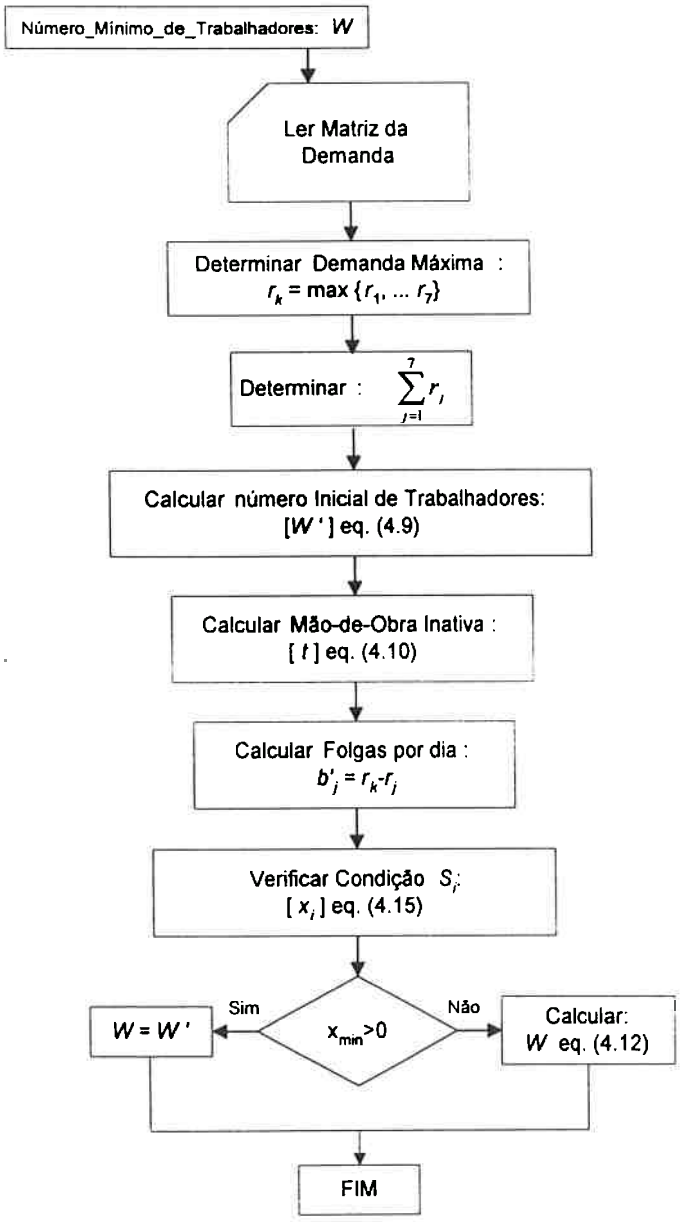


Figura 4.2 – Algoritmo 1: Cálculo do Número Mínimo dos Níveis de Escalação

4.6 Geração da Programação Inicial (Algoritmo 2)

Devido ao fato de o método de otimização por reparos depender de uma solução inicial, será preciso então gerá-la. Segundo Dorn et al. (DORN:1994) a qualidade da solução inicial, em função do tempo, não faz muita diferença para se encontrar uma "boa" solução, independente da heurística de reparo a ser utilizada.

Os diferentes parâmetros necessários para criar-se uma solução inicial são: (a) a definição dos elementos principais do sistema (número, tipo e extensão dos turnos) (seção 4.3); (b) o número de trabalhadores/categorias (algoritmo 1, seção 4.5); e (c) a definição do plano de operação (seção 4.4). Levando-se em consideração os dados anteriores projetou-se a heurística pela qual gera-se essa programação inicial que está representada nas figuras 4.3 e 4.4. Este algoritmo baseia-se principalmente na designação dos turnos ao empregado que até o momento da alocação apresente o menor tempo de horas trabalhadas e paralelamente cumpra com o número total de turnos e com a restrição exigida por dia (restrições *hards*), para esse mesmo tipo de turno (empregado com mais folga). Para isto, são necessários alguns pré-cálculos de modo a definirem-se o número e o tipo de turnos que deverão ser alocados e o total de horas trabalhadas, incluindo-se todos os turnos por empregado. Estes dois critérios servirão de base para a designação de um determinado turno a um específico empregado/categoria.

No capítulo 5, serão apresentados os dados e a solução correspondente a vários exemplos de aplicação. A seguir serão ampliadas estas informações bem como a heurística utilizada.

Procedimento do Algoritmo 2

1ª Etapa: Lê-se da base de dados a extensão completa do ciclo dos turnos, o número-tipo e duração dos turnos correspondente à demanda de turnos para cada dia da semana e o número de trabalhadores, calculado pelo algoritmo 2.

2ª Etapa: Começa-se a designação dos turnos inicialmente a partir dos correspondentes dos finais de semana, para todos os tipos de turnos (*turnos*), todos os dias da extensão do ciclo (*dias*) e todos os trabalhadores exigidos na demanda (*k*). (Vide figura 4.3)

3ª Etapa: Para cada designação, a função "Classifica ( )" (vide figura 4.4) define quem dos empregados possui até o momento o menor número de horas trabalhadas. Para isto, cria-se uma lista definida como "folgados", onde se classificará os empregados de acordo com dois critérios que serão definidos e calculados na próxima etapa.

4ª Etapa: Para o número de empregados e total de dias do ciclo, calcula-se dos critérios para a escolha do empregado com mais folga, a quantidade de turnos para cada tipo de turno por empregado (*nshift*) e o total de horas trabalhadas incluindo-se todos os tipos de turnos, também por empregado (*hworked*). (Vide figura 4.4)

5ª Etapa: Para todos os empregados da lista dos "folgados" verificam-se os critérios acima nomeados, isto é, que o trabalhador corrente da lista [i] dos "folgados" tenha uma quantidade de turnos em referência menor que o selecionado (*livre*) ou que o total de

horas trabalhadas de todos os turnos seja menor que o selecionado (*livre*). (Vide figura 4.4)

6ª Etapa: Aloca-se o empregado da lista dos "folgados" para essa designação para que ele cumpra com os critérios anteriormente estabelecidos. Volta-se e repete-se o processo a partir da 2ª etapa. (Vide figura 4.3 e 4.4)

7ª Etapa: O processo assim se repete até que se complete o número de turnos especificados, para todos os dias da programação e todos os trabalhadores exigidos pela demanda como é indicado na 2ª etapa.

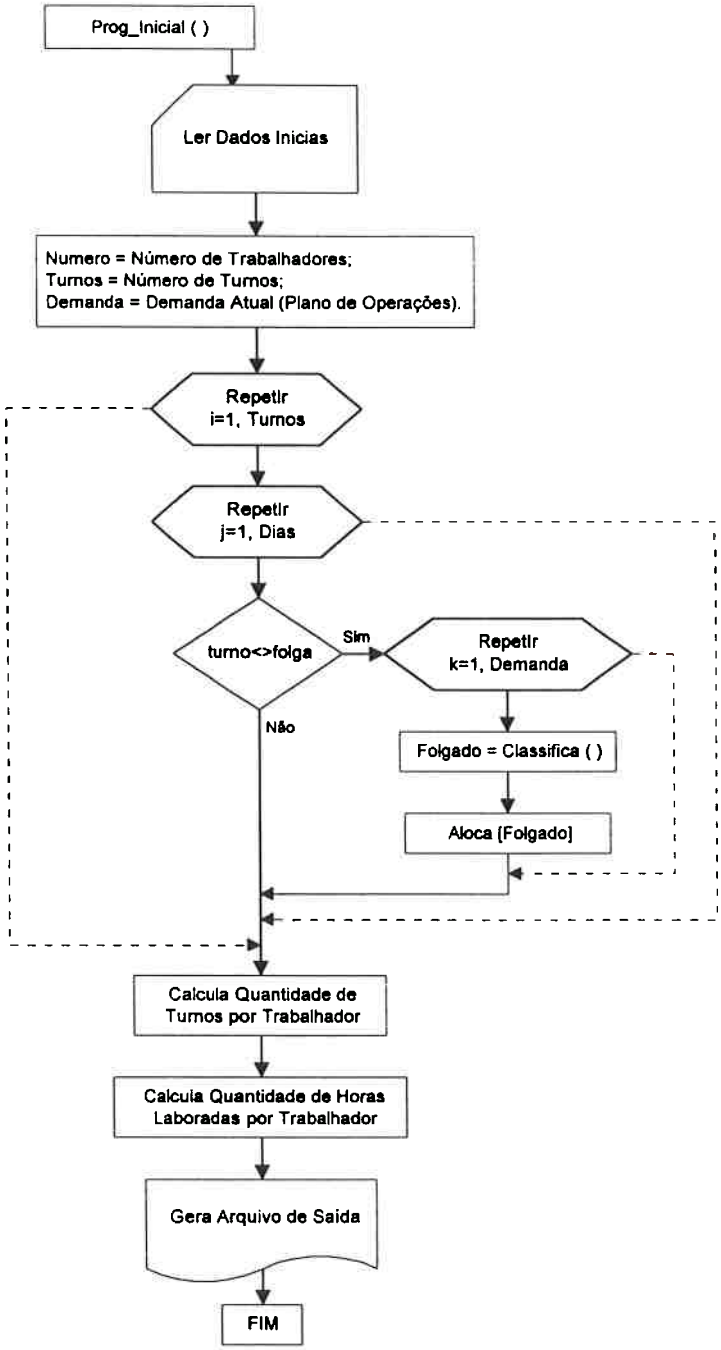


Figura 4.3 – Algoritmo 2: Geração da Programação Inicial

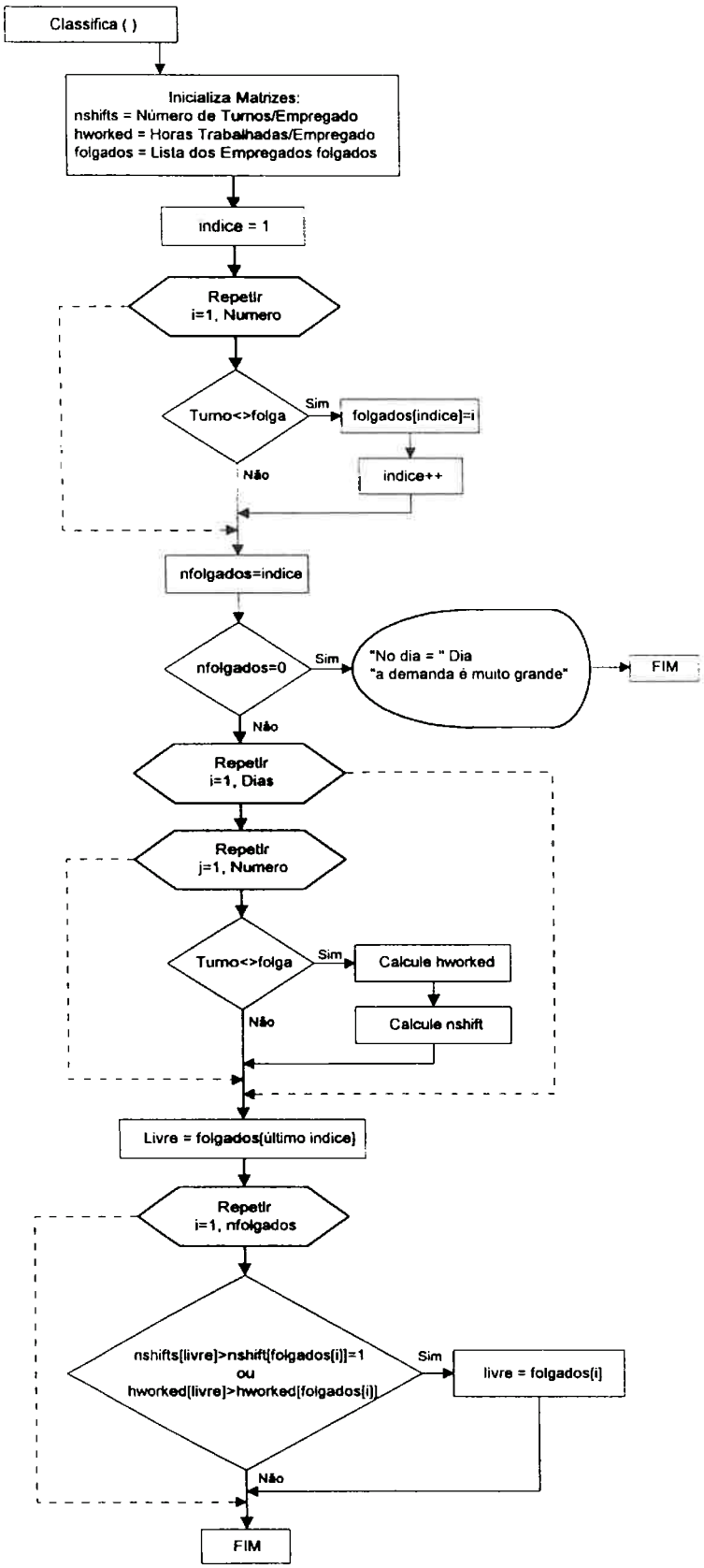


Figura 4.4 – Algoritmo 2: Geração da Programação Inicial – Função "Classifica( )"

4.7 Modelo de Programações de Turnos baseado em um Problema de Satisfação por Restrições Nebulosas - MPSRN (Algoritmo 3)

As restrições de nossa aplicação são de natureza dinâmica, isto quer dizer que a instância concreta depende da instância corrente do problema. Em outra palavras, isto significa que programações de turnos diferentes conduzem a diferentes instâncias de restrições do mesmo tipo. Portanto, é necessário definir estas restrições em uma estrutura de linguagem baseada no uso de variáveis (vide figura 4.5 e 4.6), onde em cada iteração, a avaliação deverá estar fundamentada nestas variáveis cujos valores são alimentados pela instância do problema. Esta metodologia foi baseada nos trabalhos de Dorn, (DORN:1992), (DORN:1994), Slany (SLANY:1996), Santa et al. (SANTA:1996) e Reichel et al. (REICHEL:1997).

Seguindo a heurística das figuras 4.5 e 4.6 explicar-se-á cada um dos processos que conformam o modelo de programações de turnos por restrições nebulosas iniciando pela definição dos requisitos do sistema. No capítulo 5, serão apresentados os resultados correspondentes à aplicação deste algoritmo e os dados de um estudo de caso.

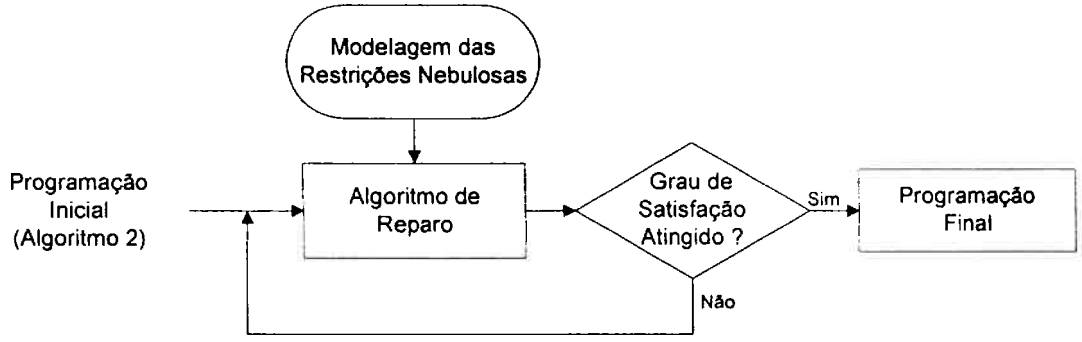


Figura 4.5- Algoritmo 3: Modelo MPSRN

4.7.1 Definição de Requisitos de Sistema

Durante o processo da programação de turnos, os requisitos a serem considerados podem se tornar numerosos. Primeiramente, existem os requisitos técnicos ergonômicos que devem ser satisfatórios para um boa programação de turnos. Em seguida, têm-se as restrições legais decorrentes de acordos coletivos, os quais colocam a programação de turnos dentro de uma estrutura mais ou menos restritiva. E, finalmente, os requisitos sociais, os quais também devem ser considerados. Abrangendo os diferentes grupos e interesses mencionados inicialmente, mais os requisitos não técnicos, é fácil imaginar as inúmeras restrições encontradas numa programação de turnos. Muitos destes requisitos são conflitantes entre si, complicando ainda mais o cenário inicial. Mesmo que um grande número de requisitos reduza o espaço da solução (tentando assegurar a quantidade de possíveis soluções, imagine somente todas as combinações possíveis de formação de turnos num período de um ano, com vários tipos de turnos), ele pode induzir a espaços de busca com poucas soluções admissíveis.

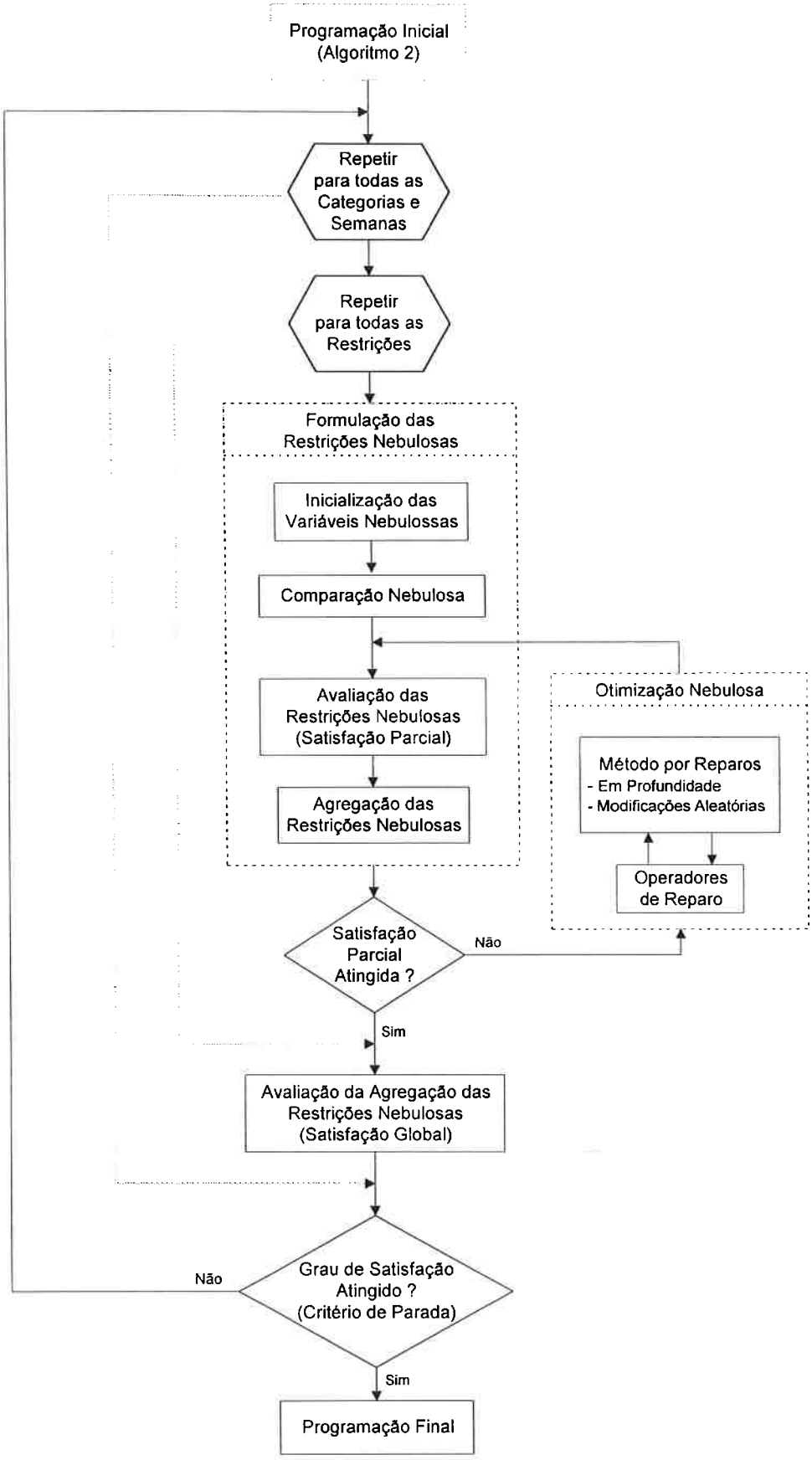


Figura 4.6 - Algoritmo 3: Detalhe do modelo MPSRN

Uma vez que os requisitos mais técnicos, tais como os ergonômicos e as restrições legais, possam ser modelados formalmente, mesmo havendo certas dificuldades, os requisitos sociais não permitem a modelagem numa linguagem formal devido à sua natureza individual e muitas vezes subjetiva. Consequentemente, este trabalho estará centrado nos aspectos técnicos, os quais serão apresentados a partir da próxima seção. A representação das restrições não técnicas dentro do sistema do computador deverá ser uma área de futura investigação, especialmente com a chegada de sistemas e máquinas mais eficientes.

4.7.2 Formulação das Restrições Nebulosas

Seguindo a heurística do modelo proposto, figuras 4.5 e 4.6, plasma-se a modelagem das restrições para nossa aplicação em uma estrutura de linguagem baseada no uso de variáveis, onde, para cada iteração, a avaliação deverá estar baseada em valores alimentados pela instância do problema em tratamento. Devido ao fato de cada restrição ter sua particular definição, estas conduzem a derivações específicas para as recomendações de início dos reparos dos algoritmos de otimização, devendo-se então também formular estes inícios de reparo para cada restrição. Algumas restrições encontradas no domínio do sistema em estudo, ou seja, *Terminal Marítimo de Contêineres*, serão definidas na seqüência.

4.7.2.1 Restrição de Distribuição Uniforme de Horas de Trabalho *(DiferençaHoras)*

Esta restrição tenta orientar a avaliação de uma programação uniforme dos turnos, fazendo com que os desvios entre as horas de trabalho semanais sejam relativamente pequenos. Uma vez realizado o cálculo da diferença de horas trabalhadas entre a semana seguinte e a corrente, detecta-se aqueles pares de semana cuja diferença é alta demais. Portanto, sua correção deverá conduzir a uma seqüência de turnos distribuída mais uniformemente e, por conseguinte, de horas trabalhadas. Esta restrição pode ser formulada, para um número definido de empregados M e uma extensão do ciclo em diferenças de semanas (N-1), como mostra a figura 4.7 e que será explicado na continuação, para as restantes restrições segue-se um procedimento equivalente.

```
/* RESTRIÇÃO DiferençaHoras */
FOR i = 1 TO M STEP 1
        FOR j = 1 TO (N-1) STEP 1
                // inicialize variável nebulosa com o cálculo das diferença de horas entre semanas
                INICIALIZE DiferençaHoras = DifHoras [i][j]
                // formule a restrição,
                COMPARAÇÃO NEBULOSA NívelSatisfação IS DiferençaHoras = 0.0
                // avalie a restrição
                AVALIE NívelSatisfação
                // insira a restrição na árvore de hierarquia de restrições, ramal : DIFERENÇAHORAS
                AGREGAÇÃO DIFERENÇAHORAS AND NívelSatisfação empregado[i].semana[j]
        NEXT j
NEXT i
```

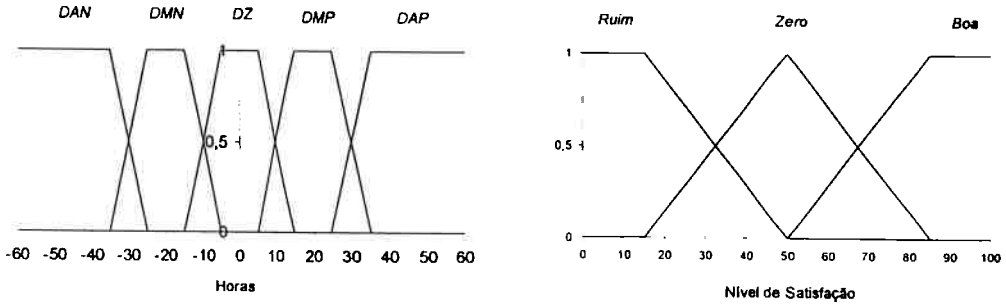
Figura 4.7 – Formulação da Restrição *DiferençaHoras*

Em INICIALIZE designa-se um valor *crisp* (*DifHoras*[i][j]), calculado pela diferença entre a semana seguinte e a corrente, a uma variável lingüística (*DiferençaHoras*) representada segundo a teoria possibilística (vide item A3 do Anexo A) por seus números e termos nebulosos conforme o mostrado na figura 4.8, dando inicio ao processo de fuzzificação para cada valor de entrada [i][j].

Na COMPARAÇÃO NEBULOSA se formula a restrição nebulosa, comparando-se a variável nebulosa (*DiferençaHoras*) a um valor *crisp* (0.0), a fim de estabelecer, no próximo passo, o grau ou nível de satisfação do valor *crisp* inicializado na etapa anterior. Para isto tem que ser definido o operador de comparação, no caso {=} (que pode ser também {<,>}, veja a formulação das próximas restrições) e como cada termo lingüístico se corresponde quando se realiza esta comparação. As regras definidas para esta restrição estão descritas na tabela 4.2. O objetivo da comparação é "capturar" incertezas, em que o operador {=} não deve ser interpretado no sentido matemático, mas como "pequenas" violações aceitáveis. Estas "pequenas" violações poderão ser definidas explícita e precisamente por meio das funções de pertinência associadas aos termos das variáveis lingüísticas como é definido na figura 4.8

Em AVALIE inicializa-se o processo de converter o valor nebuloso (*NívelSatisfação)* a um valor *crisp* (defuzzificação) determinando-se então o grau de satisfação. O método mais freqüentemente usado é o baseado no cálculo do centro de gravidade (*CG*) da função de pertinência do conjunto nebuloso resultante da comparação nebulosa (outros métodos são descritos na seção A6.5). Esta avaliação é possível de acordo com a definição das regras para cada restrição e também na seleção de seus respectivos operadores nebulosos (estudados na seção 3.5.5).

No processo de AGREGAÇÃO insere-se o valor da variável nebulosa (*NívelSatisfação*), avaliada no passo anterior, a uma estrutura hierárquica de restrições, usando-se o operador *minimum*. Esta técnica é repetida para todas as restrições avaliadas com o fim de agregá-las para uma segunda avaliação, obtendo-se assim uma lista das maiores violações ou conflitos do conjunto de restrições. Dependendo dos operadores de modificações dos algoritmos de reparo, seria útil encontrar não somente o maior conflito e sim o segundo e terceiro maior da lista. Geralmente, este processo retorna a maior violação, sendo de um tipo que possa ser manuseada por um passo reparo disponível, a ser visto na seção 4.7.3.2 em maiores detalhes.



onde DAP = *diferença_alta_positiva* DMP = *diferença_media_positiva*
DZ = *diferença_zero* DMN = *diferença_media_negativa*
DAN = *diferença_alta_negativa*

Figura 4.8 - Representação Qualitativa da Restrição *DiferençaHoras*

Tabela 4.2 – Regras de Inferência para a Comparação Nebulosa da Restrição *DiferençaHoras*

1. SE *DiferençaHoras* É *diferença_alta_positiva*	ENTÃO	*NívelSatisfação* É	*ruim.*
2. SE *DiferençaHoras* É *diferença_media_positiva*	ENTÃO	*NívelSatisfação* É	*zero.*
3. SE *DiferençaHoras* É *diferença_zero*	ENTÃO	*NívelSatisfação* É	*boa.*
4. SE *DiferençaHoras* É *diferença_media_negativa*	ENTÃO	*NívelSatisfação* É	*zero.*
5. SE *DiferençaHoras* É *diferença_alta_negativa*	ENTÃO	*NívelSatisfação* É	*ruim.*

E por último deverá também definir-se as recomendações de início de reparo para esta restrição que servirão, por sua vez, de guia para os operadores das modificações. Para o caso específico da restrição *DiferençaHoras* escolha-se aleatoriamente entre os dias da semana em conflito.

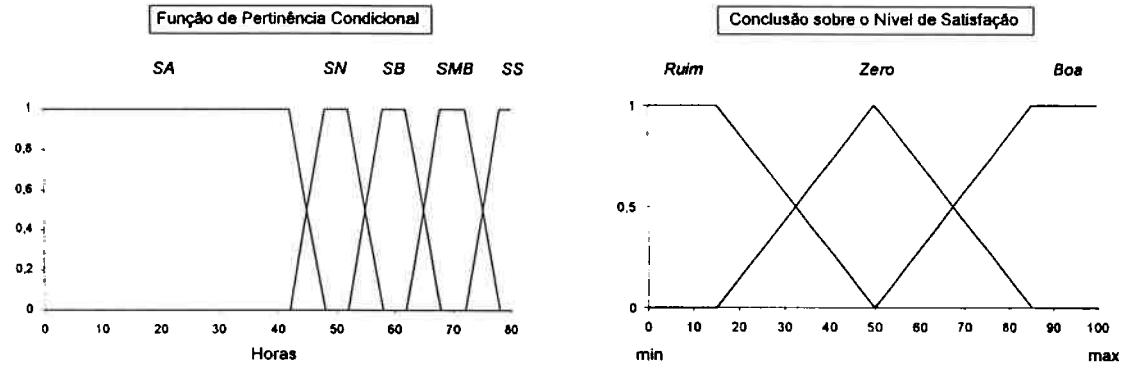
4.7.2.2 Restrição de Horas Trabalhadas (*HorasTrabalho*)

Segundo as leis e acordos sindicato-empresas, sobre arranjo de pessoal no terminal, um empregado semanalmente não deve exceder uma média de 50 horas por semana. É necessário, então, calcular o valor da carga horária de cada empregado por semana para a extensão completa do ciclo a fim de possibilitar a avaliação desta restrição. A formulação da restrição segue a mesma estrutura utilizada para a definição da restrição anterior, para um número de empregados M e uma extensão completa do ciclo em semanas N, como se mostra na figura 4.9

```
/* RESTRIÇÃO HorasTrabalho */
FOR i = 1 TO M STEP 1
        FOR j = 1 TO N STEP 1
                // inicialize variável nebulosa com o cálculo das horas trabalhadas
                INICIALIZE HorasTrabalho = HorasTrab [i][j]
                // formule a restrição,
                COMPARAÇÃO NEBULOSA NivelSatisfação IS HorasTrabalho < 50
                // avalie a restrição
                AVALIE NivelSatisfação
                // insira a restrição na árvore de hierarquia de restrições, ramal : HORASTRABALHO
                AGREGAÇÃO HORASTRABALHO AND NivelSatisfação empregado[i].semana[j]
        NEXT j
NEXT i
```

Figura 4.9 – Formulação da Restrição *HorasTrabalho*

A variável nebulosa *HorasTrabalho* possui os seguintes termos lingüísticos: *satisfação_alta, satisfação_normal, satisfação_baixa, satisfação_muito_baixa,* e *sem_satisfação*, e é representada na figura 4.10, da mesma forma, as regras de inferência para avaliação desta restrição, na tabela 4.3. As recomendações para os inícios dos reparos para esta restrição poderá ser em qualquer um dos dias, começando do primeiro e variando de segunda a domingo da semana em conflito.



onde SS = *sem_satisfação* SMB = *satisfação_muito_baixa*
SB = *satisfação_baixa* SN = *satisfação_normal*
SA = *satisfação_alta*

Figura 4.10 - Representação Qualitativa da Restrição *HorasTrabalho*

Tabela 4.3 – Regras de Inferência para a Comparação Nebulosa da Restrição *HorasTrabalho*

1. SE *HorasTrabalho*	É	*sem_satisfação*	ENTÃO *NívelSatisfação* É	*ruim.*
2. SE *HorasTrabalho*	É	*satisfação_muito_baixa*	ENTÃO *NívelSatisfação* É	*ruim.*
3. SE *HorasTrabalho*	É	*satisfação_baixa*	ENTÃO *NívelSatisfação* É	*ruim.*
4. SE *HorasTrabalho*	É	*satisfação_normal*	ENTÃO *NívelSatisfação* É	*zero.*
5. SE *HorasTrabalho*	É	*satisfação_alta*	ENTÃO *NívelSatisfação* É	*zero.*

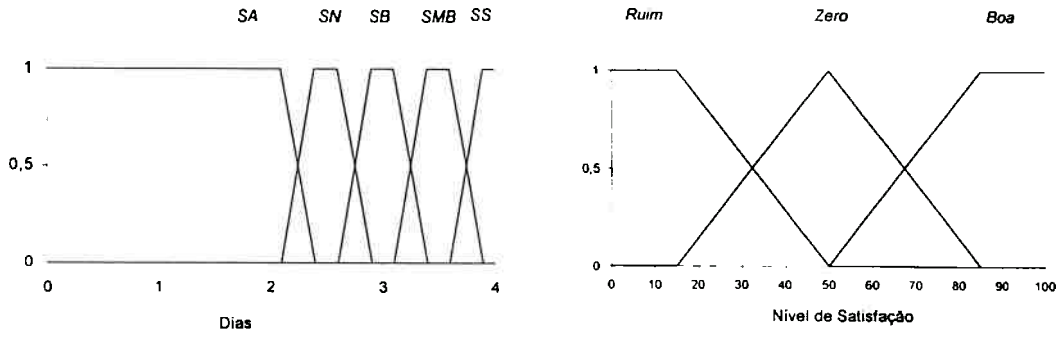
4.7.2.3 Restrição para os Finais de Semana *(FinaisSemana)*

Finais de semana é tópico de grande importância na maioria dos problemas de programação de turnos e, de acordo com a literatura, tem se dado pouca importância. Nesta aplicação trata-se a avaliação dos finais de semana de forma a melhorar a qualidade da programação. A restrição tenta evitar designações de turnos para operários de uma mesma categoria sobre os dias consecutivos em dois finais de semanas seguidos retornando uma avaliação "ruim" para alocações próximas de 4 dias. De igual maneira às restrições anteriores, formulou-se esta para um predefinido número de empregados M e para a extensão do ciclo em pares de finais de semana consecutivos (N-1) como mostra a figura 4.11

```
/* RESTRIÇÃO FinaisSemana */
FOR i = 1 TO M STEP 1
        FOR j = 1 TO (N-1) STEP 1
                // inicialize variável nebulosa com o cálculo dos dias trabalhados durante dois finais de
                // semana consecutivos
                INICIALIZE FinaisSemana = FinSemana [i][j]
                // formule a restrição,
                COMPARAÇÃO NEBULOSA: NívelSatisfação IS FinaisSemana < 4
                // avalie a restrição
                AVALIE NívelSatisfação
                // insira a restrição na árvore de hierarquia de restrições, ramal : FINAISSEMANA
                AGREGAÇÃO FINAISSEMANA AND NívelSatisfação empregado[i].semana[j]
        NEXT j
NEXT i
```

Figura 4.11 – Formulação da Restrição *FinaisSemana*

As funções de pertinências da variável lingüística *FinaisSemana* e seus valores lingüísticos estão representados na figura 4.12. O conjunto de regras de inferência estão descritos na tabela 4.4. As recomendações de início para os reparos começam a partir do primeiro dia do primeiro final de semana e variam até o quarto dia do segundo final de semana do par de finais de semana em conflito.



onde SS = *sem_satisfação* SMB = *satisfação_muito_baixa*
SB = *satisfação_baixa* SN = *satisfação_normal*
SA = *satisfação_alta*

Figura 4.12 - Representação Qualitativa da Restrição *FinaisSemana*

Tabela 4.4 – Regras de Inferência para a Comparação Nebulosa da Restrição *FinaisSemana*

1. SE *FinaisSemana*	É *sem_satisfação*	ENTÃO *NívelSatisfação* É	*ruim.*
2. SE *FinaisSemana*	É *satisfação_muito_baixa*	ENTÃO *NívelSatisfação* É	*ruim.*
3. SE *FinaisSemana*	É *satisfação_baixa*	ENTÃO *NívelSatisfação* É	*ruim.*
4. SE *FinaisSemana*	É *satisfação_normal*	ENTÃO *NívelSatisfação* É	*zero.*
5. SE *FinaisSemana*	É *satisfação_alta*	ENTÃO *NívelSatisfação* É	*zero.*

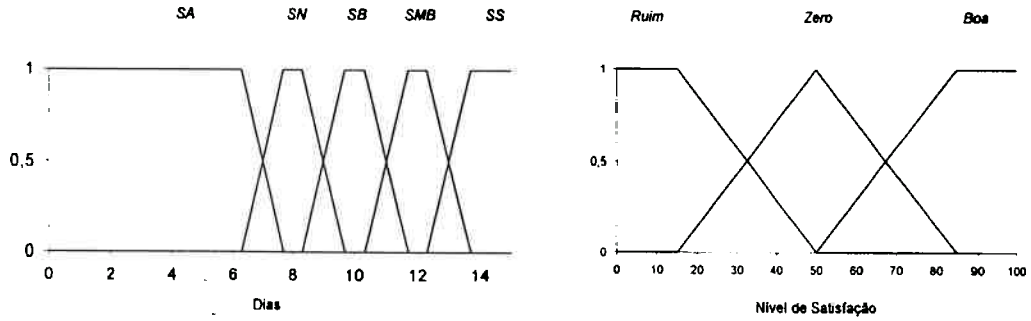
4.7.2.4 Restrição para Evitar Seqüências de Turnos Excedentes *(SeqüênciaTurnos)*

As pesquisas em programação de turnos têm demonstrado que devem ser evitadas seqüências de turnos acima de seis dias consecutivos para um mesmo empregado. Após vários turnos consecutivos, os períodos de descanso deverão ser cada vez mais importantes. Dever-se-á então detectar através de uma revisão da atual programação dos turnos, o número de blocos de seqüências de turnos - sem folga - para cada um dos empregados e seus tamanhos. A restrição deverá produzir avaliações "ruins" para tamanhos de seqüências de seis ou mais turnos, seguindo a formulação pela já conhecida estrutura para um número pre-estabelecido de empregados M e o número dos blocos de seqüências de turnos B, como é definido na figura 4.13.

```
/* RESTRIÇÃO SeqüênciaTurnos */
FOR i = 1 TO M STEP 1
        // pega o número de blocos de seqüência de turnos sem folga para cada empregado [i]
        t1 = B [i]
        FOR j = 1 TO t1 STEP 1
                // inicialize variável nebulosa com o tamanho de cada bloco de seqüência de turnos
                INICIALIZE SequenciaTurnos = SeqTurnos [i][j]. tamanho
                // formule a restrição,
                COMPARAÇÃO NEBULOSA NívelSatisfação IS SequenciaTurnos < 6
                // avalie a restrição
                AVALIE NívelSatisfação
                // insira a restrição na árvore de hierarquia de restrições, ramal : SEQUENCIATURNOS
                AGREGAÇÃO SEQUENCIATURNOS AND NívelSatisfação empregado[i].bloco[j]
        NEXT j
NEXT i
```

Figura 4.13 – Formulação da Restrição *SeqüênciaTurnos*

A variável lingüística *SeqüênciaTurnos* possui cinco valores ou termos lingüísticos com suas respectivas pertinências, representadas na figura 4.14. Na tabela 4.5, detalham-se as regras de inferência para avaliação desta restrição. A recomendação de início de reparos para esta restrição foi definida para o primeiro dia do bloco em conflito, variando até o tamanho do mesmo.



onde SS = *sem_satisfação*
SB = *satisfação_baixa*
SA = *satisfação_alta*
SMB = *satisfação_muito_baixa*
SN = *satisfação_normal*

Figura 4.14 - Representação Qualitativa da Restrição *SeqüênciaTurnos*

Tabela 4.5 – Regras de Inferência para a Comparação Nebulosa da Restrição *SeqüênciaTurnos*

1. SE *SeqüênciaTurnos* É *sem_satisfação*	ENTÃO *NívelSatisfação* É	*ruim.*
2. SE *SeqüênciaTurnos* É *satisfação_muito_baixa*	ENTÃO *NívelSatisfação* É	*ruim.*
3. SE *SeqüênciaTurnos* É *satisfação_baixa*	ENTÃO *NívelSatisfação* É	*ruim.*
4. SE *SeqüênciaTurnos* É *satisfação_normal*	ENTÃO *NívelSatisfação* É	*zero.*
5. SE *SeqüênciaTurnos* É *satisfação_alta*	ENTÃO *NívelSatisfação* É	*zero.*

4.7.2.5 Restrição para Evitar Alocações Consecutivas sem Folga (*SeqüênciaFolgas*)

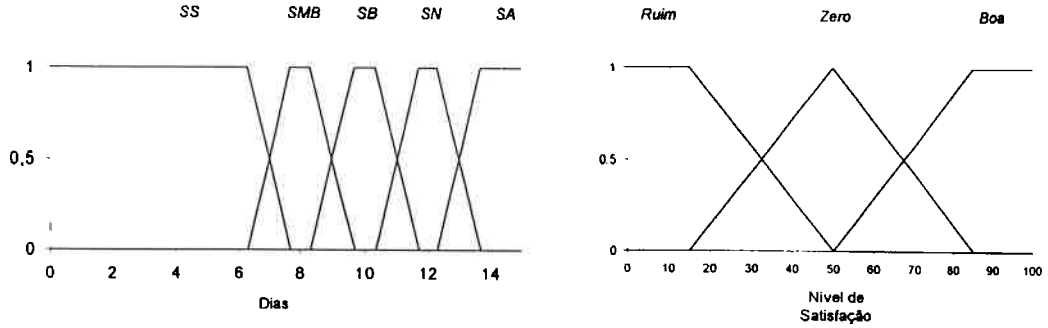
Esta restrição evita que um mesmo empregado seja alocado a turnos consecutivos sem a devida folga, especialmente no caso de uma designação diurna após ter cumprido um turno noturno. Calcular-se-ão, para cada empregado, as horas de folga entre dois turnos consecutivos de dias diferentes, detectando desta forma, por exemplo, a presença de um turno noturno seguido de um turno de manhã sem a sua devida folga.

Do mesmo modo, esta restrição tenta elaborar uma melhor distribuição das folgas, permitindo, no possível, que estas sejam da mesma extensão para um mesmo empregado. Assim, formulou-se a restrição, para um definido número de empregados M e número de pares de turnos consecutivos em dias diferentes Q, como se mostra na figura 4.15.

```
/* RESTRIÇÃO SeqüênciaFolgas */
FOR i = 1 TO M STEP 1
        // pega o número de pares de turnos consecutivos em dias diferentes para cada empregado [i]
        t1 = Q [i]
        FOR j = 1 TO t1 STEP 1
                // inicialize variável nebulosa com as horas de folga de cada par de turnos consecutivos
                // em dias diferentes
                INICIALIZE SequenciaFolgas = SeqFolgas [i][j]. horas
                // formule a restrição,
                COMPARAÇÃO NEBULOSA: NívelSatisfação IS SequenciaFolgas > 8
                // avalie a restrição
                AVALIE NívelSatisfação
                // insira a restrição na árvore de hierarquia de restrições, ramal : SEQUENCIAFOLGAS
                AGREGAÇÃO SEQUENCIAFOLGAS AND NívelSatisfação empregado[i].par[j]
        NEXT j
NEXT i
```

Figura 4.15 – Formulação da Restrição *SeqüênciaFolgas*

A figura 4.16 mostra a definição da variável lingüística *SeqüênciaFolgas* e suas funções de pertinência a fim de representar qualitativamente esta restrição. O respectivo conjunto de regras de inferência necessários para a comparação nebulosa está indicado na tabela 4.6. O início de reparo é exatamente o dia da violação da restrição.



onde SS = *sem_satisfação* SMB = *satisfação_muito_baixa*
SB = *satisfação_baixa* SN = *satisfação_normal*
AS = *satisfação_alta*

Figura 4.16 - Representação Qualitativa da Restrição *SeqüênciaFolgas*

Tabela 4.6 – Regras de Inferência para a Comparação Nebulosa da Restrição *SeqüênciaFolgas*

1. SE *SeqüênciaFolgas*	É *sem_satisfação*	ENTÃO *NívelSatisfação* É *ruim.*
2. SE *SeqüênciaFolgas*	É *satisfação_muito_baixa*	ENTÃO *NívelSatisfação* É *ruim.*
3. SE *SeqüênciaFolgas*	É *satisfação_baixa*	ENTÃO *NívelSatisfação* É *ruim.*
4. SE *SeqüênciaFolgas*	É *satisfação_normal*	ENTÃO *NívelSatisfação* É *zero.*
5. SE *SeqüênciaFolgas*	É *satisfação_alta*	ENTÃO *NívelSatisfação* É *zero.*

4.7.3 Otimização Nebulosa

Devido à complexidade de aplicações realísticas de programações, é difícil escolher um método de otimização que, baseado nas técnicas clássicas de enumeração, garanta uma solução ótima. Pesquisas no campo da teoria da complexidade demonstram que vários problemas, especialmente de aplicações do mundo real, são de difícil tratamento. A busca de espaços está sempre avançando nas capacidades computacionais e o estado da arte dos computadores limita o avanço de abordagens enumerativas. Portanto, a abordagem aqui apresentada está comprometida com um método que tem por objetivo solucionar este tipo de problema. O compromisso está entre a viabilidade da solução do problema e o tempo que este requer para chegar-se a uma solução. O método concentra-se na melhoria iterativa de uma solução preliminar de um problema. (vide seção 3.4.2). Heurísticas, ligadas a esta abordagem, decidem que tipo de modificação deve ser escolhida, com base na avaliação dos resultados de uma solução.

Uma das vantagens das técnicas de melhorias iterativas são as modificações, produzidas a partir de passos de reparo, a um custo baixo. Uma outra vantagem, segundo Dorn et al. (DORN:1994), é o fato de a qualidade da solução inicial não ser fator decisivo para encontrar-se uma boa solução. Além disso, as soluções satisfatórias são encontradas em um tempo relativamente curto devido à característica "em qualquer momento". Esta característica significa que o algoritmo pode parar logo após cada

iteração, ficando para o usuário decidir se ele vai querer continuar com o processamento de melhoria iterativa ou com o resultado obtido que já pode ser satisfatório.

Uma discussão mais exaustiva sobre as técnicas de melhoria iterativa ou heurísticas baseadas em reparos pode ser encontrada em Minton et al (MINTON:1992) e Dorn et al (DORN:1994). Os primeiros comparam experimentalmente um algoritmo baseado em reparos com uma abordagem construtiva com amplo sucesso para a técnica iterativa. Dorn et al. comparam quatro abordagens diferentes de heurísticas de reparo.

Resumindo, a otimização nebulosa conta com as seguintes partes:

- uma função de avaliação das restrições nebulosas definidas na seção anterior;
- um conjunto de operadores de modificações ou passos reparos, estes dependem definitivamente do problema em estudo;
- um procedimento para definir como devem ser aplicados estes operadores de modificações, esta é uma das principais questões de estabelecer como a engenharia do conhecimento está correlacionada com a representação do problema;
- o algoritmo de controle como um todo, baseado nesta aplicação nas técnicas de melhoria iterativa.

Cada uma destas partes serão descritas na seqüência.

4.7.3.1 Avaliação das Restrições Nebulosas

Os objetivos gerais no desenvolvimento da programação de turnos são minimizar tanto a sobrecarga quanto a sub-carga de horas de trabalho de uma mesma categoria ou empregado, sem infringir as inúmeras restrições de ordem legal, regulamentar, sindical e empresarial.

O objetivo da otimização então é encontrar boas programações dentro de um universo de soluções possíveis. Quando se fala de otimização, há também de se prestar atenção à avaliação da programação de turnos. As funções de avaliação servirão de guias no processo de otimização. Estas funções de avaliação são:

- **Função de Avaliação Global**
  As avaliações globais consideram a programação como um todo, fornecendo uma pontuação global à programação atualizada dos turnos.

- **Função de Avaliação Parcial**
  A avaliação parcial considera somente partes de uma programação e, sendo assim, fornece sugestões para melhorar áreas específicas do problema.

A otimização é guiada predominantemente pela pontuação e comparação destas duas avaliações. A partir das restrições violadas, é possível criar estratégias de melhorias para as programações. Utiliza-se então a teoria dos conjuntos nebulosos como definido em (NEGOITA:1985), (ZADEH:1989), (DORN:1992), (DORN:1995) e (SLANY:1996), e no anexo A para a modelagem destas avaliações como será apresentado na continuação.

Com a finalidade de facilitar a representação da função de avaliação parcial toma-se a restrição *DiferençaHoras* (seção 4.7.2.1) como modelo de exemplo, e descrito na seqüência; para as demais restrições segue-se um procedimento equivalente. Para a avaliação desta restrição na programação, os seus cinco valores lingüísticos são mapeados sobre outros três que descrevem o grau ou nível de satisfação: *ruim, zero, bom,* como se mostra na figura 4.17. A partir destes valores e sendo $d_{[i,(j,j+1)]}$ o valor da diferença de horas trabalhadas por um empregado i na semana j e j+1, com M categorias ou empregados e N semanas, uma dada programação P pode ser avaliada de acordo com esta restrição, a partir da seguinte função de avaliação parcial:

$$Sat.p_{[DiferencaHoras]}(P) = \bigcap_{\substack{i=1 \\ j=1}}^{N,M} Sat.p_{[DiferencaHoras]}(d)_{[i,(j,j+1)]} \qquad (4.18)$$

O operador nebuloso *and* foi utilizado para tomar o mínimo dos argumentos na equação (4.18), mas outros modelos matemáticos (vide seção 3.5.5) poderiam ser testados para a aplicação em estudo.

Nas tabelas 4.2 e 4.7, são definidas as regras que medem o grau de satisfação para esta restrição. Estas regras podem ser interpretadas diretamente como regras de inferência nebulosa (vide anexo A).

O cálculo do nível de satisfação para a restrição *DiferençaHoras* é ilustrado na figura 4.17. Admita-se, por exemplo, que determinada situação leva a um valor de 4 horas para esta restrição como sendo $d_{(4,3)}$ considerando somente o quarto empregado ou categoria e a diferença entre a terceira e quarta semana, a questão agora é estabelecer o seu grau de satisfação. Para decidir isto com as regras de inferência dadas, devem ser combinados os termos lingüísticos vagos e termos *crisp* com valores numéricos incertos. Isto é realizado com as funções de pertinência definidas na figura 4.17, para as partes condicional e de conclusão.

Nestas condições o valor numérico de entrada $d_{(4,3)}$ = 4 horas se relaciona aproximadamente com os termos lingüisticos *diferença_zero* e *diferença_media_positiva.* Seguindo as linhas pontilhadas, as funções de pertinência da conclusão para as regras, 2 e 3, que aparecem como resultado do cálculo, são as duas funções de pertinência: *zero*[*DiferençaHoras*] $d_{(4,3)}$ e *boa*[*DiferençaHoras*] $d_{(4,3)}$.

$$Sat.p_{[DiferencaHoras]}d_{(4,3)} = \max\left\langle zero_{[DiferencaHoras]}d_{(4,3)}, boa_{[DiferencaHoras]}d_{(4,3)}\right\rangle \qquad (4.19)$$

Sua combinação, indicada na equação (4.19), é uma nova função de pertinência definindo a satisfação parcial de $d_{(4,3)}$. Obteve-se como resultado que a satisfação da restrição de *DiferençaHoras* para $d_{(4,3)}$ é uma distribuição possibilística (vide anexo A) que é, de um modo geral, mais para *boa* que para *zero.* Os operadores nebulosos que podem ser usados para a operação *máxima* dos argumentos na equação (4.19) foram estudados na seção 3.5.5

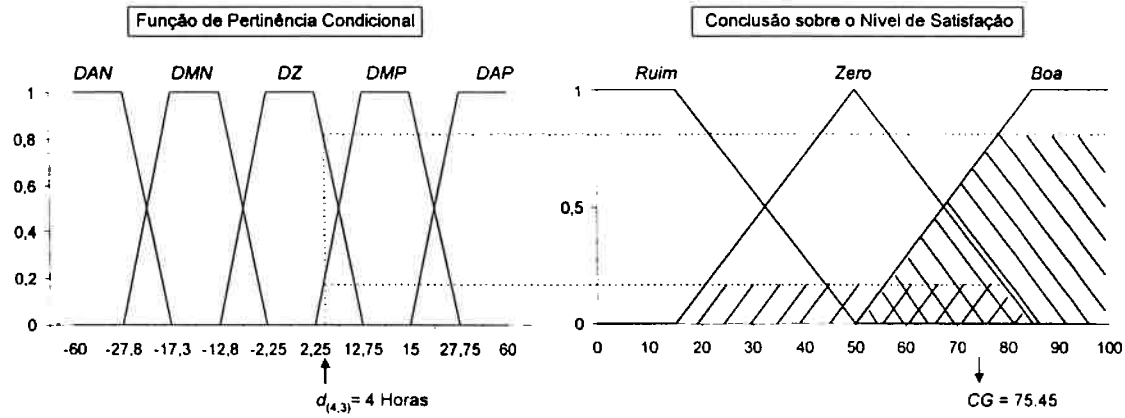
A satisfação deve também ser defuzzificada, utilizando algum dos métodos apresentados no anexo A. No exemplo, utilizar-se-á o centro de gravidade da superfície e tomar-se-á o valor da coordenada x como resultado. Este valor representa a satisfação parcial (vide figura 4.6) que indica o grau de satisfação da restrição analisada, e pode ser computado pela equação (4.20).

$$Defuzzy\left\langle Sat.p_{[DiferencaHoras]}d_{[i,(j,j+1)]}\right\rangle = \frac{\int_{x=\min}^{max} x \cdot Sat.p_{[DiferencaHoras]}d_{[i,(j,j+1)]}(x)dx}{\int_{x=\min}^{max} Sat.p_{[DiferencaHoras]}d_{[i,(j,j+1)]}(x)dx} \tag{4.20}$$

Para avaliar cada programação durante as etapas de refinamento dos métodos iterativos de otimização, é necessário computar uma função de avaliação de modo a calcular a satisfação global da programação em análise, como referenciado no algoritmo da figura 4.6. Para uma dada programação P com M categorias, N semanas, B blocos de seqüências de turnos e Q pares de turnos consecutivos, a função de avaliação global, *Sat.g,* é definida pela equação (4.21).

$$\begin{aligned} Sat.g(P) = & \sum_{\substack{i=1 \\ j=1}}^{M,N} \left\langle Sat.p_{[DiferencaHoras]} + Sat.p_{[HorasTrabalho]} + Sat.p_{[FinaisSemana]} \right\rangle_{ij} \\ & + \sum_{\substack{i=1 \\ j=1}}^{M,B} \left\langle Sat.p_{[SequenciaTurnos]} \right\rangle_{ij} + \sum_{\substack{i=1 \\ j=1}}^{M,Q} \left\langle Sat.p_{[SequenciaTurnos]} \right\rangle_{ij} \end{aligned} \tag{4.21}$$

Modelo Exemplo



onde DAP = *diferença_alta_positiva*
DZ = *diferença_zero*
DAN = *diferença_alta_negativa*
DMP = *diferença_media_positiva*
DMN = *diferença_media_negativa*

Figura 4.17 – Representação do Cálculo do Nível de Satisfação para a entrada *crisp* $d_{(4,3)}$ na Restrição *DiferençaHoras*

Tabela 4.7 – Regras de Inferência para a Comparação Nebulosa da Restrição *DiferençaHoras*

1. SE *DiferençaHoras* É	*diferença_alta_positiva*	ENTÃO	*NívelSatisfação* É	*ruim*.
2. SE *DiferençaHoras* É	*diferença_media_positiva*	ENTÃO	*NívelSatisfação* É	*zero*.
3. SE *DiferençaHoras* É	*diferença_zero*	ENTÃO	*NívelSatisfação* É	*boa*.
4. SE *DiferençaHoras* É	*diferença_media_negativa*	ENTÃO	*NívelSatisfação* É	*zero*.
5. SE *DiferençaHoras* É	*diferença_alta_negativa*	ENTÃO	*NívelSatisfação* É	*ruim*.

Neste exemplo modelo, compute-se o nível de satisfação de $d_{(4,3)}$, a partir da equação (4.18) tem-se:

$$Sat.p(P) = \bigcap_{\substack{i=1 \\ j=1}}^{N,M} Sat.p(4) \tag{4.22}$$

Interpretou-se o operador nebuloso da equação 4.22 inicialmente como uma *t-norma* (T_M) (vide seção 3.5.5, equação 3.8) – operação representada pela figura 4.17 -, assim:

$$T_M(\mu_A(x), \mu_B(x)) = \min(\mu_A(x), \mu_B(x)) \tag{4.23}$$

$\mu_{(diferença\_zero)} =$ 0.8333 $\quad T_M =$ 0.8333
$\mu_{(diferença\_media\_positiva)} =$ 0.1667 $\quad T_{M'} =$ 0.1667

Usando as regras de inferência nebulosa (tabela 4.7), encontrou-se somente as regras 2 e 3 as quais contribuem para os resultados calculados tal como indica o diagrama acima. De acordo com este resultado, $d_{(4,3)}$ =4 horas é mais *boa* que *zero*.

$$Sat.p_{[DiferencaHoras]}d_{(4,3)} = \max\left\langle zero_{[DiferencaHoras]}d_{(4,3)}, boa_{[DiferencaHoras]}d_{(4,3)}\right\rangle \tag{4.24}$$

Interpretou-se o operador nebuloso max. da equação 4.24 inicialmente como uma *t-conorma* (S_M) (vide seção 3.5.5, equação 3.9) –deste modo têm-se:

$$S_M(\mu_A(x), \mu_B(x)) = \max(\mu_A(x), \mu_B(x)) \tag{4.25}$$

$S_M =$ 0.8333
$S_{M'} =$ 0.1667

A defuzzificação é calculada segundo a equação (4.20) e reproduzida para o modelo exemplo na equação 4.26, assim:

$$Defuzzy\left\langle Sat.p_{[DiferencaHoras]}d_{(4,3)}\right\rangle = \frac{\int_{x=\min}^{\max} x \cdot Sat.p_{[DiferencaHoras]}d_{(4,3)}(x)dx}{\int_{x=\min}^{\max} Sat.p_{[DiferencaHoras]}d_{(4,3)}(x)dx} \tag{4.26}$$

Defuzzificação = 75.450;

Isto quer dizer que o valor *crisp* de $d_{(4,3)}$ =4 horas possui um grau parcial de satisfação de 75.45 utilizando os operadores max-min.

Para o cálculo da avaliação global tem-se que calcular, utilizando-se o mesmo procedimento anterior, as outras avaliações parciais, para todos os valores de entrada desta restrição, assim como para cada uma das restrições restantes, fazendo-se ao final de uma avaliação completa da atual programação uma somatória de todos estas

satisfações parciais, isto com o fim de verificar se houve uma melhoria na avaliação global em comparação com a anterior programação.

4.7.3.2 Passos de Reparo

Os passos de reparo do problema são conduzidos pelos operadores de modificação. Eles permitem deslocar uma instância válida do problema a outra. Tais modificações são a base para que cada algoritmo de otimização tente chegar, em termos de avaliação, a uma instância superior, modificando a instância corrente do problema.

Em geral, é sempre possível identificar certos tipos de passos de reparo. Em nossa aplicação, a definição dos passos de reparo está quase diretamente ditada pelo plano de operação. Observando este plano é possível definir-se para cada dia, que tipo de turno e quantos devem ser alocados. Para ter-se bom desempenho, os passos de reparo deverão ser implementados de maneira simples, já que são bastante utilizados durante o processo de otimização. A maior parte das modificações em um problema são simples trocas de instâncias válidas (DORN:1995).

As modificações na programação dos turnos são efetuadas então pelos passos de reparo. Sem nenhum guia, estas modificações poderiam se suceder aleatoriamente. Para evitar isso, são identificadas as posições onde as restrições são violadas, às quais são utilizadas como informação para que os passos de reparo atuem. Isto pode ser em qualquer posição na programação de turnos. Como se verá na próxima seção e na figura 4.18, a posição da restrição violada na programação também determinará que passo reparo deva ser aplicado. Nesta situação, os passos de reparo assumem esta posição como um ponto de partida para tentar encontrar uma possível modificação, enquanto interagem com a completa programação de turnos. A indicação da posição inicial, definida para cada restrição violada, permite reduzir atrasos na aplicação de um passo reparo. A posição de início poderia ser também aleatória, ou ainda, qualquer uma que seja apropriada ao problema em tratamento.

Observando o plano de operação do nosso problema em estudo, mostrado no capítulo 5 (tabela 5.2), pode-se identificar inicialmente três grupos principais, nos quais as operações são equivalentes dentro do mesmo grupo: (a) de segunda a sexta, as operações são idênticas em relação aos mesmos tipos de turnos (turno da manhã, turno da tarde e turno da noite), definindo, deste modo, um passo reparo para cada turno; (b) um quarto operador de modificação focaliza-se nas mudanças que envolvem o turno de fim-de-semana de 12 horas, aos sábados e domingos e, finalmente, (c) é definido um passo de reparo para o turno de fim-de-semana de 4 horas aos sábados. A seguir, serão descritos os cinco diferentes tipos de passos de reparo.

4.7.3.2.1 Passos de Reparos dos Turnos TM, TD e TN

O plano de operação mostra que nos dias de segunda a sexta são requeridos sempre turnos de manhã (TM), de tarde (TD) e de noite (TN). Se uma violação de uma restrição e sua posição derivada de início estiverem dentro dessa faixa, aplica-se um operador de modificação para trocar a posição com outra dentro da mesma faixa. De fato, duas trocas têm de ser realizadas com a finalidade de preservar os requisitos do

plano de operação, pois, o número de turnos alocados em cada categoria não pode ser afetado por uma troca, mantendo as restrições *hards* inalteradas. Além disso, fazendo-se uma segunda troca oposta, também é preservada a restrição dos turnos requeridos de um dia. Se a violação estiver em uma posição onde um turno esteja selecionado, seja este TM, TD ou TN, o operador procura por uma posição na mesma categoria onde nenhum turno esteja alocado, tendo o cuidado para que sempre seja utilizado nesta operação o mesmo tipo de turno. As trocas envolvem um intercâmbio com uma categoria livre, por exemplo, uma que não tenha sido alocada. A operação é ilustrada na tabela 4.8 e implementada segundo o algoritmo das figuras 4.19 e 4.20 para o passo reparo corresponde ao turno da manhã (TM), para os outros operadores de modificação segue-se um procedimento similar.

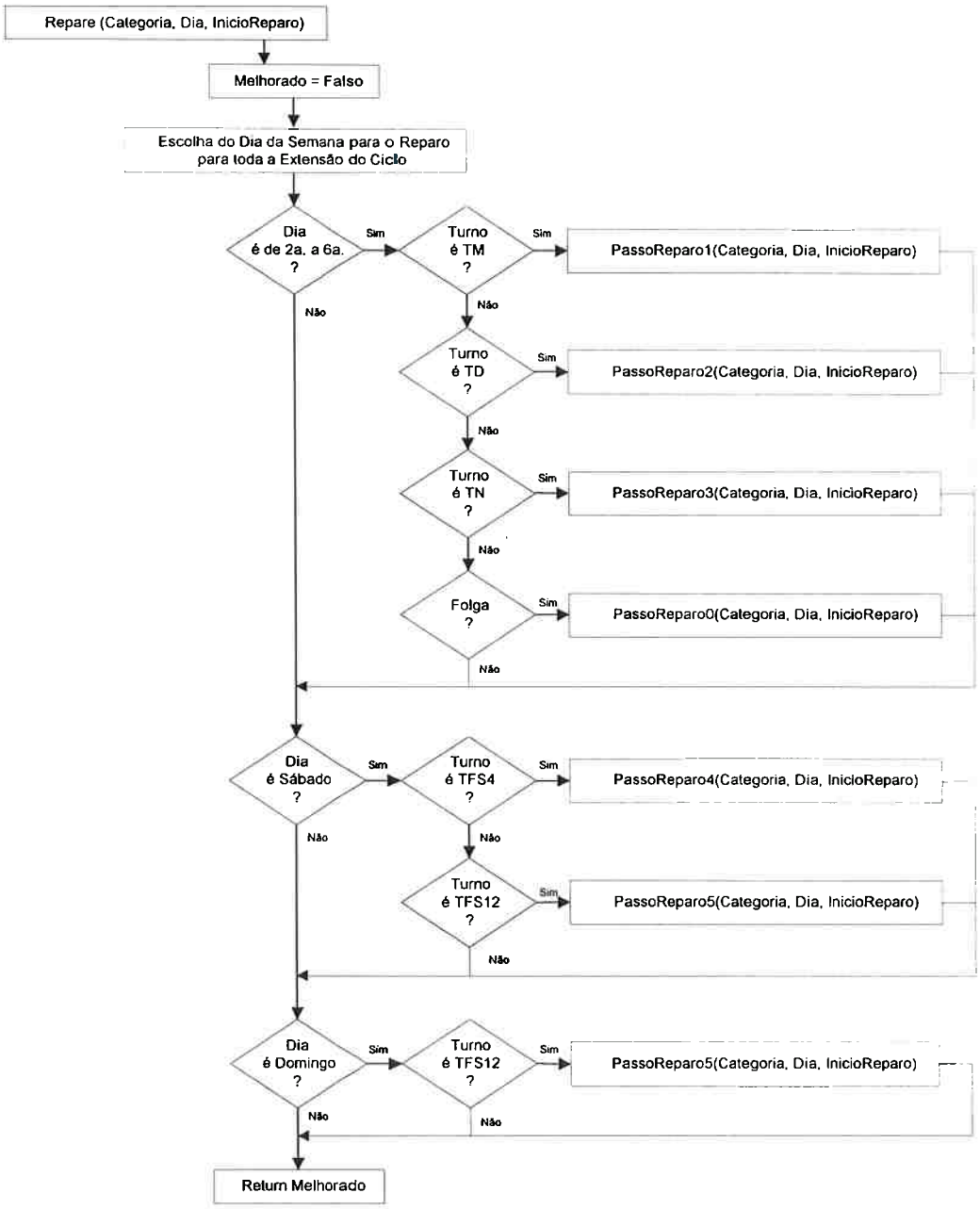


Figura 4.18 - Algoritmo de Controle para a definição dos Passos Reparos

Tabela 4.8 – Representação da Operação de um Passo de Reparo para os Turnos TM, TD e TN

	Semana 0							Semana 1						
Dia	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	Segunda	Terça	Quarta	Quinta	Sexta	Sábado	Domingo	Segunda	Terça	Quarta	Quinta	Sexta	Sábado	Domingo
A	TM		TM		TN		TFS12	TN	TD		TM	TD		
B	TM	TM	TD	TM	TM			TD	TM	TM		TM		
C	TM		TM	TM	TD			TD		TD	TM		TFS4	
D	TD	TN		TD		TFS4		TM		TM	TD	TM		
E	TD	TD	TD	TM	TD				TM	TM	TN	TN		TFS12
F		TD	TN		TM			TM	TD	TD		TM	TFS12	
G		TM	TM	TN		TFS12		TM		TN	TD	TD		
H	TN		TM	TD	TM				TN	TM	TM			

Violação em (Categoria D, Dia 0)
Troca 1: (Categoria D, Dia 0) - (Categoria D, Dia 8)
Troca 2: (Categoria F, Dia 8) - (Categoria F, Dia 0)

4.7.3.2.2 Passo de Reparo dos Turnos TFS4

Este passo de reparo enfoca os turnos de fim-de-semana de 4 horas (TFS4). O plano de operação mostra que este tipo de turno é usado somente aos sábados, durante os quais define-se a operação de troca. A tabela 4.9 apresenta um exemplo deste tipo de passo reparo.

Tabela 4.9 – Representação da Operação de um Passo de Reparo para os Turnos TFS4

	Semana 0							Semana 1						
Dia	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	Segunda	Terça	Quarta	Quinta	Sexta	Sábado	Domingo	Segunda	Terça	Quarta	Quinta	Sexta	Sábado	Domingo
A	TM		TM		TN		TFS12	TN	TD		TM	TD		
B	TM	TM	TD	TM	TM			TD	TM	TM		TM		
C	TM		TM	TM	TD			TD		TD	TM		TFS4	
D	TD	TN		TD		TFS4		TM		TM	TD	TM		
E	TD	TD	TD	TM	TD				TM	TM	TN	TN		TFS12
F		TD	TN		TM			TM	TD	TD		TM	TFS12	
G		TM	TM	TN		TFS12		TM		TN	TD	TD		
H	TN		TM	TD	TM				TN	TM	TM			

Violação em (Categoria C, Dia 12)
Troca 1: (Categoria C, Dia 12) - (Categoria C, Dia 5)
Troca 2: (Categoria D, Dia 5) - (Categoria D, Dia 12)

4.7.3.2.3 Passo de Reparo dos Turnos TFS12

O passo de reparo para turnos de fim-de-semana de 12 horas (TFS12) é similar ao descrito anteriormente, mas com a variante de que sejam permitidas as trocas apenas os sábados e domingos. A tabela 4.10 demonstra como é realizada uma destas modificações.

Tabela 4.10 – Representação da Operação de um Passo de Reparo para os Turnos TFS12

	Semana 0							Semana 1						
Dia	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	Segunda	Terça	Quarta	Quinta	Sexta	Sábado	Domingo	Segunda	Terça	Quarta	Quinta	Sexta	Sábado	Domingo
A	TM		TM		TN		TFS12	TN	TD		TM	TD		
B	TM	TM	TD	TM	TM			TD	TM	TM		TM		
C	TM		TM	TM	TD			TD		TD	TM		TFS4	
D	TD	TN		TD		TFS4		TM		TM	TD	TM		
E	TD	TD	TD	TM	TD				TM	TM	TN	TN		TFS12
F		TD	TN		TM			TM	TD	TD		TM	TFS12	
G		TM	TM	TN		TFS12		TM		TN	TD	TD		
H	TN		TM	TD	TM				TN	TM	TM			

Violação em (Categoria A, Dia 6)
Troca 1: (Categoria A, Dia 6) - (Categoria D, Dia 13)
Troca 2: (Categoria E, Dia 13) - (Categoria F, Dia 6)

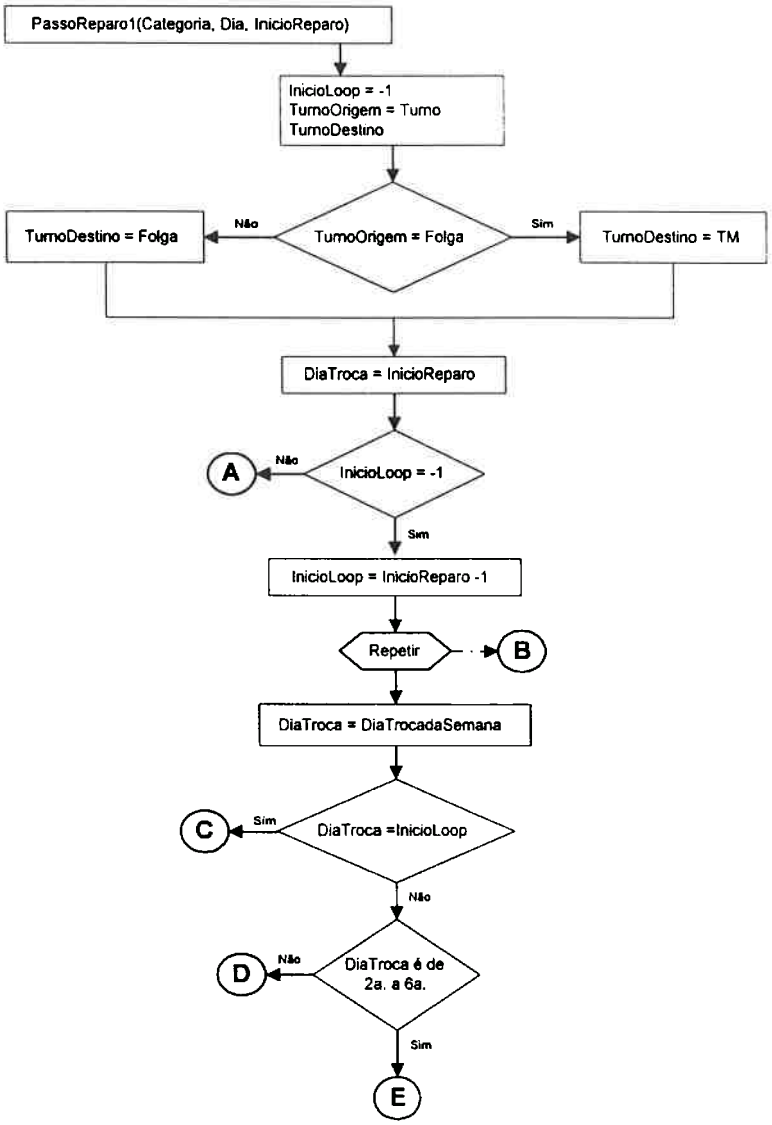


Figura 4.19 - Algoritmo de Definição do Passo Reparo 1 (Turno TM)

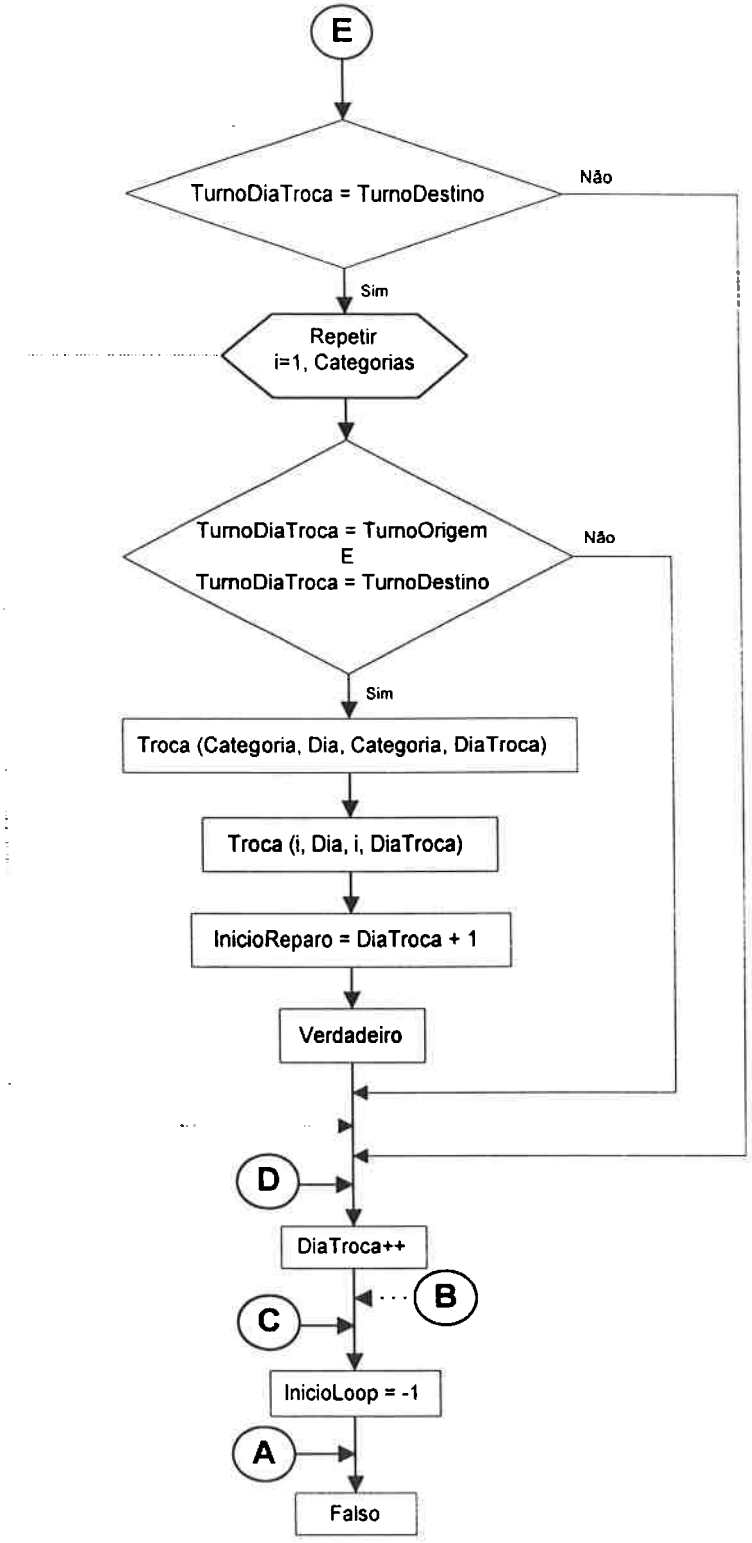


Figura 4.20 - Algoritmo de Definição do Passo Reparo 1 (Turno TM) (continuação)

4.7.3.3 O Algoritmo de Controle

Durante a implementação da aplicação em questão, serão introduzidos dois algoritmos de controle baseados nas técnicas de melhoria por reparos. Estes são: reparo em profundidade e reparo por modificações aleatórias, algoritmos descritos inicialmente nas seções 3.4.2.1 e 3.4.2.2 e representados a partir da figura 4.21 até a figura 4.25.

O algoritmo de controle baseado no método por reparos em profundidade recebe uma programação inicial que é logo após avaliada; os cálculos produzem uma lista de restrições avaliadas que são classificadas pela severidade das violações. A partir de cada restrição violada, por exemplo, pode-se derivar possíveis candidatas a modificações com suas posições de início de acordo com o tipo de restrição. Para aumentar o número de possíveis modificações tratadas em cada nível de profundidade da busca, o algoritmo tenta fazer todas as possíveis modificações atravessando a lista de restrições violadas e aplicando para cada restrição operadores de modificação sobre todas as posições de início. Por exemplo, o algoritmo começa com um nível de profundidade igual a 1, toma a restrição mais severamente violada e começa com a primeira possível candidata recomendada a reparo. Em seguida aplica os operadores de modificação nesta posição, até encontrar uma melhoria ou tentar todas as operações possíveis com esta específica e inicial candidata. Quando é esgotada a busca para uma candidata inicial, trata-se da próxima, e assim por diante. Se foi atravessada a lista de candidatas iniciais, seleciona-se a próxima restrição violada. Se todas estas tentativas falharem, será aumentado o nível de profundidade e começar-se-á novamente o processo. O número de tentativas deverá ser definido como critério de parada do algoritmo com a melhor avaliação geral até o momento, este foi estabelecido após um grande número de avaliações para nosso caso aplicação ser de 40 tentativas. O algoritmo de controle que apresentou os melhores desempenhos baseado nesta técnica é mostrado nas figuras 4.21 e 4.22.

De modo também a avaliar uma técnica não-determinística implementou-se um algoritmo de controle baseado em reparos por modificações aleatórias. Numa primeira abordagem, fez-se com que o algoritmo decidisse fortuitamente os pontos de início para os reparos durante a extensão completa do ciclo e, por conseguinte, os operadores que devem ser aplicados; se a programação avaliada não apresentar nenhuma melhoria, então será gerada aleatoriamente uma nova posição de início para os reparos conforme se mostra na figura 3.4 da seção 3.4.2.2, correspondente ao método clássico de procura aleatória. Como será visto no capítulo seguinte, o desempenho desta primeira abordagem não foi das melhores. Diferentes mudanças no algoritmo base foram realizadas para nosso caso aplicação, a fim de melhorar o desempenho e aproveitar as vantagens desta abordagem em relação ao algoritmo de controle determinístico (reparos em profundidade) que pode levar a um consumo exagerado de tempo ao examinar todos os níveis da árvore de busca em um espaço que pode ser muito amplo.

O algoritmo apresentado nas figuras 4.23, 4.24 e 4.25 representa uma modificação no algoritmo base onde os sorteios dos inícios dos reparos, para decidir que turnos e que operadores serão aplicados; estão limitados ao conjunto de dias em que a restrição analisada pode ser reparada, sendo *tentativa_de_reparo1* uma função recursiva que testa um determinado passo reparo aplicando um operador de modificação *profundidade* vezes.

Em nosso estudo experimentaram-se estes dois procedimentos, uma técnica determinística (reparo em profundidade) e outra estocástica (reparo por modificações

aleatórias), estabelecendo comparações de ganho em eficiência e tempo de computação para nossa aplicação em estudo.

No capitulo 5, será apresentada a aplicação exemplo e os diferentes testes de desempenho dos algoritmos aqui descritos. O sucesso da busca está fortemente acoplado com alguns parâmetros que são levados em consideração como o número de restrições violadas e possíveis modificações com a candidata inicial.

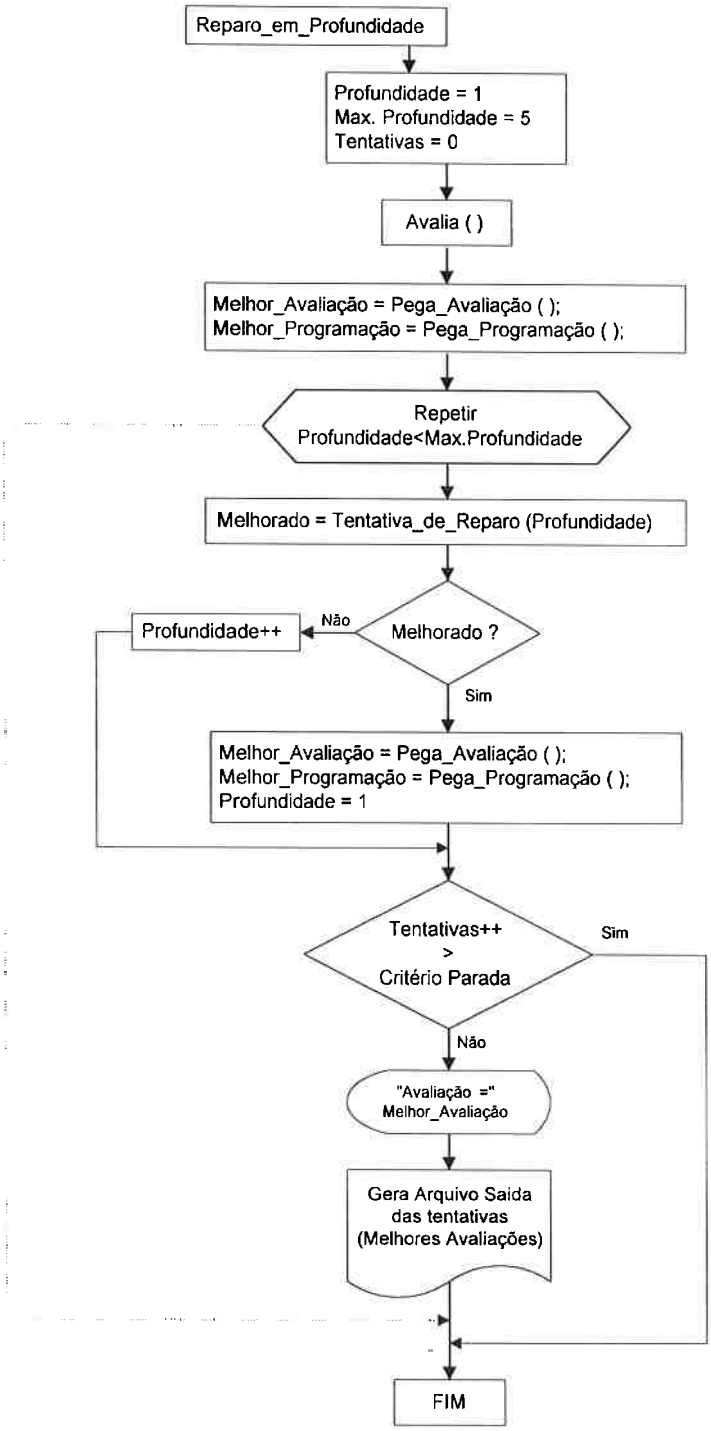


Figura 4.21 - Algoritmo de Controle baseado no Método por Reparos em Profundidade

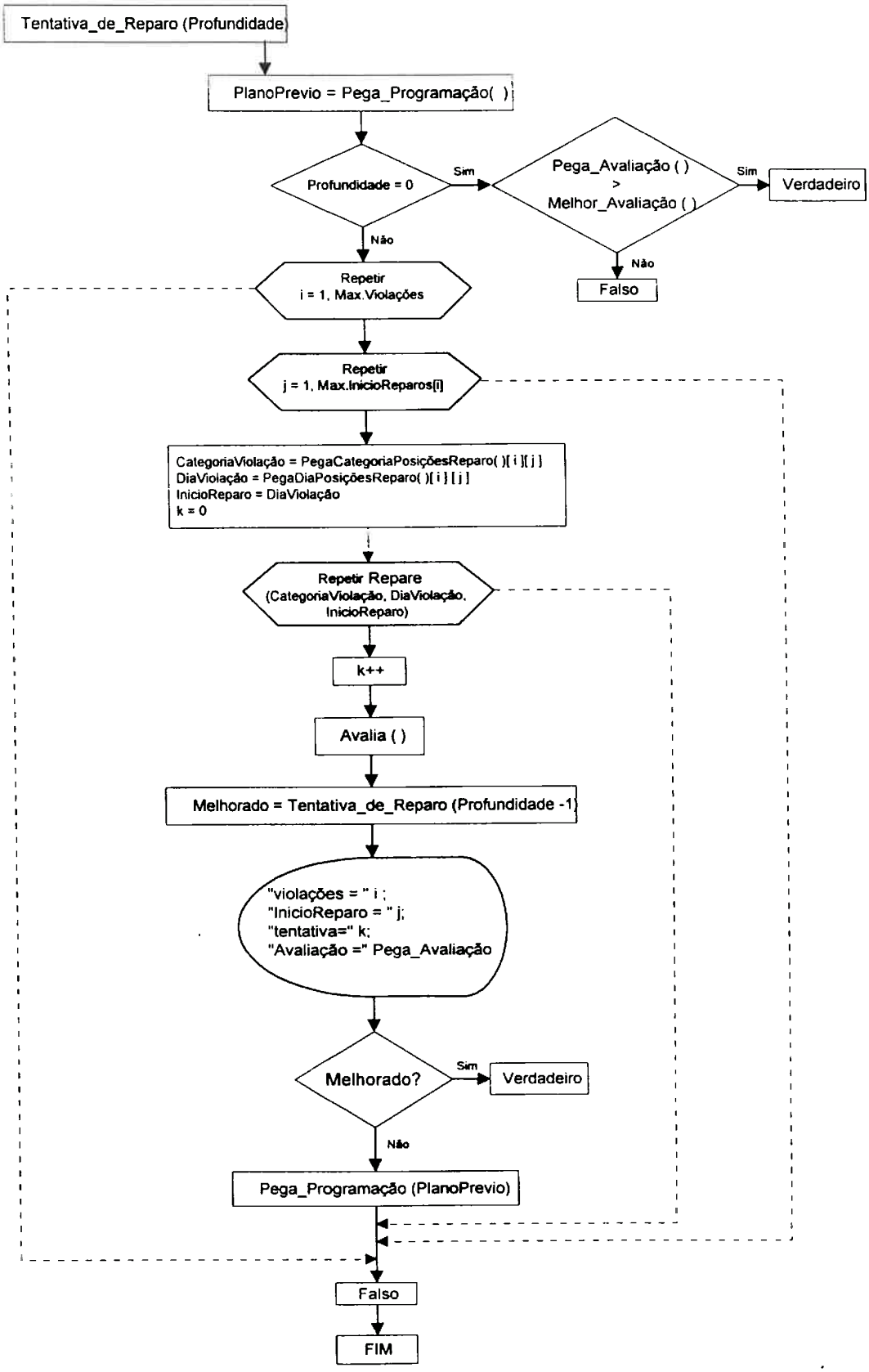


Figura 4.22 - Algoritmo de Controle baseado no Método por Reparos em Profundidade - Função "Tentativa_de_Reparo( )"

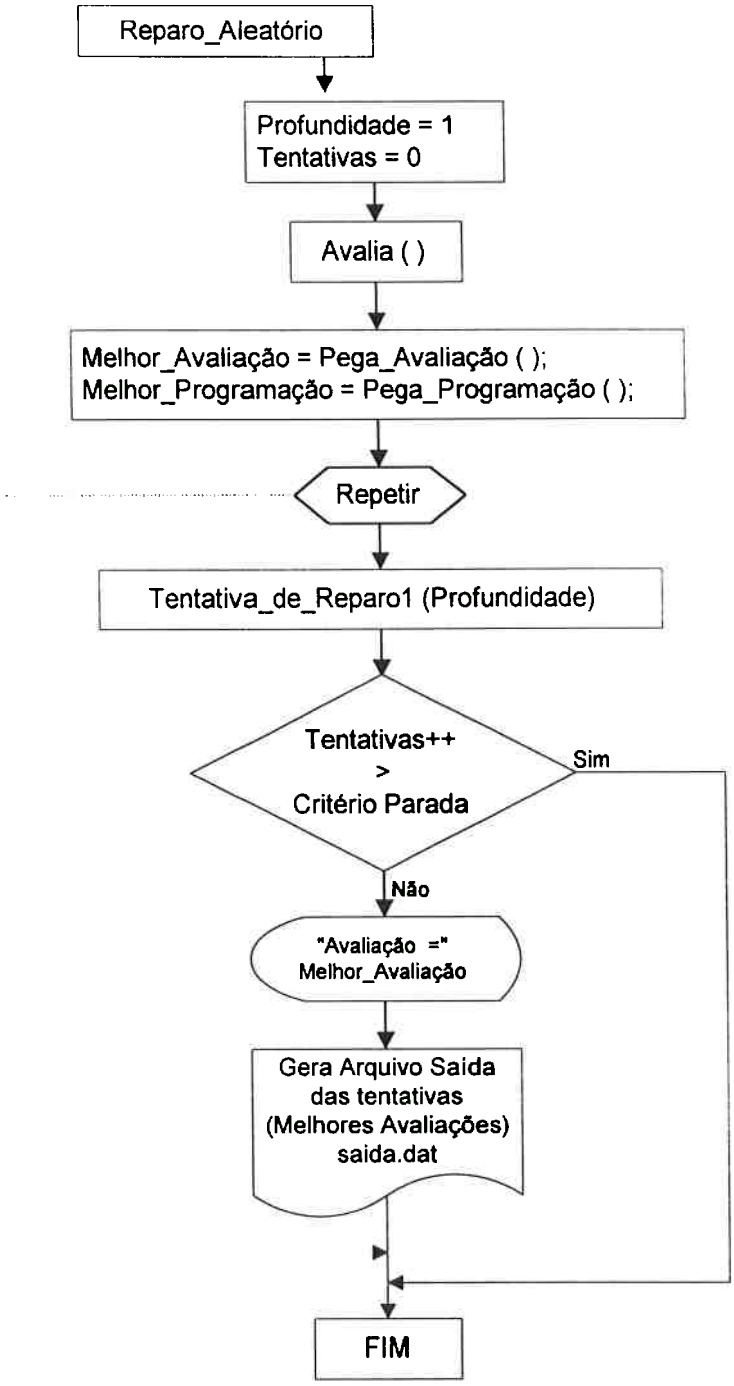


Figura 4.23 - Algoritmo de Controle baseado no Método por Reparos por Modificações Aleatórias

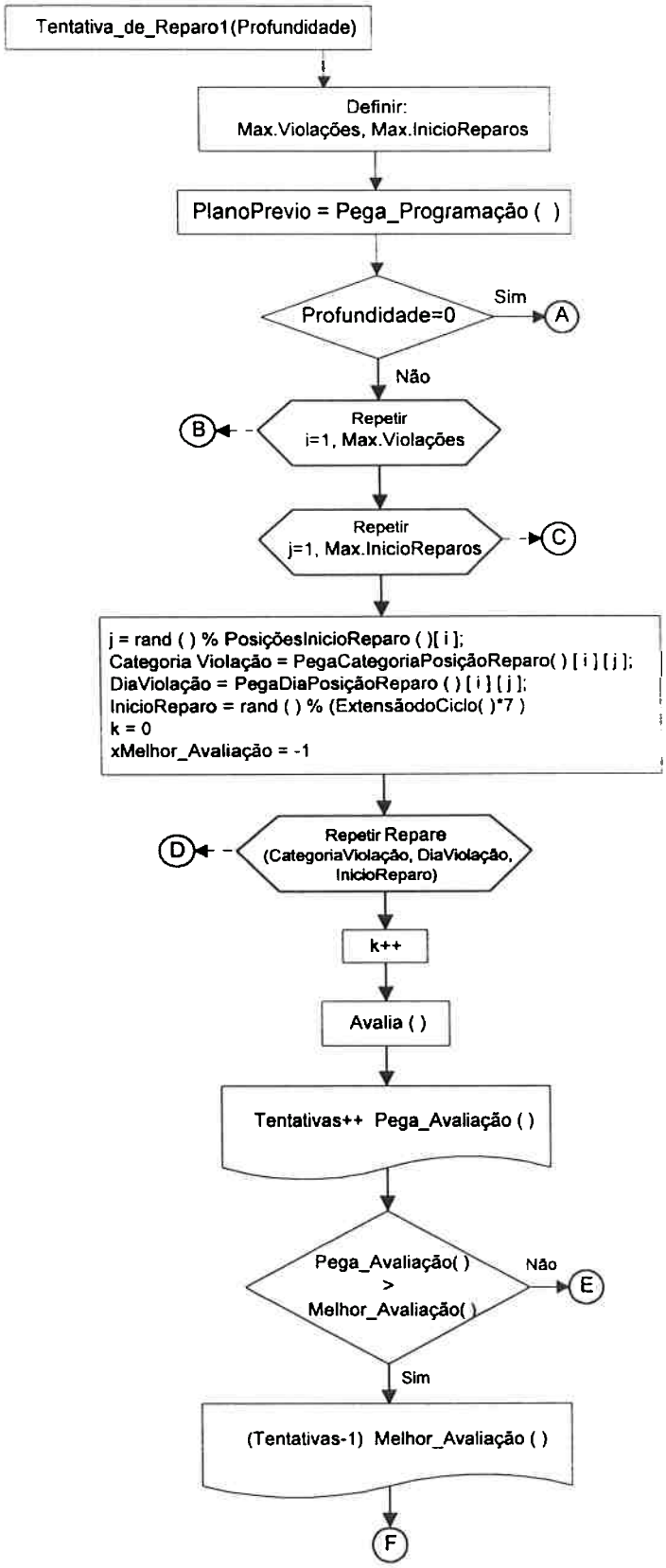


Figura 4.24 - Algoritmo de Controle baseado no Método por Reparos por Modificações Aleatórias - Função "Tentativa_de_Reparo1( )"

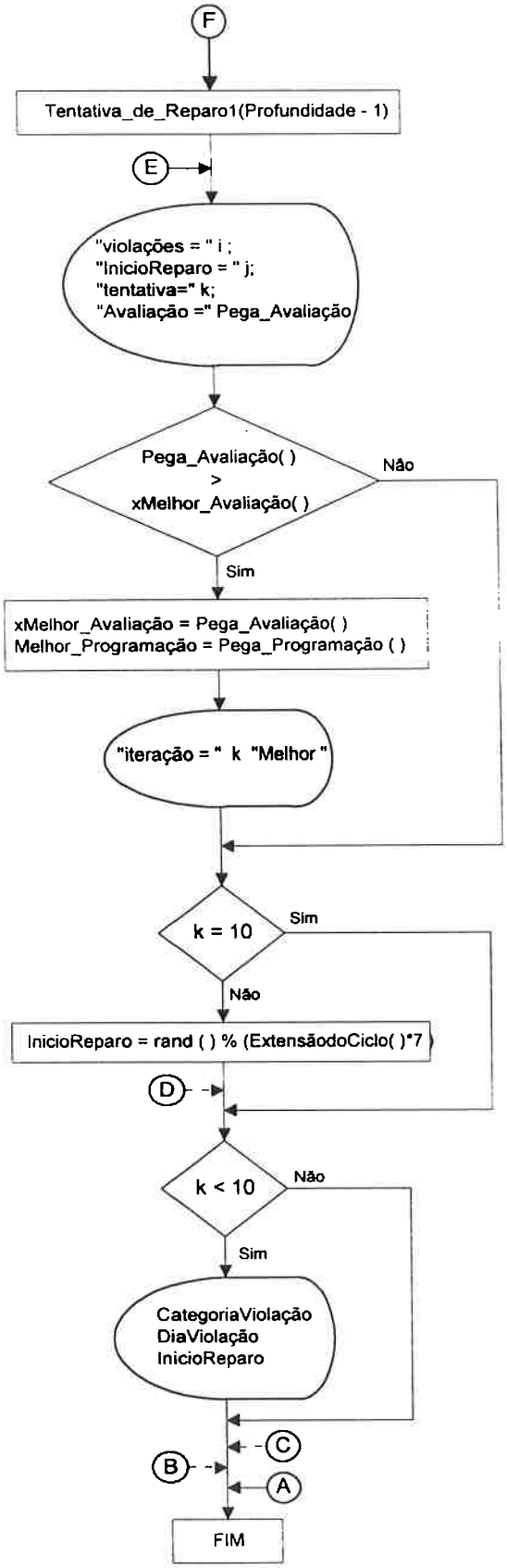


Figura 4.25 - Algoritmo de Controle baseado no Método por Reparos por Modificações Aleatórias - Função "Tentativa_de_Reparo1( )" – (continuação)

Capítulo 5

A APLICAÇÃO :

Processo de Estiva Ideal em um Terminal de Contêineres

5.1 Introdução

No capítulo 4, foi apresentado SIPRODT, um modelo de sistema para a programação dinâmica de turnos no processo de estiva em um terminal de contêineres. Neste capítulo, serão apresentadas aplicações deste sistema através de dois procedimentos. O primeiro para o projeto de uma nova programação de turnos, a partir da definição dos elementos principais do sistema e o plano de operações. Calculam-se o número mínimo de trabalhadores e a programação final dos turnos a partir da melhoria iterativa de uma programação inicial. Este primeiro procedimento servirá de base na geração de um conjunto de dados para o novo terminal, permitindo, desta forma, estabelecer após várias simulações, os elementos principais que mais se adequam a uma demanda de trabalho esperada, além do nível de operários necessários. A base de dados será utilizada no segundo procedimento, no exercício da programação dinâmica de turnos (SIPRODT), mostrando-se várias aplicações com diferentes exemplos de demandas.

Serão também descritos, neste capítulo, os testes empíricos correspondentes ao processo de seleção dos parâmetros dos MN's projetados e comparados em termos de ganho em eficiência e tempo de computação para duas diferentes técnicas de algoritmos de reparo: uma determinística (algoritmo em profundidade) e a outra estocástica (algoritmo de reparo por modificações aleatórias).

Em decorrência do atual processo de modernização dos portos nacionais, em que a organização do trabalho nos terminais passa por uma profunda reestruturação, há uma dificuldade para se apreender a realidade atual devido à sua complexidade. Propõem-se aqui vários exemplos de um modelo ideal de um processo de estiva representativo de um terminal de contêineres, considerando os termos básicos definidos no decorrer do trabalho.

5.2 Procedimento para o Projeto de uma Nova Programação de Turnos

O procedimento para a elaboração de um novo projeto de um programador de turnos para um terminal ideal de contêineres deverá cumprir os passos descritos a seguir.

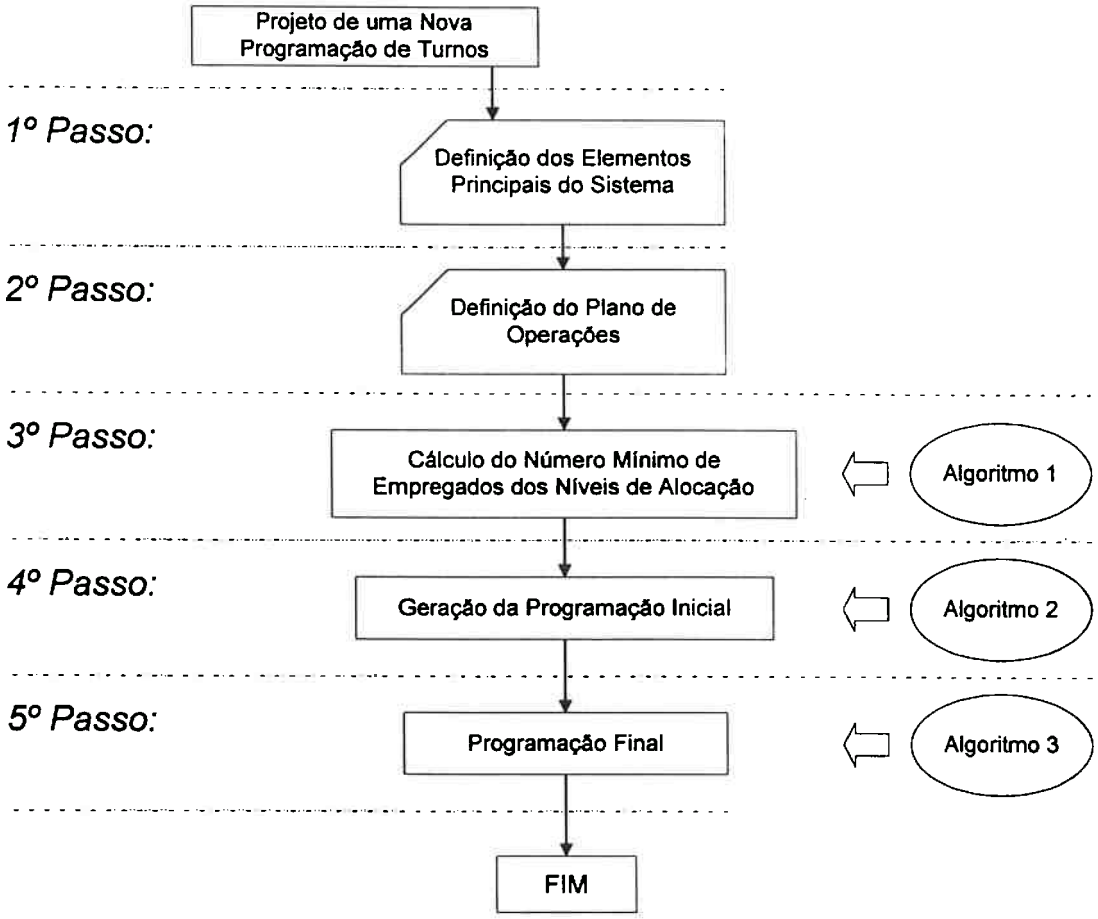


Figura 5.1 – Procedimento para a Elaboração de um Novo Projeto de Programação de Turnos

5.2.1 Definição dos Elementos Principais do Sistema

O primeiro passo no projeto de uma nova programação de turnos é decidir sobre os elementos principais do sistema, especialmente aqueles relacionados ao tipo e à extensão dos turnos.

Existe uma tendência cada vez maior ao trabalho contínuo nos terminais de contêineres, isto é, 24 horas por dia, 7 dias por semana. Uma grande quantidade destes terminais operam a base de 3 turnos com 8 horas cada um, durante dias da semana, e utilizam também algumas outras extensões de turnos, dependendo da movimentação

esperada, para atender as operações dos finais de semana. Como exemplo, são apresentados os dados da tabela 5.1 que definem os elementos principais representativos no atendimento de um processo de estiva para um terminal ideal de contêineres.

Tabela 5.1 - Definição dos Elementos Principais do Sistema

Item	Abreviatura	Definição	Extensão
1	TM	Turno de Manhã	8 horas
2	TD	Turno de Tarde	8 horas
3	TN	Turno de Noite	8 horas
4	TFS4	Turno de Fim-de-Semana	4 horas
5	TFS12	Turno de Fim-de-Semana	12 horas

5.2.2 Definição do Plano de Operações

O segundo passo é estimar o requerimento diário de trabalhadores ou categorias para atender ao processo em estudo. Para este cálculo, em termos de trabalhadores-turnos por dia, serão necessárias, além do número de turnos por dia e de sua extensão, informações sobre a movimentação dos contêineres, a produtividade esperada dos guindastes, a disponibilidade em quantidade e tipo dos equipamentos e os níveis típicos de escalação (tamanho dos ternos). A tabela 5.2 representa um exemplo de um plano de operações para o processo de estiva de um terminal de contêineres.

Tabela 5.2 - Definição do Plano de Operação

2ª-feira	3ª-feira	4ª-feira	5ª-feira	6ª-feira	Sábado	Domingo
3 – TM	2 – TM	3 – TM	3 – TM	2 – TM	3 – TFS4	4 – TFS12
2 – TD	3 – TD	1 – TD	2 – TD	3 – TD	2 – TFS12	
1 – TN	1 – TN		1 – TN	1 – TN		

5.2.3 Cálculo do Número Mínimo de Empregados dos Níveis de Alocação (Algoritmo 1)

Conforme a seção 4.5, e seguindo o algoritmo 1 da figura 4.3, será calculada a quantidade mínima dos níveis de escalação para as categorias e/ou trabalhadores deste estudo.

Dado o plano de operação (Tabela 5.2) obtem-se a requisição de mão de obra para uma semana de trabalho:

$$r_1, \ldots r_7 = 6, 6, 4, 6, 6, 5, 4;$$

A demanda máxima por dia e por semana é calculada a partir de:

$$r_k = \max \{r_1, \ldots r_7\} = 6;$$

$$\sum_{j=1}^{7} r_j = 37$$

Na seqüência, será calculado o número inicial de trabalhadores W', equação (4.19) e a mão de obra inativa t, equação (4.10), como:

$$W' = \max\left\{6, \left[\frac{37}{5}\right]\right\} = 7.4$$

$$t = 0;$$

e as folgas por dia:

$$b'_j = 6 - r_j,\ j = 1, \ldots, 7;$$

$$b'_1, \ldots, b'_7 = 0, 0, 2, 0, 0, 1, 2;$$

Da equação (4.15), tem-se:

$$x_1, \ldots, x_7 = 2, 5, 4, 3, 4, 4, 5;$$

Devido ao fato de x_{min} ser positivo, resulta neste caso, ser indiferente a utilização de quaisquer das equações 4.8 e 4.13 para o cálculo do nível mínimo de escalação. Recomenda-se sempre o uso do algoritmo revisado de Baker (equação 4.13), assim:

$$W = 7.4 + \max\{0, [(-2-0)/3]\} = 7.4$$

Portanto, será próximo de 8 o número mínimo das categorias ou da força de trabalho para atender o definido plano de operações e estes serão chamados de A até H.

5.2.4 Geração da Programação Inicial (Algoritmo 2)

Definidos os elementos principais do sistema (número, tipo e extensão dos turnos- tabela 5.1), assim como o plano de operação (tabela 5.2) e realizado o cálculo do mínimo número de trabalhadores/categorias, algoritmo 1, seção anterior, segue a geração através do algoritmo 2, seção 4.6 e figuras 4.4 e 4.5, da programação inicial de turnos para a demanda em estudo e definida na tabela 5.2. Segue a avaliação desta programação inicial em função das restrições próprias do sistema e formuladas na seção 4.7.2

Tabela 5.3 – Programação Inicial

	Semana 1							Semana 2							Semana 3							Semana 4						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Categorias	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.
A	3	0	1	0	2	0	5	1	0	2	1	0	4	0	3	2	1	1	2	5	0	0	0	0	2	1	5	0
B	2	3	1	1	2	0	5	1	0	0	2	1	4	0	0	3	2	1	0	5	0	2	1	0	1	2	5	0
C	2	1	0	3	2	0	5	1	1	0	2	0	0	0	0	2	1	3	0	4	5	2	1	1	0	2	4	5
D	1	2	0	0	3	4	5	2	1	1	0	2	0	0	1	2	1	0	3	0	5	1	2	0	0	2	4	5
E	1	2	0	1	0	4	0	3	2	1	0	2	5	5	1	0	0	2	1	0	0	3	2	1	1	0	4	5
F	0	1	2	1	0	4	0	2	3	0	1	2	5	5	1	0	0	2	1	0	0	1	3	2	1	0	0	5
G	1	2	0	2	1	5	0	0	2	1	1	3	4	5	2	1	0	0	2	4	5	1	0	0	2	3	0	0
H	0	0	1	2	1	5	0	0	2	0	3	1	0	5	2	1	0	1	2	4	5	0	2	1	3	1	0	0

	Semana 5							Semana 6							Semana 7							Semana 8						
	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
Categorias	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.
A	3	2	1	1	0	4	5	2	0	1	0	2	4	0	3	1	0	2	1	5	0	1	2	0	1	2	5	0
B	0	3	2	1	1	4	5	0	2	1	0	2	4	0	1	3	0	2	0	5	0	0	2	1	0	1	5	0
C	1	0	1	3	2	0	5	1	2	0	1	0	0	0	2	0	1	3	2	4	5	1	0	2	1	0	0	5
D	1	0	0	2	3	0	5	1	2	1	1	0	0	0	2	0	1	0	3	4	5	2	1	1	2	0	0	5
E	2	1	0	2	0	0	0	3	1	2	1	1	5	5	0	2	0	0	2	4	0	3	1	0	2	1	0	5
F	2	1	0	0	2	4	0	1	3	0	2	1	5	5	0	2	1	1	2	0	0	0	3	0	1	2	4	5
G	1	2	1	0	1	5	0	2	0	0	2	3	0	5	1	1	2	1	0	0	5	1	2	0	0	3	4	0
H	0	2	0	1	2	5	0	0	1	0	3	2	4	5	1	2	0	1	1	0	5	2	0	1	3	2	4	0

Notação:
0: Folga
1: TM Turno de Manhã
2: TD Turno de Tarde
3: TN Turno de Noite
4: TFS4 Turno de Fim-de-Semana
5: TFS12 Turno de Fim-de-Semana

5.2.4.1 Avaliação da Programação Inicial

Restrição da Distribuição Uniforme das Horas de Trabalho (DiferençaHoras)

Avalia-se aqui a programação inicial em relação à diferença de horas trabalhadas semanalmente por cada categoria, a fim de permitir uma seqüência de turnos mais uniformemente distribuída, melhorando assim a qualidade da programação, tal como foi definido na seção 4.7.2.1. Este cálculo é representado na tabela 5.4, onde se consideram violações a esta restrição os valores ≥ 24 e < -24 horas e indicados na tabela por um retângulo sombreado.

Tabela 5.4 - Diferença de Horas Trabalhadas Semanalmente por Categoria

Categorias	Sem.2 -Sem.1	Sem.3-Sem.2	Sem.4-Sem.3	Sem.5-Sem.4	Sem.6-Sem.5	Sem.7-Sem.6	Sem.8-Sem.7
A	-8	24	-24	20	-20	16	0
B	-24	8	8	4	-20	8	0
C	-20	16	8	-4	-20	24	-12
D	-8	12	-4	-4	-4	8	4
E	28	-32	24	-24	40	-44	24
F	28	-32	20	-16	28	-24	8
G	4	-8	-16	20	-8	8	-16
H	0	12	-16	4	4	4	-8

Restrição das Horas Trabalhadas (HorasTrabalho)

Para a verificação desta restrição, calcula-se o total de horas trabalhadas semanalmente por categorias ou por empregados. Como foi definido na seção 4.7.2.2, um empregado ou categoria não deverá exceder o total de 50 horas de trabalho semanais, indicando em um quadro sombreado na tabela 5.5 as categorias e as semanas que infringem esta restrição.

Tabela 5.5 - Total de Horas Trabalhadas Semanalmente por Categoria

Categorias	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6	Semana 7	Semana 8
A	36	28	52	28	48	28	44	44
B	52	28	36	44	48	28	36	36
C	44	24	40	48	44	24	48	36
D	40	32	44	40	36	32	40	44
E	28	56	24	48	24	64	20	44
F	28	56	24	44	28	56	32	40
G	44	48	40	24	44	36	44	28
H	36	36	48	32	36	40	44	36

Restrição para os Finais de Semana (FinaisSemana)

A programação inicial é avaliada em seus finais de semana, de modo a prevenir que uma categoria ou trabalhador seja alocado durante os quatro dias correspondentes a dois finais de semana seguidos, segundo o formulado na seção 4.7.2.4. O cálculo em relação a esta restrição para a programação inicial apresentada é mostrado na tabela 5.6 onde está detalhada também a violação por um quadro sombreado.

Tabela 5.6 - Cálculo sobre os Finais de Semana

	Fin.de-Sem. 1,2	Fin.de-Sem. 2,3	Fin.de-Sem. 3,4	Fin.de-Sem. 4,5	Fin.de-Sem. 5,6	Fin.de-Sem. 6,7	Fin.de-Sem. 7,8
Categorias							
A	2	2	2	3	3	2	2
B	2	2	2	3	3	2	2
C	1	2	4	3	1	2	3
D	2	1	3	3	1	2	3
E	3	2	2	2	2	3	2
F	3	2	1	2	3	2	2
G	3	4	2	1	2	2	2
H	2	3	2	1	3	3	2

Restrição para Evitar Seqüência de Turnos Excedentes (SeqüênciaTurnos)

Avalia-se a programação inicial, identificando os turnos consecutivos a fim de evitar seqüências acima de 6 turnos para um mesmo empregado ou categoria, como o formulado na seção 4.7.2.3. A tabela 5.7 mostra a revisão desta programação e detalha em um quadro sombreado aquelas seqüências que estão infringindo esta restrição.

Restrição para Evitar Alocações Consecutivas sem Folga (SeqüênciaFolgas)

Verifica-se da programação inicial, a folga existente entre alocações consecutivas dos turnos para cada empregado. Com isto, espera-se evitar alocações diurnas seguidas de noturnas para um mesmo empregado ou categoria, como formulado na seção 4.7.2.5. A tabela 5.8 detalha as violações a esta restrição (quadros sombreados) referentes à programação inicial gerada e analisada.

Tabela 5.7 – Identificação de Turnos Consecutivos

	Semana 1							Semana 2							Semana 3							Semana 4						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Categorias	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.
A	3	0	1	0	2	0	5	1	0	2	1	0	4	0	3	2	1	1	2	5	0	0	0	0	2	1	5	0
B	2	3	1	1	2	0	5	1	0	0	2	1	4	0	0	3	2	1	0	5	0	2	1	0	1	2	5	0
C	2	1	0	3	2	0	5	1	1	0	2	0	0	0	0	2	1	3	0	4	5	2	1	1	0	2	4	5
D	1	2	0	0	3	4	5	2	1	1	0	2	0	0	1	2	1	0	3	0	5	1	2	0	0	2	4	5
E	1	2	0	1	0	4	0	3	2	1	0	2	5	5	1	0	0	2	1	0	0	3	2	1	1	0	4	5
F	0	1	2	1	0	4	0	2	3	0	1	2	5	5	1	0	0	2	1	0	0	1	3	2	1	0	0	5
G	1	2	0	2	1	5	0	0	2	1	1	3	4	5	2	1	0	0	2	4	5	1	0	0	2	3	0	0
H	0	0	1	2	1	5	0	0	2	0	3	1	0	5	2	1	0	1	2	4	5	0	2	1	3	1	0	0

	Semana 5							Semana 6							Semana 7							Semana 8						
	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
Categorias	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.
A	3	2	1	1	0	4	5	2	0	1	0	2	4	0	3	1	0	2	1	5	0	1	2	0	1	2	5	0
B	0	3	2	1	1	4	5	0	2	1	0	2	4	0	1	3	0	2	0	5	0	0	2	1	0	1	5	0
C	1	0	1	3	2	0	5	1	2	0	1	0	0	0	2	0	1	3	2	4	5	1	0	2	1	0	0	5
D	1	0	0	2	3	0	5	1	2	1	1	0	0	0	2	0	1	0	3	4	5	2	1	1	2	0	0	5
E	2	1	0	2	0	0	0	3	1	2	1	1	5	5	0	2	0	0	2	4	0	3	1	0	2	1	0	5
F	2	1	0	0	2	4	0	1	3	0	2	1	5	5	0	2	1	1	2	0	0	0	3	0	1	2	4	5
G	1	2	1	0	1	5	0	2	0	0	2	3	0	5	1	1	2	1	0	0	5	1	2	0	0	3	4	0
H	0	2	0	1	2	5	0	0	1	0	3	2	4	5	1	2	0	1	1	0	5	2	0	1	3	2	4	0

Notação:
0: Folga
1: TM Turno de Manhã
2: TD Turno de Tarde
3: TN Turno de Noite
4: TFS4 Turno de Fim-de-Semana
5: TFS12 Turno de Fim-de-Semana

Tabela 5.8 - Identificação de Seqüências de Folgas

	Semana 1							Semana 2							Semana 3							Semana 4						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Categorias	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.
A	3	0	1	0	2	0	5	1	0	2	1	0	4	0	3	2	1	1	2	5	0	0	0	0	2	1	5	0
B	2	3	1	1	2	0	5	1	0	0	2	1	4	0	0	3	2	1	0	5	0	2	1	0	1	2	5	0
C	2	1	0	3	2	0	5	1	1	0	2	0	0	0	0	2	1	3	0	4	5	2	1	1	0	2	4	5
D	1	2	0	0	3	4	5	2	1	1	0	2	0	0	1	2	1	0	3	0	5	1	2	0	0	2	4	5
E	1	2	0	1	0	4	0	3	2	1	0	2	5	5	1	0	0	2	1	0	0	3	2	1	1	0	4	5
F	0	1	2	1	0	4	0	2	3	0	1	2	5	5	1	0	0	2	1	0	0	1	3	2	1	0	0	5
G	1	2	0	2	1	5	0	0	2	1	1	3	4	5	2	1	0	0	2	4	5	1	0	0	2	3	0	0
H	0	0	1	2	1	5	0	0	2	0	3	1	0	5	2	1	0	1	2	4	5	0	2	1	3	1	0	0

	Semana 5							Semana 6							Semana 7							Semana 8						
	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
Categorias	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.
A	3	2	1	1	0	4	5	2	0	1	0	2	4	0	3	1	0	2	1	5	0	1	2	0	1	2	5	0
B	0	3	2	1	1	4	5	0	2	1	0	2	4	0	1	3	0	2	0	5	0	0	2	1	0	1	5	0
C	1	0	1	3	2	0	5	1	2	0	1	0	0	0	2	0	1	3	2	4	5	1	0	2	1	0	0	5
D	1	0	0	2	3	0	5	1	2	1	1	0	0	0	2	0	1	0	3	4	5	2	1	1	2	0	0	5
E	2	1	0	2	0	0	0	3	1	2	1	1	5	5	0	2	0	0	2	4	0	3	1	0	2	1	0	5
F	2	1	0	0	2	4	0	1	3	0	2	1	5	5	0	2	1	1	2	0	0	0	3	0	1	2	4	5
G	1	2	1	0	1	5	0	2	0	0	2	3	0	5	1	1	2	1	0	0	5	1	2	0	0	3	4	0
H	0	2	0	1	2	5	0	0	1	0	3	2	4	5	1	2	0	1	1	0	5	2	0	1	3	2	4	0

Notação:

0:	Folga	2:	TD Turno de Tarde	4:	TFS4 Turno de Fim-de-Semana
1:	TM Turno de Manhã	3:	TN Turno de Noite	5:	TFS12 Turno de Fim-de-Semana

5.2.5 Programação Final (Algoritmo 3)

Gerada e avaliada a programação inicial, proceder-se-á à melhora da qualidade desta programação visando o cumprimento de todas as restrições definidas para o processo em estudo. Para isto, formularam-se estas restrições como um problema de satisfação por restrições nebulosas (MPSRN), segundo o algoritmo 3 da figura 4.7, seção 4.7; e, baseados nas técnicas de melhorias iterativas representadas pelo algoritmo fundamentado no método por reparo em profundidade (figura 4.22) e pelo algoritmo baseado no método por reparos por modificações aleatórias (figura 4.24), "repara-se" iterativamente a programação inicial até que sejam satisfeitas as restrições ou até que seja definido um critério de finalização da busca da solução.

A figura 5.2 mostra resultados empíricos comparando o desempenho entre uma heurística de reparo em profundidade *versus* uma heurística de reparo por modificações aleatórias para nossa aplicação em estudo, cada uma com 3 seqüências iterativas diferentes (experimentos), que começam com a mesma solução sub-ótima inicial. Diferentes experimentos foram necessários para representar a variabilidade dos resultados obtidos com a busca aleatória.

Posto que estas curvas não diferem muito entre si, pode-se concluir que o método de otimização apresentado é bastante "robusto" e competente para encontrar soluções adequadas. Nota-se nesta mesma figura que o método de reparo em profundidade apresenta um melhor desempenho com soluções consideradas "ótimas" (cumprimento de todas as restrições) em termos do tempo efetivo de processamento a partir de uma função de avaliação ao redor de 24000 unidades em um tempo de 10 minutos para uma primeira solução. A técnica de reparo por modificações aleatórias "pega" com maior facilidade soluções sub-ótimas, não conseguindo, no mesmo horizonte de tempo em comparação à heurística de reparo em profundidade, o cumprimento de todas as restrições.

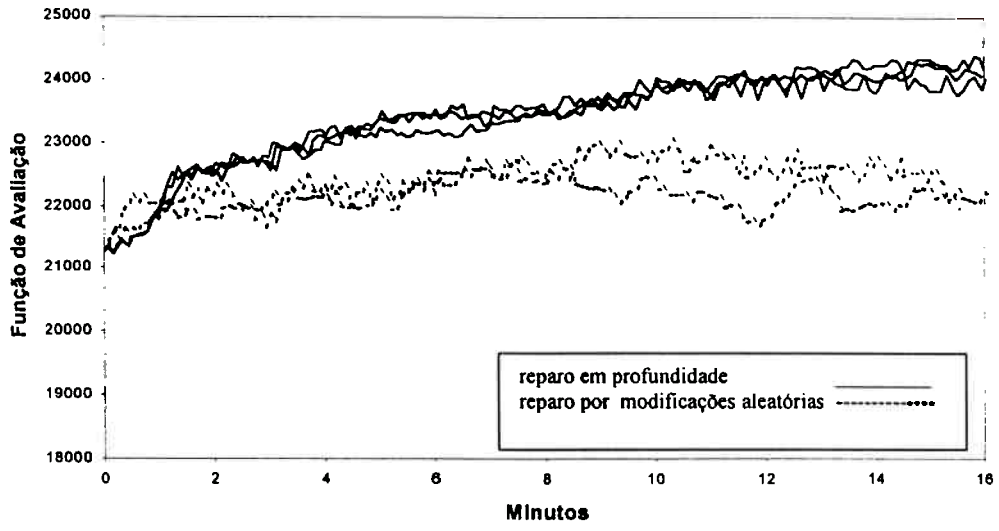


Figura 5.2 – Desempenho do Método por Reparos em Profundidade *versus* o Método por Reparos por Modificações Aleatórias

Os resultados da otimização da figura 5.3, cujos dados importantes são resumidos na tabela 5.9, focalizam-se na técnica de reparo em profundidade. A figura representa 3 experimentos diferentes com a mesma base de dados (tabela 5.2). A tabela 5.9 mostra as estatísticas das soluções encontradas para cada experimento nos primeiros 14 minutos de processamento. A primeira solução ótima foi encontrada com uma avaliação de 24011,3 unidades em um tempo de processamento de 10 minutos e com 107 avaliações completas da programação de turnos, em um Pentium II, 400 MHz, com 64 MB de memória, sobre sistema operacional LINUX, configuração utilizada para todos os experimentos.

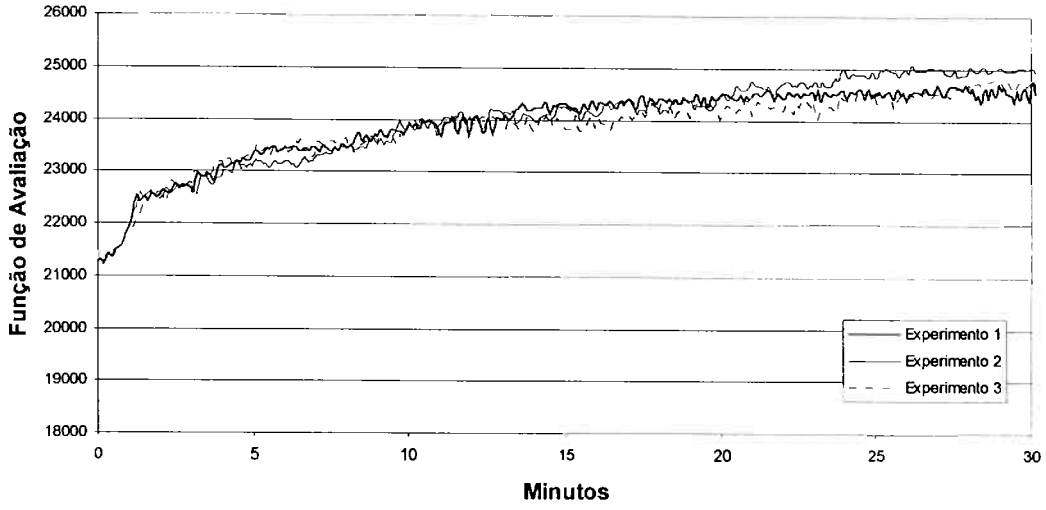


Figura 5.3 – Resultados da Otimização (Método de Reparo em Profundidade)

Tabela 5.9 – Resultados da Otimização, Estatísticas das Soluções dos Experimentos (vide figura 5.3)

Tempo de Processamento (minutos)	Experimentos	No. Avaliações à Programação (iterações)	Função de Avaliação	No. de Restrições Violadas				
				Finas Semanas	Diferença Horas	Seqüência Folgas	Seqüência Turnos	Horas Trabalho
0,0	Programação Inicial	1	21275,2	2	11	6	3	6
10,0	Experimento 1	106	23849,2	0	0	2	5	1
	Experimento 2	108	23962,5	1	0	0	2	0
	Experimento 3	107	24011,3	0	0	0	0	0
10,1	Experimento 1	107	23913,6	0	0	2	5	0
	Experimento 2	109	23818,7	0	1	0	2	1
	Experimento 3	108	23972,9	0	0	0	0	0
10,2	Experimento 1	108	23848,2	0	0	1	6	1
	Experimento 2	110	23784,5	0	1	0	1	1
	Experimento 3	109	23961,3	0	0	0	0	0
11,0	Experimento 1	116	23823,1	0	1	1	5	1
	Experimento 2	118	23934,6	0	0	0	0	0
	Experimento 3	117	23769,9	1	0	0	1	2
11,3	Experimento 1	119	23944,2	1	0	1	4	0
	Experimento 2	121	24014,0	0	0	1	3	0
	Experimento 3	119	23955,0	0	0	0	0	0
11,5	Experimento 1	121	23887,8	0	1	0	5	1
	Experimento 2	123	24049,7	0	0	1	4	0
	Experimento 3	121	24076,7	0	0	0	0	0
11,6	Experimento 1	122	23721,2	0	0	0	5	2
	Experimento 2	124	24138,5	0	0	1	4	0
	Experimento 3	122	24076,7	0	0	0	0	0
11,7	Experimento 1	123	23981,3	1	0	0	3	1
	Experimento 2	125	23937,2	0	3	1	3	2
	Experimento 3	123	24050,4	0	0	0	0	0
12,2	Experimento 1	128	24080,8	0	0	0	4	0
	Experimento 2	129	24091,9	0	0	1	4	1
	Experimento 3	128	24027,2	0	0	0	0	0
12,7	Experimento 1	133	23689,5	0	1	0	3	1
	Experimento 2	134	24175,5	0	0	1	3	0
	Experimento 3	133	24076,7	0	0	0	0	0
13,2	Experimento 1	138	23990,8	0	0	0	3	1
	Experimento 2	139	24083,4	0	0	1	4	0
	Experimento 3	138	24017,6	0	0	0	0	0
13,5	Experimento 1	141	24198,3	0	0	0	2	0
	Experimento 2	142	23984,7	0	0	1	3	2
	Experimento 3	141	23834,6	0	0	0	0	0
14,0	Experimento 1	146	24232,4	0	0	0	0	0
	Experimento 2	147	23968,6	0	1	1	3	1
	Experimento 3	146	23778,1	1	1	0	2	1

Nas figuras 5.4 até 5.8 observam-se as curvas de progresso sobre a melhoria da qualidade da programação para cada uma das restrições analisadas, em função do número de restrições violadas *versus* tempo. Utiliza-se o método de reparo em profundidade por apresentar um melhor desempenho frente ao método por modificações aleatórias. Vê-se nestas figuras a falta de continuidade na melhoria de cada restrição, o que expressa a natureza conflitante entre elas, umas da principais fontes de dificuldades no projeto de boas programações. Conclui-se também que é totalmente possível satisfazer todas as restrições simultaneamente.

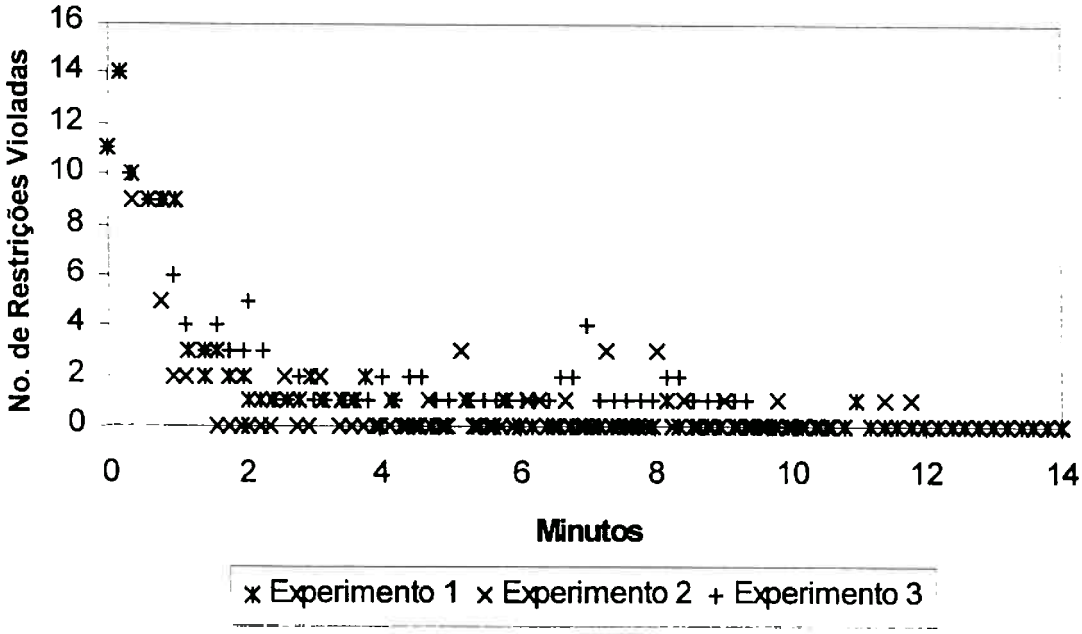


Figura 5.4 - Restrição da Distribuição Uniforme das Horas de Trabalho

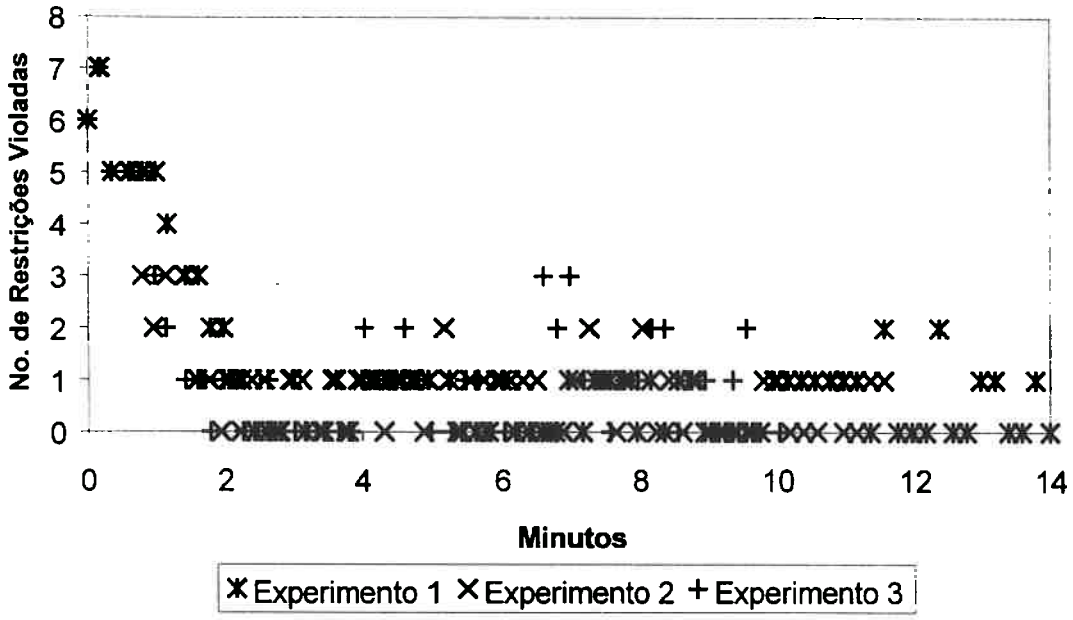


Figura 5.5 - Restrição das Horas Trabalhadas

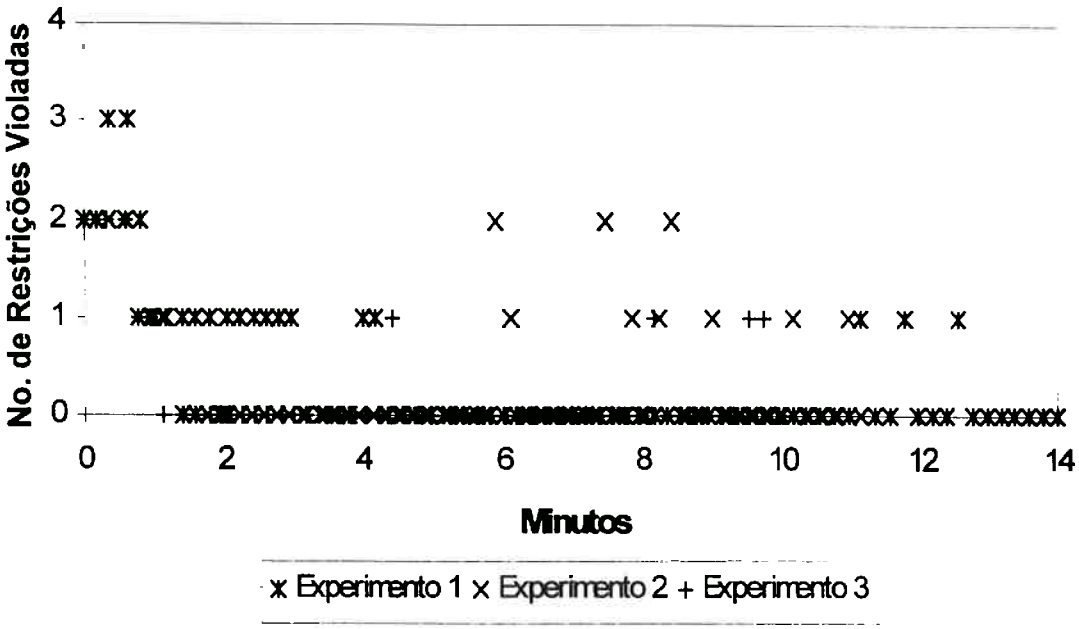


Figura 5.6 - Restrição para os Finais-de-Semana

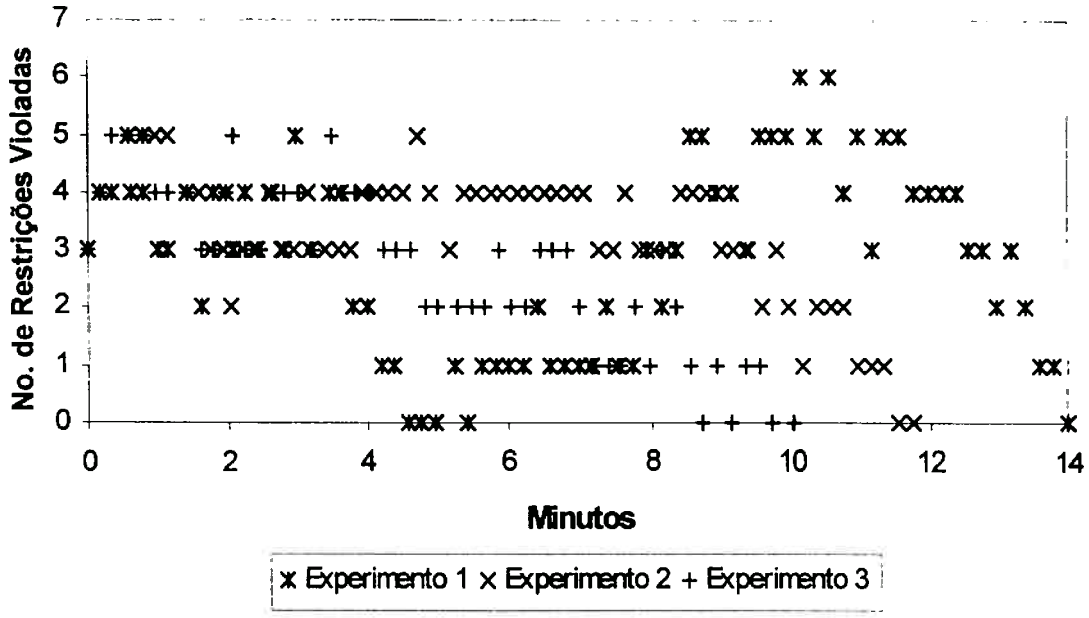


Figura 5.7 - Restrição para Evitar Seqüências de Turnos Excedentes

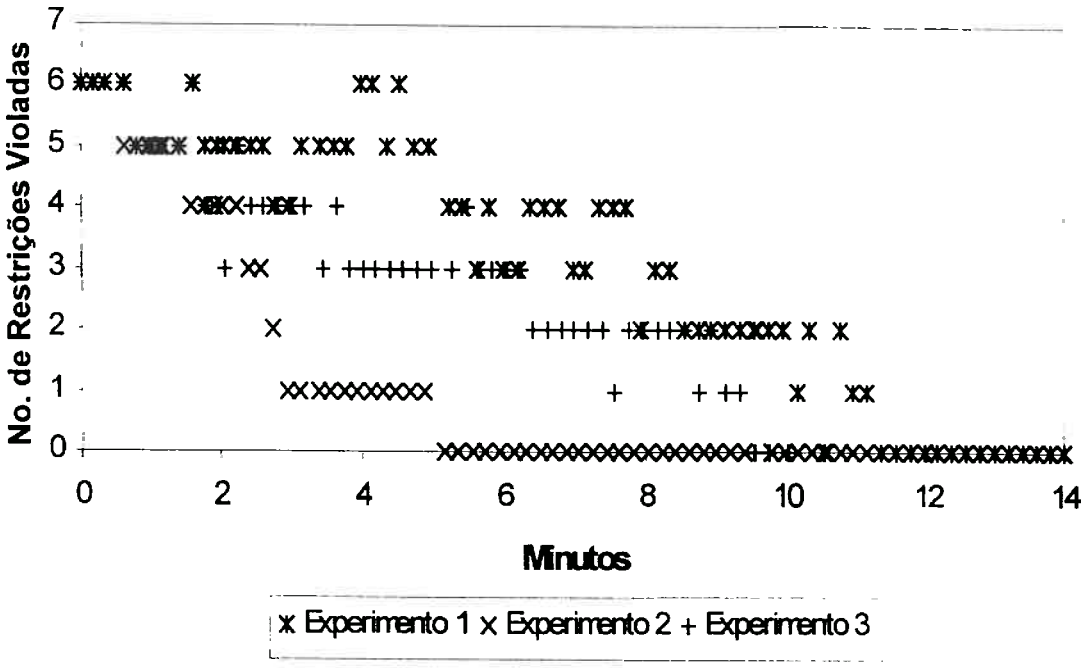


Figura 5.8 - Restrição para Evitar Alocações Consecutivas sem Folga

A tabela 5.10 apresenta uma primeira solução de uma programação de turnos do exemplo estudado, onde são satisfeitas todas as restrições.

Tabela 5.10 - Programação Final

	Semana 1							Semana 2							Semana 3							Semana 4						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Categorias	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.
A	3	0	1	1	2	0	5	1	2	0	1	2	0	5	3	0	1	1	2	4	0	2	0	1	2	3	0	0
B	2	3	0	0	0	4	5	0	1	1	2	1	5	0	1	3	0	1	1	5	0	2	0	0	0	2	5	0
C	2	1	0	3	2	0	5	1	0	1	2	0	4	5	0	2	1	3	2	0	5	0	2	1	1	2	0	5
D	0	2	1	2	3	0	5	0	1	1	1	2	4	0	1	2	0	1	3	0	5	1	2	0	1	0	4	5
E	1	2	0	1	0	4	0	1	0	2	3	2	0	0	1	0	2	2	0	5	0	3	3	0	1	1	4	5
F	1	1	2	1	2	4	0	2	2	0	0	0	5	5	0	1	1	2	1	4	0	1	1	0	0	2	0	5
G	1	2	0	0	1	5	0	2	3	0	1	3	4	0	2	2	0	0	0	4	5	0	1	2	2	1	5	0
H	0	0	1	2	1	5	0	3	2	0	0	1	0	5	2	1	0	0	2	0	5	1	2	1	3	0	4	0

	Semana 5							Semana 6							Semana 7							Semana 8						
	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
Categorias	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.
A	3	0	0	0	1	4	5	2	0	1	0	2	0	5	0	1	1	2	0	4	5	1	2	0	1	2	5	0
B	1	3	0	1	2	4	0	2	2	1	1	2	4	0	1	0	1	2	3	5	0	2	2	1	0	0	5	0
C	0	2	0	3	2	0	5	1	2	0	1	1	4	0	0	1	1	3	0	4	0	0	1	2	0	1	0	5
D	1	0	1	2	3	0	5	1	2	0	1	2	0	0	2	3	0	0	2	0	5	0	0	1	2	0	4	5
E	0	1	1	2	1	0	5	3	0	0	2	0	5	5	1	2	2	0	2	4	0	1	1	0	2	1	0	5
F	2	1	2	0	0	4	0	1	3	0	0	1	5	0	3	2	0	1	2	5	0	3	3	0	1	2	0	5
G	1	2	1	1	0	5	0	0	1	2	2	3	0	5	2	0	0	1	1	0	5	1	2	0	1	3	4	0
H	2	2	0	1	2	5	0	0	1	1	3	0	4	5	1	2	0	1	1	0	5	2	0	1	3	2	4	0

Notação:
0: Folga
1: TM Turno de Manhã
2: TD Turno de Tarde
3: TN Turno de Noite
4: TFS4 Turno de Fim-de-Semana
5: TFS12 Turno de Fim-de-Semana

5.3 Procedimento para a Programação Dinâmica de Turnos (SIPRODT)

Estabelecidos os elementos principais do sistema como base de dados para um particular terminal e o nível de operários, é possível agora definir a programação de turnos e seus respetivos ajustes cada vez que for necessário, seja por mudanças na demanda de serviço ou por acontecimentos eventuais que afetem a disponibilidade dos empregados. O procedimento segue o algoritmo 1 da figura 4.1, seção 4.1, onde o único dado de entrada seria o novo plano de operações contendo as correspondentes correções. Na continuação, apresentar-se-á a descrição e solução de um problema exemplo com 3 diferentes demandas.

Descrição do Problema

O objetivo do problema é encontrar a programação de turnos para um terminal com número inicial de 36 empregados. Estes empregados são agregados uniformemente em 9 categorias, cada uma com 4 trabalhadores. Foi assumido que cada categoria cobriria em sua totalidade os requerimentos das operações, portanto, não há interdependências entre estas.

Com os elementos do sistemas definidos na tabela 5.1 e para 3 diferentes planos de operações seqüenciais descritos na tabela 5.11 até 5.13, calcula-se o número mínimo de categorias necessárias e constroe-se a programação de turnos para cada uma destas demandas. Requere-se também o cálculo do número de horas e turnos acumulados para cada categoria no final de atendimento para a última demanda de operações.

Tabela 5.11 - Definição do Plano de Operação 1

2ª-feira	3ª-feira	4ª-feira	5ª-feira	6ª-feira	Sábado	Domingo
2 – TM	1 – TM	2 – TM	2 – TM	2 – TM	3 – TFS4	4 – TFS12
2 – TD	2 – TD	1 – TD	1 – TD	1 – TD	1 – TFS12	
1 – TN	1 – TN	1 – TN	1 – TN	1 – TN		

Tabela 5.12 - Definição do Plano de Operação 2

2ª-feira	3ª-feira	4ª-feira	5ª-feira	6ª-feira	Sábado	Domingo
3 – TM	4 – TM	3 – TM	3 – TM	3 – TM	3 – TFS4	4 – TFS12
2 – TD	2 – TD	2 – TD	2 – TD	3 – TD	2 – TFS12	
1 – TN	1 – TN		1 – TN			

Tabela 5.13 - Definição do Plano de Operação 3

2ª-feira	3ª-feira	4ª-feira	5ª-feira	6ª-feira	Sábado	Domingo
2 – TM	3 – TM	2 – TM	2 – TM	3 – TM	2 – TFS4	3 – TFS12
3 – TD	2 – TD	1 – TD	3 – TD	2 – TD	2 – TFS12	
1 – TN		1 – TN	1 – TN	1 – TN		

Solução

A figura 5.9 mostra os resultados da otimização para o primeiro plano de operações. Das 9 categorias disponíveis só foram necessárias 6 para o atendimento da demanda. A primeira solução foi encontrada para o experimento 2 a 2.3 minutos de processamento. A tabela 5.14 detalha o número de restrições violadas na programação inicial, as funções de avaliação, o número de avaliações da programação inicial até encontrar uma solução e o tempo desta para o melhor experimento. Os experimentos 1 e 3 encontraram soluções ótimas a 3 e 2.6 minutos, respectivamente.

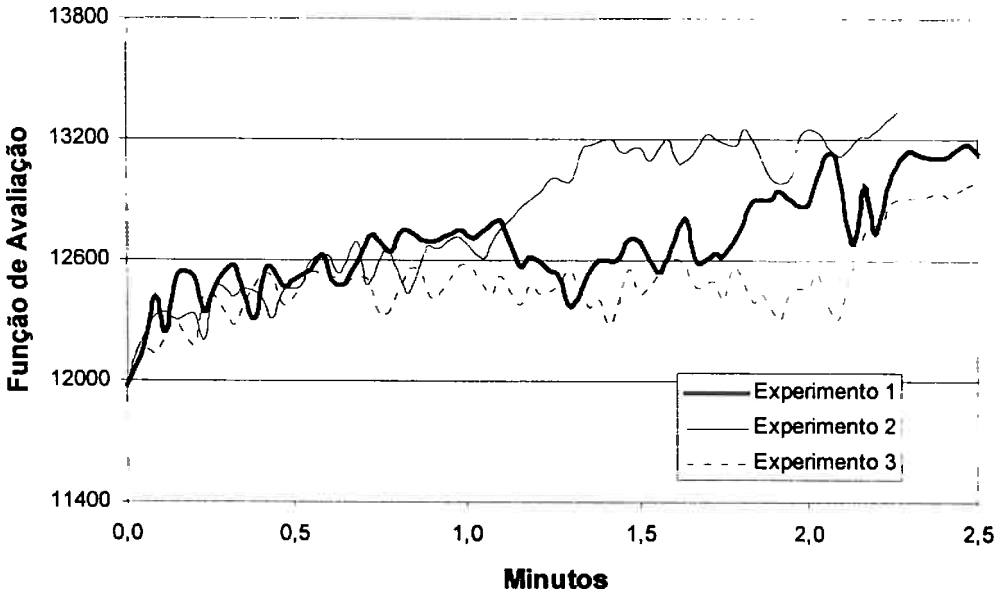


Figura 5.9 – Resultados da Otimização (Plano de Operações 1 – Tabela 5.11)

Tabela 5.14 - Resultados da Otimização do Melhor Experimento (vide figura 5.9)

	Tempo de Processamento (minutos)	Categorias	Experimentos	No. Avaliações à Programação (iterações)	Função de Avaliação	No. de Restrições Violadas				
						Finas Semanas	Diferença Horas	Seqüência Folgas	Seqüência Turnos	Horas Trabalho
	0,0	6	Programação Inicial	1	11976,4	6	1	7	1	2
Solução	2,3	6	Programação Final	57	13344,9	0	0	0	0	0

A programação final para o plano de operações 1 é apresentada na tabela 5.15. O número de dias de folgas e de turnos por categoria resultados da programação é apresentado na tabela 5.16.

Tabela 5.15 – Programação Final (Plano de Operações 1 – Tabela 5.11)

	Semana 1							Semana 2							Semana 3						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Categorias	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.
A	3	2	0	0	0	4	5	0	3	0	2	1	4	0	0	2	1	1	1	5	0
B	2	0	1	1	2	0	5	2	0	0	1	2	4	5	1	3	0	3	2	0	5
C	2	1	3	3	0	4	5	2	1	1	3	0	5	0	2	0	1	2	3	0	5
D	1	0	0	2	1	0	5	3	0	2	1	1	0	5	1	0	2	0	0	4	5
E	1	2	1	0	1	4	0	1	2	1	0	3	0	5	2	1	3	0	0	4	5
F	0	3	2	1	3	5	0	1	2	3	0	0	4	5	3	2	0	1	1	4	0

	Semana 4							Semana 5							Semana 6						
	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
Categorias	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.
A	3	2	1	3	0	5	0	1	2	0	1	3	5	0	2	2	0	1	1	5	0
B	0	3	2	1	3	0	5	1	2	1	0	0	4	5	3	0	1	1	0	4	0
C	1	0	1	2	0	4	5	0	0	1	2	1	4	0	1	3	2	0	0	0	5
D	2	1	3	0	1	4	0	2	1	3	3	0	4	5	2	1	0	2	3	0	5
E	1	2	0	0	2	0	5	3	3	2	0	1	0	5	0	2	3	0	1	4	5
F	2	0	0	1	1	4	5	2	0	0	1	2	0	5	1	0	1	3	2	4	5

Notação:

0: Folga
1: TM Turno de Manhã
2: TD Turno de Tarde
3: TN Turno de Noite
4: TFS4 Turno de Fim-de-Semana
5: TFS12 Turno de Fim-de-Semana

Tabela 5.16 – Total de Turnos por Tipo e Horas por Categoria (Plano de Operações 1 – Tabela 5.11)

		Turnos Programação					Total Horas
Categorias	Folgas	TM	TD	TN	TFS4	TFS12	Programação
A	14	9	7	5	2	5	236
B	13	9	7	5	3	5	240
C	13	9	7	5	3	5	240
D	13	9	7	5	3	5	240
E	13	9	7	5	3	5	240
F	12	9	7	5	4	5	244

Os resultados da otimização por reparos da programação inicial para o plano de operações 2 estão representados na figura 5.10 e resumidos na tabela 5.17 para o melhor experimento. Neste caso especial, o processo iterativo de SIPRODT incrementou mais uma categoria da calculada pelo algoritmo 2, encontrando uma solução ótima a 26.3 minutos em 220 avaliações completas da programação e para 9 categorias. Novamente o experimento 2 apresentou o melhor desempenho; soluções ótimas também foram encontradas em 33.9 e 26.3 minutos para os experimentos 1 e 3, respectivamente.

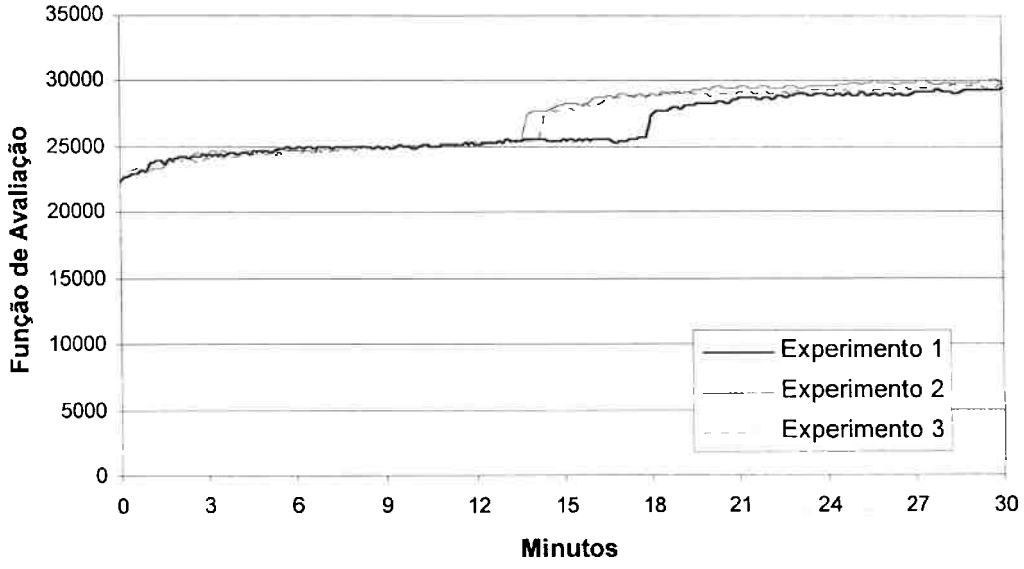


Figura 5.10 – Resultados da Otimização (Plano de Operações 2 – Tabela 5.11)

Tabela 5.17 - Resultados da Otimização do Melhor Experimento (vide figura 5.10)

	Tempo de Processamento (minutos)	Categorias	Experimentos	No. Avaliações à Programação (iterações)	Função de Avaliação	No. de Restrições Violadas				
						Finas Semanas	Diferença Horas	Seqüência Folgas	Seqüência Turnos	Horas Trabalho
	0,0	8	Programação Inicial	1	22402,3	2	10	8	10	12
	13,7	9	Programação Inicial	134	27352,0	0	11	11	2	7
Solução	26,3	9	Programação Final	220	29858,5	0	0	0	0	0

A programação final, satisfazendo todas as restrições do sistema e para a melhor solução acima, pode ser vista na tabela 5.18. A tabela 5.19 apresenta o número de folgas e turnos por categoria, observa-se a distribuição uniforme dos mesmos no final da programação. A tabela 5.20 mostra o acumulado de turnos e horas trabalhadas para todas as 9 categorias disponíveis inicialmente para a programação.

Tabela 5.18 - Programação Final (Plano de Operações 2 – Tabela 5.12)

	Semana 1							Semana 2							Semana 3						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Categorias	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.
A	2	0	1	1	1	0	5	2	1	0	0	2	4	0	0	2	1	0	2	5	0
B	3	3	0	1	2	0	5	1	2	2	2	1	0	0	1	1	0	1	0	5	0
C	0	0	1	3	0	0	5	2	1	0	3	2	0	0	1	1	0	1	2	4	5
D	1	2	1	1	0	0	5	0	2	1	0	2	0	0	1	1	0	0	2	4	5
E	1	2	2	0	1	4	0	3	3	0	1	0	5	0	2	1	0	0	1	0	5
F	1	1	2	0	0	4	0	0	0	1	0	0	5	5	2	0	2	1	1	0	0
G	0	1	0	2	1	4	0	0	1	2	1	1	0	5	3	0	1	2	1	0	5
H	2	1	0	2	2	5	0	1	1	0	1	1	4	5	0	3	2	2	0	0	0
I	0	1	0	0	2	5	0	1	0	1	2	0	4	5	0	2	1	3	0	4	0

	Semana 4							Semana 5							Semana 6						
	22	23	24	25	26	27	28	29	30	31	32	33	34	35	29	30	31	32	33	34	35
Categorias	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.
A	1	3	0	1	0	0	5	1	2	1	1	0	4	0	2	1	2	0	0	5	0
B	3	2	1	1	0	4	5	2	1	1	0	2	4	0	1	0	0	2	1	5	0
C	1	0	2	0	1	0	5	0	2	1	1	1	4	0	2	1	0	0	1	0	5
D	1	0	0	2	1	0	5	0	1	2	3	0	0	0	0	2	1	0	0	4	5
E	2	1	0	2	1	0	0	0	3	2	0	1	5	0	0	2	1	1	0	4	5
F	2	1	0	3	2	4	0	3	0	0	0	2	5	5	0	1	2	1	1	0	0
G	0	1	2	0	2	4	0	1	0	0	2	1	0	5	3	0	0	1	2	0	5
H	0	1	1	0	2	5	0	1	1	0	2	0	0	5	1	3	0	2	2	4	0
I	0	2	1	1	0	5	0	2	1	0	1	2	0	5	1	1	1	3	2	0	0

	Semana 7							Semana 8							Semana 9						
	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
Categorias	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.
A	3	3	0	2	1	0	0	0	1	2	1	0	4	5	0	2	1	1	0	5	0
B	0	1	0	2	1	0	5	0	1	2	0	0	4	0	0	2	0	0	1	5	0
C	2	1	0	0	2	0	5	1	2	1	2	2	0	0	3	0	0	1	0	4	5
D	1	2	1	0	2	0	5	3	3	0	2	1	0	0	1	1	2	1	0	4	5
E	0	2	0	1	2	4	0	1	0	1	0	2	5	0	1	1	1	0	2	0	5
F	0	1	2	0	1	4	5	0	1	1	3	2	5	0	1	1	1	2	2	0	0
G	2	1	1	3	0	4	0	2	1	0	1	0	0	5	0	0	2	2	1	0	5
H	1	0	1	1	0	5	0	2	0	0	1	1	4	5	2	3	0	0	1	0	0
I	1	0	2	1	0	5	0	1	2	0	0	1	0	5	2	1	0	3	2	4	0

Notação:

0: Folga
1: TM Turno de Manhã
2: TD Turno de Tarde
3: TN Turno de Noite
4: TFS4 Turno de Fim-de-Semana
5: TFS12 Turno de Fim-de-Semana

Tabela 5.19 – Total de Turnos por Tipo e Horas por Categoria (Plano de Operações 2 – Tabela 5.12)

		Turnos Programação					Total Horas
Categorias	Folgas	TM	TD	TN	TFS4	TFS12	Programação
A	24	16	11	3	3	6	324
B	24	16	11	3	3	6	324
C	24	16	11	3	3	6	324
D	24	16	11	3	3	6	324
E	24	16	11	3	3	6	324
F	24	16	11	3	3	6	324
G	24	16	11	3	3	6	324
H	24	16	11	3	3	6	324
I	24	16	11	3	3	6	324

Tabela 5.20 – Total Acumulado de Turnos por Tipo e Horas Acumuladas por Categoria

	Turnos Acumulados					Total Horas
Categorias	TM	TD	TN	TFS4	TFS12	Acumuladas
A	20	10	8	8	5	396
B	20	10	8	9	5	400
C	20	10	8	9	5	400
D	20	10	8	9	5	400
E	20	10	8	9	5	400
F	20	10	8	10	5	404
G	11	3	3	6	0	160
H	11	3	3	6	0	160
I	11	3	3	6	0	160

A figura 5.11 apresenta os resultados da otimização na busca da programação final para o plano de operações 3. A tabela 5.21 mostra os resultados para o experimento de melhor desempenho. Observa-se, como no caso anterior, a necessidade do incremento de uma categoria para encontrar uma solução ótima em um tempo considerado prudente para a complexidade do problema. A primeira solução foi encontrada para o experimento 2 a 22.7 minutos de processamento e 277 avaliações completas da programação para 8 categorias. Nos experimentos 1 e 3 foram consumidos 26.3 e 26.1 minutos respectivamente para chegarem a uma solução ótima.

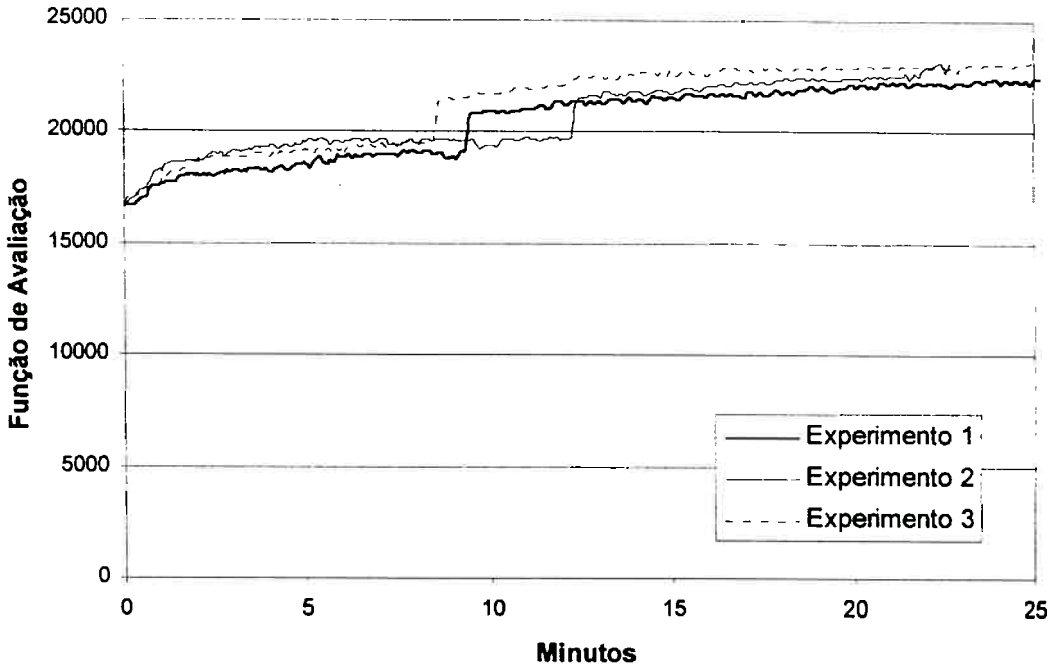


Figura 5.11 – Resultados da Otimização (Plano de Operações 3 – Tabela 5.13)

Tabela 5.21 - Resultados da Otimização do Melhor Experimento (vide figura 5.11)

	Tempo de Processamento (minutos)	Categorias	Experimentos	No. Avaliações à Programação (iterações)	Função de Avaliação	No. de Restrições Violadas				
						Finas Semanas	Diferença Horas	Seqüência Folgas	Seqüência Turnos	Horas Trabalho
	0,0	7	Programação Inicial	1	16763,9	2	10	5	6	10
	12,4	8	Programação Inicial	173	21291,5	0	7	5	1	3
Solução	22,7	8	Programação Final	277	23141,6	0	0	0	0	0

A tabela 5.22 mostra a programação final para esta solução, observando o cumprimento de todas as restrições. Devido ao fato de serem necessárias 8 categorias, foram alocadas as 8 correspondentes do nível de operários, começando com as categorias de menores horas trabalhadas até aquele instante. A tabela 5.23 mostra a distribuição final de turnos e horas na programação, e a tabela 5.24 apresenta o acumulado final de turnos e horas para a totalidade das categorias no final do exercício, com os três planos de operações consecutivos.

Tabela 5.22 – Programação Final (Plano de Operações 3 – Tabela 5.13)

	Semana 1							Semana 2							Semana 3							Semana 4						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Categorias	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.
B	0	2	3	0	2	0	5	1	1	1	0	2	0	0	0	1	3	0	1	5	0	2	0	1	1	2	4	5
C	0	0	2	3	2	0	5	1	1	0	2	1	0	0	0	0	2	3	0	5	0	2	1	0	0	0	4	0
D	1	0	1	2	0	0	5	2	0	1	2	1	4	0	1	0	0	2	3	0	5	2	1	0	0	1	0	5
E	1	2	0	2	1	0	0	3	2	0	1	0	5	0	1	2	0	2	1	4	5	3	0	0	1	0	0	5
F	2	1	0	1	3	4	0	2	0	3	2	2	5	0	2	1	0	1	1	0	5	0	2	3	2	1	0	0
G	2	1	0	1	1	4	0	2	2	0	0	1	0	5	2	1	1	0	0	0	0	0	0	2	3	2	5	0
H	2	1	1	0	0	5	0	0	0	0	1	3	4	5	2	2	0	1	2	0	0	1	1	1	2	3	0	0
I	3	0	0	2	1	5	0	0	1	2	3	0	0	5	3	0	1	2	2	4	0	1	2	0	2	1	5	0

	Semana 5							Semana 6							Semana 7							Semana 8						
	29	30	31	32	33	34	35	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
Categorias	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.	2a.f.	3a.f.	4a.f.	5a.f.	6a.f	Sáb.	Dom.
B	0	0	3	2	1	0	0	0	2	1	0	0	5	0	2	1	3	0	2	4	0	1	2	1	2	0	5	0
C	1	0	0	3	2	5	0	1	2	1	1	0	5	0	2	1	2	3	2	4	0	1	1	2	1	0	5	0
D	2	1	0	2	3	4	0	1	1	0	0	1	0	5	2	2	0	0	3	5	0	2	0	3	2	1	0	0
E	1	2	0	2	1	0	0	3	0	0	2	1	4	5	0	2	1	1	0	0	0	3	2	0	2	1	0	5
F	0	1	2	1	2	5	0	0	0	3	2	1	0	5	1	0	0	1	0	0	0	2	0	0	0	1	4	5
G	2	1	1	0	0	0	5	2	1	0	3	2	0	0	3	0	0	2	1	5	5	0	1	1	3	2	4	0
H	2	0	0	1	1	0	5	2	0	2	2	3	4	0	1	0	1	2	1	0	5	2	0	0	0	3	0	5
I	3	2	1	0	0	4	5	2	1	0	1	2	0	0	0	1	0	2	1	0	5	0	1	0	1	2	0	0

Notação:
0: Folga
1: TM Turno de Manhã
2: TD Turno de Tarde
3: TN Turno de Noite
4: TFS4 Turno de Fim-de-Semana
5: TFS12 Turno de Fim-de-Semana

Tabela 5.23 – Total de Turnos por Tipo e Horas por Categoria (Plano de Operações 3 – Tabela 5.13)

		Turnos Programação					Total Horas
Categorias	Folgas	TM	TD	TN	TFS4	TFS12	Programação
B	22	12	11	4	2	5	284
C	22	12	11	4	2	5	284
D	22	12	11	4	2	5	284
E	22	12	11	4	2	5	284
F	22	12	11	4	2	5	284
G	22	12	11	4	2	5	284
H	22	12	11	4	2	5	284
I	22	12	11	4	2	5	284

Tabela 5.24 – Total Acumulado de Turnos por Tipo e Horas Acumuladas por Categoria

	Turnos Acumulados					Total Horas
Categorias	TM	TD	TN	TFS4	TFS12	Acumuladas
A	20	10	8	8	5	396
B	32	21	12	11	10	684
C	32	21	12	11	10	684
D	32	21	12	11	10	684
E	32	21	12	11	10	684
F	32	21	12	12	10	688
G	23	14	7	8	5	444
H	23	14	7	8	5	444
I	23	14	7	8	5	444

5.4 Testes Experimentais do Algoritmo 3 com outros Operadores Nebulosos

Como foi definido na seção 3.5.7, a seleção dos parâmetros tem um efeito determinante na qualidade do modelo nebuloso a ser utilizado no sistema e não existe nenhum método formal que facilite a escolha da melhor combinação de parâmetros dos MN's para uma definida aplicação que não seja diferente da experimentação. Utilizar-se-á, então, dos procedimentos apresentados na mesma seção, a seleção empírica para estabelecer os parâmetros que representem o melhor desempenho para a aplicação em estudo, ou seja, do planejamento de uma programação de turnos.

Diferentes modelos nebulosos foram projetados na implementação do algoritmo 3. Para a fase experimental, este trabalho se restringe aos definidos nas tabelas 5.25 e 5.26 seguindo a estrutura definida para a descrição dos MN's da seção 3.5.8. Para o conectivo "e" utiliza-se o operador "mínimo".

Tabela 5.25 – Operadores da Agregação do MN Projetado (Agregação-Defuzzificação)

MN (*,*,k,0)	Definição	Nome
1	$\min(\mu_A(x), \mu_B(x))$	Operador T de Zadeh (ZADEH:1973)
2	$\max(\mu_A(x), \mu_B(x))$	Operador T de Zadeh (ZADEH:1973)
3	$O_{fuzzy\_and}(\mu_A(x), \mu_B(x)) = \gamma \cdot \min(\mu_A(x), \mu_B(x)) + \frac{(1-\gamma)(\mu_A(x) + \mu_B(x))}{2}$	*Fuzzy and* (ZIMMERMANN:1996)
4	$O_{fuzzy\_or}(\mu_A(x), \mu_B(x)) = \gamma \cdot \max(\mu_A(x), \mu_B(x)) + \frac{(1-\gamma)(\mu_A(x) + \mu_B(x))}{2}$	*Fuzzy or* (ZIMMERMANN:1996)

Tabela 5.26 – Método de Defuzzificação Utilizado

MN (*,*,*,l)	Definição	Nome
l	$\bar{y} = \frac{\sum_{i=1}^{l} y_i \mu_{S_{l'}}(y_i)}{\sum_{i=1}^{l} \mu_{S_{l'}}(y_i)}$	Centro de Gravidade

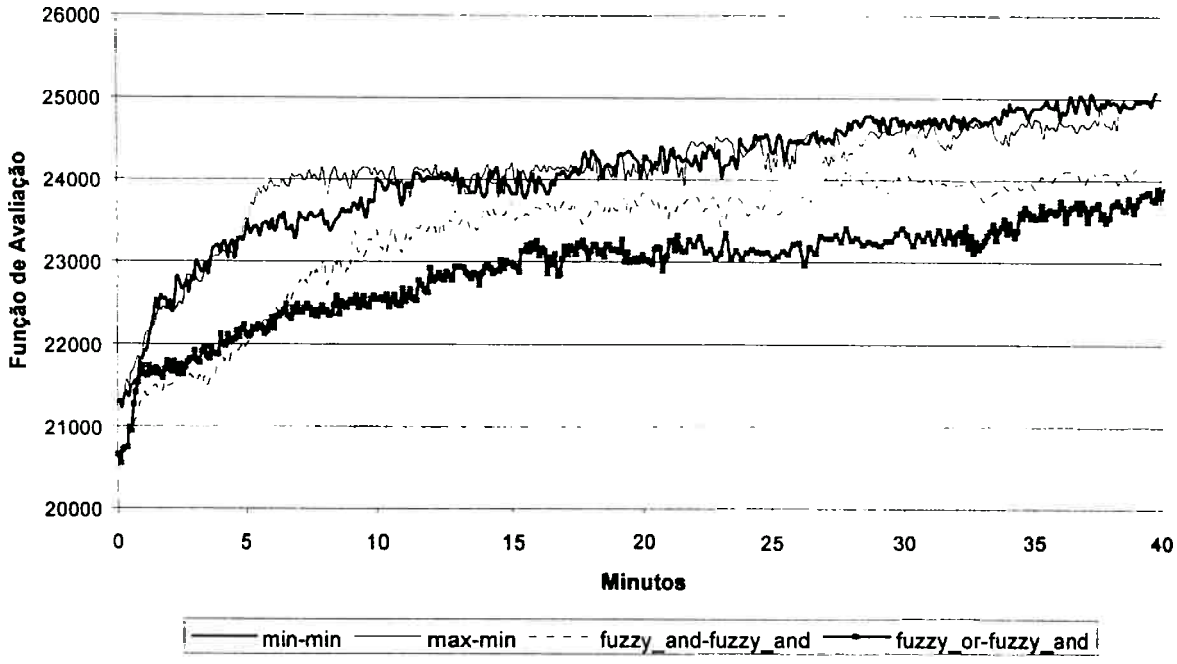


Figura 5.12 – Desempenho da Otimização com diferentes Operadores Nebulosos

A figura 5.12 mostra as curvas de desempenho da otimização comparando diferentes configurações de modelos nebulosos. Observa-se que indiferente dos operadores utilizados, haverá sempre uma melhoría da qualidade da solução em relação à programação inicialmente gerada.

Após vários experimentos, observou-se que, a curva de otimização que corresponde à composição "max-min" apresenta nos tempos iniciais um melhor desempenho, porém sem satisfazer todas as restrições, entretanto, foi o modelo nebuloso possuindo a composição "min-min" que apresentou o melhor desempenho em relação à satisfação das restrições *versus* tempo. E por isso, utilizou-se estes operadores para o projeto dos MN's na representação das restrições no algoritmo 3.

Capítulo 6

CONCLUSÕES E PROPOSTAS PARA FUTUROS TRABALHOS

6.1 Conclusões

O problema de programações é considerado de difícil tratamento na teoria e na prática. O problema, do ponto de vista teórico, referente à busca de uma programação ótima e sujeita a um número limitado de restrições, sofre de uma excessiva complexidade combinatória e são em sua maioria *NP hard*. Problemas práticos, como o de programações de turno, são complexos devido ao número e à variedade de suas restrições, a maioria delas sendo *soft* (restrições potencialmente relaxáveis à preferência humana), em vez de restrições fisicamente *hard*. Além disso, uma "boa" programação precisa ser avaliada freqüentemente entre um número de restrições conflitantes entre si e, que por sua vez, não podem ser definidas com precisão.

O uso de técnicas matemático-analíticas para resolver problemas práticos de programações tem sido considerado limitado devido à falta de uma linguagem expressiva para a representação dos objetivos e das restrições.

A principal área de aplicação da teoria dos conjuntos nebulosos a programações está na estrutura sistemática que esta oferece para a representação, aplicação, propagação e relaxamento de restrições imprecisas, e para a avaliação da programação com respeito a objetivos definidos vagamente. Portanto, as programações de turnos podem ser formuladas como um problema de otimização através de um problema de satisfação parcial de restrições (onde as restrições e os objetivos são tratados exatamente da mesma maneira), cuja tarefa é a busca por uma programação que possua o máximo grau de satisfação dos objetivos e das restrições, as quais podem estar sujeitas a imprecisões.

O ponto central desta tese é, portanto, a integração entre problemas clássicos de satisfação de restrições e a teoria dos conjuntos nebulosos, com o objetivo de aplicá-los ao tratamento de um problema de natureza combinatória, como são as programações de turnos no processo de estiva em um terminal de contêineres.

A metodologia aqui proposta tenta resolver as dificuldades próprias dos problemas combinatórios através de uma combinação entre a representação das restrições por meio da teoria dos conjuntos nebulosos e sua solução, utilizando métodos de melhoria iterativa com base em passos-reparos. Esta metodologia fundamentou um modelo de sistema de programação dinâmica de turnos, batizado de SIPRODT, que resolve o problema de programações de turnos no processo de estiva de um terminal de contêineres, proporcionando paralelamente a flexibilidade suficiente para atingir a

variabilidade freqüente e imprevisível da demanda de serviços assim como de acontecimentos imprevistos com o corpo de empregados.

Demonstrou-se como as restrições podem ser representadas por conjuntos nebulosos. Esta representação é motivada pelas imprecisões resultadas das restrições definidas vagamente e que são freqüentemente expressas pelos programadores humanos usando variáveis lingüísticas, definíveis operacionalmente, mapeando-se a faixa preferida dos valores dos parâmetros sobre os correspondentes das funções de pertinência dos conjuntos nebulosos, representando essa restrição. O mapeamento pode representar a preferência subjetiva de um programador humano, ou pode ser derivado de um sistema de gerenciamento em hierarquias refletindo as preferências atuais entre possíveis objetivos conflitantes na estratégia organizacional. Da mesma maneira, eficazes heurísticas, tais como as baseadas nas técnicas de melhoria iterativa, guiadas pelas avaliações das restrições nebulosas, permitem que esta metodologia, além de ter um comportamento bem definido e ser matematicamente válida, tenha um entendimento facilmente intuitivo.

As técnicas de melhoria iterativa são métodos de busca que começam a partir de uma solução inicial e tentam encontrar soluções melhores por modificações "locais". Desafortunadamente, problemas de programações possuem muitas soluções que diferem em sua qualidade e as que são consideradas como "boas" geralmente não se encontram próximas. Portanto, métodos de busca baseados em melhorias podem bloquear-se em ótimos locais. A característica importante destes métodos deve ser, portanto, a capacidade de escapar destes ótimos locais. Desta maneira, foi introduzida nesta tese uma heurística construtiva a um baixo custo computacional para gerar uma solução inicial (programação inicial – algoritmo 2); foram definidos e implementados operadores ou passos reparos para modificar a solução inicial, dependendo estes da aplicação em estudo; e, finalmente, foram introduzidos dois algoritmos de controle baseados nas técnicas de melhoria por reparos, um de natureza determinística, reparo em profundidade, apresentou um melhor desempenho na aplicação que o outro algoritmo, de caracter estocático, reparo por modificações aleatórias, como foi demonstrado no capítulo 5. Demonstrou-se também que através desta metodologia é totalmente possível satisfazer simultaneamente todos os requerimentos próprios de um problema de programações, uns dos maiores problemas no projeto de "boas" soluções produto da natureza conflitante entres os requisitos do sistema.

Uma importante questão no tratamento do problema de programação de turnos através de restrições nebulosas está na dificuldade de quantificar os benefícios desta abordagem. Entretanto, segundo a literatura especializada, toda heurística baseada em reparos tem alcançado soluções mais rápidamente que as abordagens construtivas usando os mesmos parâmetros; contudo, a cronometragem do resultado não é realmente comparável devido aos diferentes ambientes de programação e tempo de busca da solução. Da mesma maneira, se o desempenho da programação formulada através de restrições nebulosas e as não-nebulosas podem ser facilmente comparado em termos de tempo tomado para computar uma programação "aceitável", torna-se muito mais difícil comparar a programação produzida sem um experimento de campo ou sem simulações experimentais contendo parâmetros ou eventos cujas distribuições de probabilidade sejam desconhecidas para o programador. Até o momento, escassos estudos tentam enfocar esta questão.

O SIPRODT, uma vez estabelecido para um cenário específico, produzirá programações cada vez que seja necessário, levando sempre um registro histórico dos turnos e das horas acumuladas por empregados/categorias. O SIPRODT foi projetado de

modo a permitir um fácil manuseio a um amplo número de usuários, refinando e modificando programações simplesmente adicionando ou mudando o plano de operações representativo da demanda. Isto se reflete na melhoria do serviço de atendimento aos clientes, facilitando o cumprimento dos prazos, por exemplo, pode-se reduzir o tempo de permanência das embarcações nos cais durante o processo de estiva. O SIPRODT vem facilitar uma melhor distribuição do recurso humano dentro dos diferentes processos/tarefas do terminal, assim como tenta equilibrar a carga de trabalho entre os empregados/categorias, distribuindo uniformemente os diferentes tipos de turnos. Caso a demanda de serviço seja menor do que a prevista inicialmente, pode-se reduzir o tempo não produtivo para um grupo de empregados programados para um definido processo, permitindo seu deslocamento para outras atividades onde sejam necessários

Com SIPRODT, administradores do terminal poderão gerar diferentes programações, explorar cenários, processar simulações e efetuar análises do tipo "e se". As programações poderão ser facilmente atualizadas incorporando-se novos padrões de turnos, horizontes de programação e novas restrições.

A conteinerização tem mudado fundamentalmente a natureza de trabalho nos terminais e tem revolucionado as práticas laborais do porto, reduzindo apreciavelmente a necessidade de mão de obra. Se a experiência dos portos especializados nos países desenvolvidos se repete nos países em desenvolvimento, se produzirá uma redução considerável do número total de empregados nos terminais, na medida em que avance a conteinerização. Isto afetará todas as categorias – empregados administrativos (planejamento e supervisão) e empregados operacionais. A categoria de pessoal operativo é a que sofre as maiores reduções pelo fato de participar diretamente com a manipulação da carga. O SIPRODT auxiliará não só no acompanhamento contínuo da demanda do serviço no terminal, mas também no estabelecimento de uma maneira racional do nível de operários necessário para um definido processo em um específico terminal.

6.2 Propostas para Futuros Trabalhos

O SIPRODT até aqui calcula o nível de escalação necessário para uma definida demanda e projeta a programação, definindo o turno que deverá cobrir cada empregado/categoria. Como uma proposta de continuação a este trabalho, o sistema poderia designar além do turno, a função que este empregado poderia realizar. Para isso se deverá criar um banco adicional de dados que incluiria uma lista dos empregados classificados por qualificações, tal como se mostra na proposta do anexo B desta tese, e definir as requisições dos empregados em função das operações pertinentes do terminal. De modo a fechar o ciclo do que seria um sistema de gerenciamento de recursos humanos dentro do terminal, deverão ser também considerados os períodos de licenças e de férias a que tem direito cada empregado, para isto deverá se incrementar mais esta restrição e seus respectivos operadores de reparo.

O sistema apresentado considera somente empregados permanentes ligados diretamente com o terminal e cumprindo com um único turno por dia. Diferentes estratégias de organização do trabalho no terminal, tais como a utilização de empregados avulsos e o uso de horas extras, poderão ser futuramente considerados,

incluindo-se uma função de custo que auxilie a tomada de decisão na designação dos turnos.

Na atualidade, muitas abordagens no tratamento de problemas de natureza combinatória parecem ser eficazes ao tipo de problema a ser resolvido. Resultaria, então, um interessante desafio: pesquisar o papel de SIPRODT como um "solucionador" geral de uma maneira mais eficiente para este tipo de problemas.

Como se observou na aplicação de programações de turnos, dois tipos de restrições, *hard* e *soft*, são comuns em aplicações da vida prática. Em SIPRODT as restrições *hards* tentam ser satisfeitas na geração da solução inicial, utilizando um método construtivo; as restrições *soft* são tratadas com base na teoria dos conjuntos nebulosos. Contudo, o desempenho do processo de busca depende amplamente de ajustes experimentais destas últimas restrições, chamadas aqui também de restrições nebulosas. Seria de grande importância pesquisar técnicas que facilitem a determinação destes números nebulosos a fim de diminuir o tempo empregado com experimentos preliminares. Estes experimentos exigiram um tempo razoavelmente alto.

Existem múltiplas alternativas para os defuzzificadores, para definir os operadores de implicação nebulosa, para selecionar os operadores de conjunção e disjunção, e para definir os conjuntos das funções de pertinências a serem usados na definição das variáveis lingüísticas. O projetista tem que selecionar o operador a ser usado em cada fase e esta decisão tem sido tomada em função de operadores mais comumente usados. Contudo, seria importante estabelecer qual etapa possui uma maior influência no comportamento e desempenho do modelo nebuloso e assim estabelecer os operadores cujas alternativas deverão ser cuidadosamente estudadas quando se desenvolve um MN para uma aplicação real. Portanto, o projetista deverá prestar atenção na fase em que a seleção do operador é, estatisticamente, mais significativa. Desta forma será possível prevenir uma analise detalhada de diferentes configurações que conduzem a um sistema com desempenhos similares.

Um dos principais desafios na pesquisa de programações, fundamentadas na teoria dos conjuntos nebulosos, é encontrar uma maneira apropriada de realizar as operações de interseção entre as diferentes restrições nebulosas a fim de estabelecer o grau de satisfação das restrições. Os operadores T e operadores médios são ferramentas flexíveis que podem cumprir essa função e, de uma maneira geral, estes operadores podem ser usados para a modelagem de processos de tomada de decisões onde os operadores de ZADEH *min* e *max* são, geralmente, os mais aplicados. No trabalho, foi mostrada uma ampla gama destes operadores onde foram selecionados somente um exemplo de operador T e um de operador médio para experimentar a particular aplicação. Os resultados mostraram melhor desempenho para os operadores de ZADEH *versus* os operadores médios. Pesquisas futuras deverão ser encaminhadas empregando-se outros operadores além dos analisados no presente estudo.

Nos últimos anos, numerosos trabalhos têm sido publicados abordando algoritmos meta-heurísticos (algoritmos genéticos, *tabu search*, *simulate annealing*) e suas aplicações práticas. Algoritmos meta-heurísticos têm demonstrado ser extremamente confiáveis para a solução de problemas de natureza combinatória. Do mesmo modo que neste trabalho foram estudadas e implementadas as técnicas de reparo em profundidade e de reparo por modificações aleatórias, estas meta-heurísticas poderão também ser analisadas e implementadas futuramente a fim de se desenvolver estratégias de busca que explorem a característica de representação das restrições nebulosas.

Finalmente, observar-se que a metodologia desenvolvida neste trabalho pode ser facilmente adaptada para outras aplicações de programações de turnos ou outros tipos de programações com a definição de novos domínios, cuidando-se de rever as restrições atuais, identificando-se e formulando-se as novas e, por último, definirem-se e implementarem-se os passos de reparo que venham a depender do novo domínio em estudo.

REFERÊNCIAS

(ALFARES:1994) Alfares, H. K., "An efficient two-stage algorithm for manpower days-off scheduling", in *15th. International Symposium on Mathematical Programming*, Ann Arbor, Michigan, U.S. A., August, 1994

(BAKER:1974) Baker, K. R., "Scheduling a Fulltime Workforce to Meet Cyclic Staff Requirements", *Management Science*, Vol. 20, pp. 1561-1568, 1974

(BAKER:1979) Baker, K. R., Burns, R. N., and Carter, M. W., "Staff Scheduling with Day-off and Workstretch Constraints", *AIIE Transactions*, Vol. 6, No. 11, pp. 286-292, 1979

(BAKER:1977) Baker, K. R., and Magazine, M. J., "Workforce Scheduling with Cyclic Demands and Day off Constraints", *Management Science*, Vol. 24, pp. 161-167, 1977

(BANDLER:1980) Bandler, W. and Kohout L., "Fuzzy power sets and fuzzy implication operators", *Fuzzy Sets and Systems,* Vol. 4, pp. 13-30, 1980

(BARBAROSOGLU:1999) Barbarosoglu, G. and Ozgur, Demet, "A Tabu Search Algorithm for the Vehicle Routing Problem", *Computer & Operations Research,* Vol. 26, pp. 255-270, 1999

(BARTHOLDI:1980) Barthholdi, J.J., Orlin, J. B. and Ratliff , H. D., "Cycle Scheduling via Integer Programs with Circular Ones", *Operations Research*, Vol. 28, pp. 1074-1085, 1980

(BEAUMONT:1997) Beaumont, Nicholas., "Scheduling staff using mixed integer programming", *Eur. J. Operational Research,* Vol. 98-3, pp. 473-484, 1997

(BERRY:1993) Berry, Pauline M., "Uncertanty in Scheduling: Probability, Problem Reduction, Abstractions and the User", published in *IEE Computing and Control Division Colloquium on "ADVANCES SOFTWARE TECHNOLOGIES FOR SCHEDULING"* No. 163, April 1993

(BILLIONNET:1999) Billionnet, Alain., "Integer programming to schedule a hierarchical workforce with variable demands", *Eur. J. Operational Research*, Vol. 114, pp. 105-114, 1999

(BLOSSOM:1993) Blossom, Aaron Paul., A comparative study of manpower scheduling algorithms: A multi-objective heuristic procedure, tabu search and simulated anneling, PhD thesis, Michigan State University, 1993

(BONISSONE:1992) Bonissone, Piero P.., "Reasoning, plausible", In Stuart C. Shapiro, editor, *Encyclopia of Artificial Intelligence,* John Wiley & Sons, Inc., 2nd edition, Vol. 2, pp. 1307-1322, 1992

(BOYCE:1995) Boyce, J.F., Dimitropoulos, C. H. D., G. vom Scheidt, Taylor, J. G., "GENET and Tabu Search for Combinatorial Optimization Problems", World Congress on Neural Networks, Washighton D.C., 1995

(BROWNELL:1976) Brownell, W. D. and Lowerre J.M., "Scheduling of Work Forces Required in Continuous Operations under Alternative Labour Policies", *Management Science*, Vol. 22, pp. 597-605, 1976

(BURNS:1978) Burns, R. N. "Manpower Scheduling with Variable Demands and Alternate Weekends Off", *INFOR*, Vol. 16, pp. 101-111, 1978

(BURNS:1987) Burns, R. N. and Koop, G. J., "A Modular Approach to Optimal Multiple-Shift Manpower Scheduling", *Operations Research*, Vol. 35, pp. 100-110, 1987

(CAMPOS:1999A) de Campos, L. M. and Huete J. F., "Independence Concepts in Possibility Theory: Part I", *Fuzzy Sets And Systems*, Vol. 103-1, pp. 127-152, 1999

(Campos:1999B) de Campos L. M. and Huete, J. F., "Independence Concepts in Possibility Theory: Part II", *Fuzzy Sets And Systems*, Vol. 103-3, pp. 487-505, 1999

(CÓRDON:1997) Cordón, O., Herrera, F. and Peregrí, A., "Applicability of the fuzzy operators in the design of fuzzy logic controllers", *Fuzzy Sets and Systems,* Vol. 86, pp. 15-41, 1997

(COSTA:1995) Costa A., Glora, A., Faraboschi, P., Pagni, A. and Rizzotto, G., "Hardware solutions for fuzzy control", *Proc. of IEEE,* Vol. 83, pp. 422-434, 1995

(CHIP:1994) CHIPS V4, *Community News,* Vol. 7-1, Febreary 1994

(DECHER:1988) Decher, R., and Pearl, J., "Network-based Heuristic for Constraints Satisfaction Problems", *Artificial Intelligence,* Vol. 34, pp. 1-38, 1988

(DORN:1992) Dorn, J., Slany, W., and Stary, C. "Uncertainty Management by Relaxation of Conflicting Constraints in Production Scheduling", *Practical Approaches to cheduling and Planning Working Notes AAAI Spring Symposium Series*, Published by the American Association of Artificial Intelligence, pp. 62-66, March 1992.

(DORN:1994) Dorn, J., Girsch, Mario., Günter, Skele and Slany, W., "Comparison of Iterative Improvement Techniques for Schedule Optimization" in 13th *UK Planning, Scheduling Workshop*, Glasgow, P. Prosser, Sep. 1994.

(DORN:1995) Dorn J. and Slany, W., "A Flow Show with Compatibility in a Steelmaking Plant", *Intelligent Scheduling.* Eds. Monte Zweben and Mark S. Fox, Morgan Kaufmann, pp. 629-654, 1995

(DRIANKOV:1993) Driankov D., Hellendoorn H. and Reinfrank M., An Introduction to Fuzzy Control, Springer Verlag, 1993

(DUBOIS:1986) Dubois, D. and Prade, H., "Decision evaluation methods under uncertainty and imprecision", In Janusz Kacprzyk and Mario Fredizzi, editors, *Combining Fuzzy Imprecision with Probabilistic Uncertanty in Decision Making,* Lecture Notes in *Economical and Mathematical Sistems,* Springer Verlag, Vol. 310, pp. 48-65, 1986

(DUBOIS:1989) Dubois, D. and Prade, H., "Processing Fuzzy Temporal Knowledge", in *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 19, No. 4, pp. 729-744. July/August 1989

(DUBOIS:1994) Dubois, D., Fargier H. and Prade, H., "Propagation and Satisfaction of Flexible constraints", in R. R. Yager and L. A. Zadeh, editors, *Fuzzy Sets, Neural Networks and Soft Computing,* Van Nostrand Reinhold, New York, pp. 166-187, 1994

(DUBOIS:1998) Didier, D. and Prade, H., "Possibility Theory is not Fully Compositional" A comment on a hort note by H.J. Greenberg, *Fuzzy Sets and Systems*, Vol. 95-1, pp. 131-134, 1998

(EICHFELD:1996) Eichfeld, H., Künemund, T., and Kenke, M., "A 12 b geral-purpose fuzzy logic CHIP", *IEEE Trans. on Fuzzy Sistems,* Vol. 4, pp. 460-475, Nov. 1996.

(EMMONS:1991) Emmons, H. and Hung, R., "Off-Day Scheduling with Hierarchical Worker Categories", *Operations Research*, Vol. 39, pp. 484-495, 1991

(FARGIER:1993) Fargier, H., Lang, J. and Schiex, T., "Selecting preferred solutions in fuzzy constraint satisfaction problems", in Hans-Jürgen Zimmermann, editor, *EUFIT'93, First European Congress on Fuzzy and Intelligent Technology,* Aachen, Germany, pp. 1128-1134, September 1993

(FIGUEIREDO:1993) Figueiredo, M., Gomide, F., Rocha, A., and Yager, R. R., "Comparation of Yager's level set method for fuzzy logic control with Mamdani's and Larsen's methods", *IEEE Trans. on Fuzzy Systems,* Vol. 1, pp. 156-159, May 1993

(FREUDER:1982) Freuder, C, "A Sufficient Condition for Backtrack-free search", *Journal of the Association for Computing Machinery,* Vol. 29, pp. 24-32, 1982

(FREUDER:1990) Freuder, C., "Complexity of K-tree Structured Constraint Satisfaction Problems", in *Proceeding of the Ninth National Conference on Artificial Conference (AAAI)*, pp. 4-9, 1990

(FREUDER:1992) Freuder, C. and Wallace, Richard J., "Partial constraint satisfaction", *Artificial Intelligence,* Vol. 58, pp. 21-70, 1992

(GENDREAU:1998) Gendreau, M., Laporte, G. and Semet, F., "A Tabu Search Heuristic for the undirected selective Travelling Salesman Problem", *European Journal of Operational Research,* Vol. 106, pp. 539-545, 1998

(GUESGUEN:1994) Gesguen, Hans Werner., "A formal for weak constraint satisfation based on fuzzy sets", in *Proceedings of ANZIIS-94*, Brisbane, Australia, pp. 199-203, 1994

(GILES:1980) Giles, R., "Lukasiewicz logic and fuzzy set theory", *Internat. J. Man-Machine Stud.* Vol. 8, pp. 313-327, 1980

(GLOVER:1989) Glover, F., "Tabu Search part I", *ORSA J. Comput.*, Vol. 1, pp. 190-206, 1989

(GLOVER:1990) Glover, F., "Tabu Search part II", *ORSA J. Comput.*, Vol. 2, pp. 4-32, 1990

(GUPTA:1991A) Gupta M. and J. Qi, "Theory of T-norms and fuzzy inference methods", *Fuzzy Set and Systems*, Vol. 40, pp. 431-450, 1991

(GUPTA:1991B) Gupta M. and Qi, J., "Design of fuzzy logic controllers based on generalized T-operators", *Fuzzy Sets and Systems,* Vol. 40, pp. 473-489, 1991

(GUTIÉRREZ:1995) Gutiérrez, J. and Salvador, L. de., "Sobre las prestaciones de los processadores de información borrosa", *ESTYLF'95-Spain,* pp. 257-262, 1995

(HARALICK:1980) Haralick, R.M. and Elliot, G.L., "Increasing Tree Search Efficiency for Constraint Satisfation Problems", *Artificial Intelligence,* Vol. 14, pp. 263-314, 1980

(HELLENDOORN:1993) Hellendoorn H. and Thomas, C., "Defuzzification in fuzzy controllers", *J. Intell. Syst.,* Vol. 1, pp. 109-123, 1993

(HESHAM:1998) Hesham K. A., "An Efficient Two-Phase Algorithm for Cyclic Days-Off Scheduling" *Computers Ops Res.* Vol. 25-11, pp. 913-923, 1998

(HUNG:1991) Hung, R., "Single Shift Workforce Scheduling under Compressed Workweek" *OMEGA,* Vol. 19, pp. 494-497, 1991

(HUNG:1994) Hung, R., "Single-shift off-day scheduling of a hierarchical workforce with variable demands", *Eur. J. Operational Research*, Vol. 78-1, pp. 49-57, 1994

(ILOG:1996) ILOG. Ilog solver 3.2 reference manual, July 1996

(JANG:1995) Jang J. R. and Sun, C. T., "Nueuro-Fuzzy modeling and control", *Proceeding of the IEEE,* Vol. 83, pp. 378-406, 1995

(KERR:1994) Kerr, R. and Slany, W., "Research Issue and Challenges in Fuzzy Scheduling", Christian Doppler Laboratory for Expert Systems, *Draft version:* 93/10/24, Mar. 1994

(KLEMENT:1994) Klement, E. P., Mesiar, R. and Pap, E., "Associative Compensatory Operators", in *Proceeding of CIFT'94: CurrentIssues in Fuzyy Technologies: Decision Models and Systems,* Trento, Italy, June 1994

(KORF:1987) Korf, Richard E., "Depth-First Iterative-Deepening", *Artificial Intelligence,* Vol. 27-1, pp. 97-109, 1987

(KUMAR:1992) Kumar, Vipin., "Algorithms for Constrait-Satisfation Problems: A Survay", *AI Magazine,* Vol. 13-1, pp. 32-44, 1992

(LAU:1997) Lau, Hoong Chuin and Lua, Seet Chong., "Efficient Multi-Skill Crew Rostering via Constrained Set", *Information Technology Institute,* Singapora 1997

(LAZARO:1995) Lazaro Jose e Aristondo Pedro, "Using Solver for nurse scheduling", *in Proc. 1st. Int'l ILOG Solver Conf.,* 1995

(LIU:19980) Liu, Zhe., Algorithms for Constraints Satisfaction Problems – CSPs. PhD Thesis, University os Waterloo, Waterloo, Ontario, Canada, 1998

(LOWERRE:1977) Lowerre, J. M., "Workstretch Properties for the Scheduling of Continuos Operations under Alternative Labour Policies", *Management Science*, Vol. 23, pp. 963-971, 1977

(MACKWORTH:1992) Mackworth, A.K., "Constraint Satisfaction", In S.C. Shapiro, editor, *Encyclopedia of Artificial Intelligency,* Vol. 1, pp. 285-293, 1992

(MARTIN:1999) Martin, Spott., "A Theory of Possibility Distributions", *Fuzzy Sets and Systems*, Vol. 102-2, pp. 135-155, 1999

(MENDEL:1995) Mendel, J. M., "Fuzzy logic systems for engineering: A tutorial", *Procceding of IEEE,* Vol. 83, pp. 378-406, March 1995

(MEYER:1998) Meyer, Harald auf'm Hofe.› "Nurse Rostering as Constraint Satisfaction with Fuzzy Constraints and Inferred Control Strategies", German Research Center for Artificial Intelligence (DFKI), Postfach 2080, 67608 Kaiserslautern, Germany, 1998

(MILLER:1976) Miller, Pierskalla and Rath, "Nurse Scheduling using Mathematical Programming", *Operations Research*, Vol. 24, pp. 857-869, 1976

(MINTON:1992) Minton, S., Johnston, M. D., Philips, A. B., Laird, P., "Minimizing Conflicts: a Heuristic repair method for Constraint Satisfaction and Scheduling Problems", *Artificial Intelligence,* Vol. 58, pp. 166-205, 1992

(MONTANARI:1974) Montanari, U., "Networks of Constraints: Fundamental properties and aplications to picture processing", *Information Sciences,* Vol. 7, pp. 95-132, 1974

(MIZUMOTO:1989) Mizumoto, M., "Pictoral representations of fuzzy connectives, Part I: cases of t-norms, t-conorms and averaging operators", *Fuzzy Sets and Systems,* Vol. 31, pp. 217-242, 1989

(NADEL:1989) Nadel, B.A., "Constrait Satisfation Algorithms", *Computacional Intelligence,* Vol. 5, pp. 188-224, 1989

(NAKANISHI:1993) Nakanishi, H., Turksen, I. B. and Sugeno, M., "A review and comparation of six reasoning methods", *Fuzzy Sets and Systems,* Vol. 57, pp. 257-294, 1993

(NAU:1993) Nau, D.S., "State-Space Search, Problem Reduction, and Iterative Deepening: A Comparative Analysis", *Technical Research Report, Institue for System Research,* University of Maryland, 1993

(NEMHAUSER:1988) Nemhauser, G.L. and Wolsey, L.A., *Integer and Combinatorial Optimization,* Jhon Wiley and Sons, New York, 1988

(NONOBE:1998) Nonobe, Koji and Toshihibe, I., "A Tabu Search Approach to the Constraints Satisfaction Problems as a General Problem Solver", *European Journal of Operational Research,* Vol. 106, pp. 599-623, 1998

(PARK:1992) Park, D. and. Cao, Z., "Investigations on the applicability of fuzzy inference", *Fuzzy Sets and Systems,* Vol. 49, pp. 151-169, 1992

(PEARSON:1997) Pearson, J. and Jeavon, P., "A Survey of Tractable Constraints Satisfaction Problems", July 14, 1997

(PEDRYCZ:1996) Pedrycz, W and Olivira, J. V. de., "An algorithm framework for development and optimization of fuzzy models", *Fuzzy Sets and Systems,* Vol. 80, pp. 37-55, 1996

(PIRLOT:1996) Pirlot, Marc, "General Local Search Methods", *European Journal of Operation Research*, Vol. 92, pp. 493-511, 1996

(POOR:1972) Poor, R., 4 Days 40 Hours, London, Pergamon Press, 1972

(PRADE:1979) Prade, Henri., "Using Fuzzy Set Theory in a Scheduling Problem: A Case Study", *Fuzzy Sets and Systems,* Vol. 2-2, pp. 153-165, April 1979

(PREDYCZ:1993) Predycz, W., Fuzzy Control and Fuzzy Sistems, Jhon Wiley Sons Inc, Second ed., 1993

(REICHEL:1997) Reichel, Claus e Slany, W., editores., **FLIPS++ Version 1.0: A reusable iterative optimization library for combinatorial problems with fuzzy constraints: Reference Manual.* Number DBAI-TR-97-11. Institute of Information Systems (E184-2), Viena University of Technology, June 1997

(RODOSÊK:1997) Rodosêk, Robert., Generation and Comparation of Contraint-Based Heuristic using the Struture of Constraints, PhD Thesis, University of London, july 1997

(RONDEAU:1997) Rondeau, L., R. Ruelas, L. Levrat and Lamotte, M., "A defuzzification method respecting the fuzzification", *Fuzzy Sets and Systems,* Vol. 86, pp. 311-320, 1997

(ROTHSTEIN:1972) Rothstein, M., "Scheduling Manpower with Mathematical Programming", *Industrial Engineering*, Vol. 4, 1972

(RUNKLER:1993) Runkler, T. A. and Glesner, M., "A set of axioms for defuzzification strategies towards a theory of rational defuzzification operators", in *IEEE Intern. conf. on Fuzzy Systems,* pp. 1161-1166, 1993

(SAADE:1996) Saade, J., "A unifying approach to defuzzification and comparing of the outputs of fuzzy controller", *IEEE Trans. on Fuzzy Systems,* Vol. 4, pp. 227-237, August 1996

(SANTA:1996) Santa, U., Slany, W. and Younes, M., "*FLIP++: a partial constraints satisfaction system for steelmaking scheduling", *Proceeding of the Second International Conference on the Practical Aplications of Constraint Technology (PACT'96): Scheduling, Resource Allocation, Planning, Configuration,* pp 211-220, London, UK, April 1996

(SLANY:1996) Slany, W., "Scheduling as a fuzzy multiple criteria optimization problem", *Fuzzy Sets and Systems,* Vol. 78, pp. 197-22, 1996

(SLIWINNSKA:1985) Sliwinnska, D., Kiszka, J. B., and Kochanska, M. E., "The influence of some fuzzy implications operators on the accuracy of a fuzzy model – part I and part II", *Fuzzy Sets and Systems,* Vol. 15, pp. 111-128 / 223-240, 1985

(SMITH:1977) Smith, L.D., and Wiggins, A., "A Computer Base Nurse Scheduling of Two Consecutive Idle Periods", *Management Science*, Vol. 19, pp. 71-75, 1977

(SUGENO:1988) Sugeno M. and Kang, G. T., "Struture identification of fuzzy model", *Fuzzy and Sets Systems,* Vol. 28, pp. 15-33, 1988

(TIBREWALA:1972) Tibrewala, R., Phillipe, D. and Browne, J., "Optimal Scheduling of Two Consecutive Idle Periods", *Management Science*, Vol. 19, pp. 71-75, 1972

(TSANG:1993) Tsang, Edward., Foundations of Constraints Satisfation, Academic Press, 1993.

(TSUKAMOTO:1997) Tsukamoto, Y., "An Approach to Fuzzy Reasoning Method", pp. 137-149. Eds. Amsterdam: North-Holland, 1979

(TURKSEN:1992) Turksen, I. B., "Interval-value fuzzy sets and 'compenatory AND'", *Fuzzy Sets and Systems,* Vol. 51, pp. 295-307, 1992

(WANG:1994) Wang, L. X., Adaptive Fuzzy Systems and Control, PTR Prentice Hall, 1994

(WARNER:1976) Warner, D. M., "Scheduling Nursing Personnel According to Nursing Preference: A Mathematical Programming Approach", *Operations Research,* Vol. 24, pp. 842-856, 1976

(WEBER:1983) Weber, S., "A general concept of fuzzy connectives, negations and implications based on t-norms and t-conorm", *Fuzzy Sets and Systems,* Vol. 11, pp. 115-134, 1983

(YAGER:1983) Yager, Ronald R., "Some Relationships between Possibility, Truth and Certainty", *Fuzzy Sets and Systems*, Vol. 11-2 , pp. 151-156, 1983

(YAGER:1988) Yager, Ronald R., "On ordered weighted averaging aggregation operatiors in multicriterio decision making", *IEEE Transactions on Systems, Man, Cybernetics,* Vol. 18, pp. 183-190, 1988

(YAGER:1993) Yager, Ronald and Filev, D., "On the issue of defuzzification and selection based on a fuzzy set", *Fuzzy Sets and Systems,* Vol. 55, pp. 255-271, 1993

(ZADEH:1965) Zadeh, L. A., "Fuzzy Sets", *Informations and Control*, New York, Academic Press, Vol. 8, pp. 338-353, 1965

(ZADEH:1973) Zadeh, L. A., "Outline of a new approach to the analysis of complex systems and decision processes", *IEEE Trans. Systems Man Cybernet,* Vol. 3, pp. 28-44, 1973

(ZADEH:1975) Zadeh, L. A, "The concept of a Linguistic Variable and its applications to approximate reasoning", *Inf. Science,* Vol. 9, pp. 43-80, 1975

(ZADEH:1987) Zadeh, L. A, "Fuzzy Sets as a Basis for a Theory of Possibility", *Fuzzy Sets and Applications: Selected Papers by L.A. Zadeh*, John Wiley & Sons, Inc., pp. 193-218, 1987

(ZADEH:1994) Zadeh, L. A., "Soft Computing and Fuzzy Logic", *IEEE Software*, Vol 11, pp. 48-56, 1994

(ZIMMERMANN:1983) Zimmermann, Hans-Jürge, and Zysno, P., "Decisions and Evaluations by hierarchical aggregatio of information", *Fuzzy Sets and Systems,* Vol. 10, pp. 243-266, 1983

(ZIMMERMANN:1996) Zimmermann, Hans-Jürge. *Fuzzy Set Theory – and Its Applications,* 3nd. edition, 1996

Anexo A

TEORIA DOS CONJUNTOS NEBULOSOS - Uma Introdução

Em 1965, Lofti A. Zadeh publica o primeiro trabalho sobre conjuntos nebulosos, intitulado *Fuzzy Sets* (ZADEH:1965), no qual conceitua o tema:

"[...] *fornece uma forma natural de tratar problemas em que a fonte de imprecisão é a ausência de critérios claramente definidos de pertinência de classes em vez da presença de variáveis aleatórias*".

A1 Introdução à Teoria dos Conjuntos Nebulosos

A noção de conjunto nebuloso surge, essencialmente, pela necessidade da modelagem de expressões lingüísticas caracterizadas pela sua natureza vaga, imprecisa e incerta. Veja-se isto num exemplo: quando se projeta uma programação de turnos, é claramente possível que, sobre um ciclo de turnos programados, existam certos desvios permissíveis das horas médias trabalhadas, a fim de serem considerados, expressam-se por meio de variáveis lingüísticas, surgindo, então, expressões que são essencialmente imprecisas e vagas. Elas podem produzir abstrações variadas em diferentes pessoas. No entanto, o raciocínio humano, a forma de pensar, baseia-se principalmente na "computação de expressões e termos lingüísticos" (ZADEH:1994). O conceito de conjunto nebuloso estabelece, portanto, um ambiente adequado para representar os termos lingüísticos do ponto de vista formal. A pertinência ou não de um elemento em um conjunto não é indubitavelmente determinada como no caso dos conjuntos clássicos. Nos conjuntos nebulosos, por terem suas fronteiras imprecisas e vagas, a pertinência de seus elementos é gradual.

Definição A1: Conjunto Nebuloso

(ZADEH:1965) Seja X um conjunto universo constituído pela reunião de elementos denotados por x. Um conjunto nebuloso A em X define-se como um conjunto de pares ordenados como:

$$A=\{(x,\mu_A(x))/x\in X\}, \tag{A1}$$

onde $\mu_A(x)$ é uma extensão da função característica, chamada *função de pertinência* (FP), que estabelece um mapeamento entre os elementos x e os números reais pertencentes ao intervalo contínuo entre [0,1]. O grau de pertinência designa o nível de certeza com que cada elemento pertence ao conjunto.

Os conjuntos nebulosos típicos são ilustrados desenhando sua FP, tomando como base o seu conjunto universo. Na figura A1, ilustram-se as funções de pertinência de conjuntos nebulosos mais comumente usados: *S, Triangular, Trapezoidal, Gaussiano, Linear por trechos e Z.*

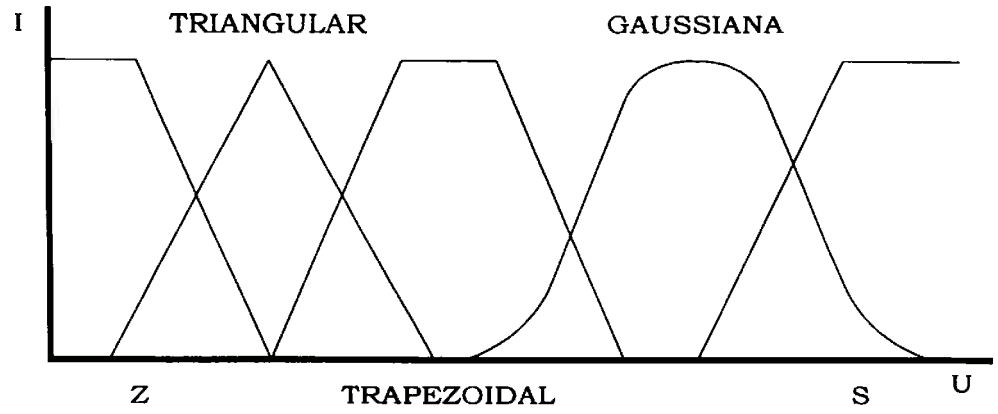


Figura A1 - Ilustração de diferentes tipos de Funções de Pertinência de Conjuntos Nebulosos

Exemplo A1:

Tem-se o conjunto *A* de "temperaturas quentes"; as funções de pertinência do conjunto nebuloso de *A* podem ser definidas como:

$\mu_A(80°C) = 1.0$ $\quad$ $\mu_A(40°C) = 0.4$
$\mu_A(70°C) = 0.9$ $\quad$ $\mu_A(20°C) = 0.2$
$\mu_A(65°C) = 0.8$ $\quad$ $\mu_A(05°C) = 0.01$

A2 Operações entre Conjuntos Nebulosos

Tal como na teoria clássica de conjuntos, operações entre conjuntos nebulosos também são necessárias para a aplicação de operações matematicamente bem definidas. Em seu artigo, L. A. Zadeh (ZADEH:1965) apresentou pela primeira vez uma série de operações teóricas para os conjuntos nebulosos, que basicamente são: *igualdade, inclusão, complemento, união, interseção, relação e convexidade*; todas estas operações são extensões da teoria clássica de conjuntos.

A2.1 Igualdade

Um conjunto nebuloso A é igual a um conjunto nebuloso B se:

$$A = B \Leftrightarrow \mu_A(x) = \mu_B(x), \ \forall x \tag{A2}$$

A2.2 Inclusão

Um conjunto nebuloso A é incluído em um conjunto nebuloso B se:

$$A \subseteq B \Leftrightarrow \mu_A(x) \leq \mu_B(x), \ \forall x \tag{A3}$$

A2.3 Complemento (vide Figura A2)

Um conjunto nebuloso A é complemento de um conjunto nebuloso B se:

$$\mu_A(x) = 1 - \mu_A(x), \ \forall x \tag{A4}$$

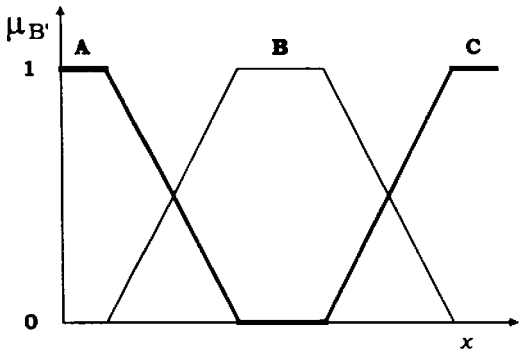


Figura A2 – Complemento μ_B.

A2.4 União (vide Figura A3)

A união de dois conjuntos nebulosos A e B é o menor conjunto nebuloso C contendo A e B, cuja função de pertinência é definida por:

$$\mu_C(x) = max[\mu_A(x), \mu_B(x)], \ \forall x \tag{A5}$$

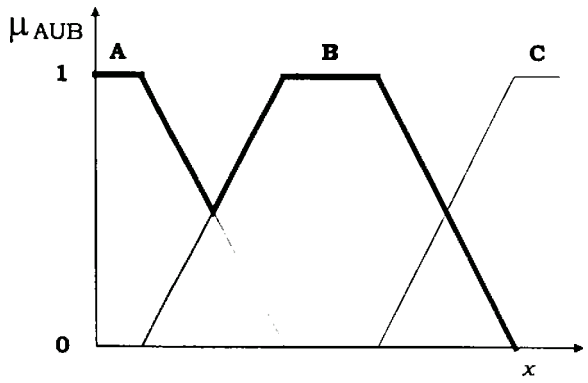


Figura A3 – União $\mu_{A \cup B}$

A2.5 **Interseção** (vide Figura A4)

A interseção de dois conjuntos nebulosos A e B é o maior conjunto nebuloso C que está contido em A e B, cuja função de pertinência é definida por:

$$\mu_C(x) = min[\mu_A(x), \mu_B(x)], \ \forall x \qquad \text{(A6)}$$

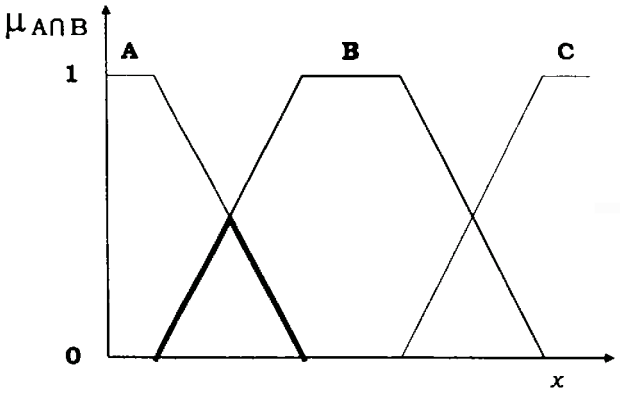


Figura A4 – Interseção $\mu_{A \cap B}$

A2.6 Relação

A relação de um conjunto nebuloso n-ésimo em X pode ser definida como um conjunto nebuloso A no espaço produto $X_1 \times ... \times X_n$. A função de pertinência para tal relação é da forma:

$$\mu_A(x_1,...x_n) \qquad x_i \in X_i \qquad i = 1,...,n \qquad \text{(A7)}$$

Exemplo A2: (ZADEH:1965)

A relação binária $x>>y$, $x, y \in R$ (R denota o conjunto dos números reais), que significa "*x é muito maior que y*", pode ser estimada como um conjunto nebuloso em $R \times R$. Uma função de pertinência apropriada poderia ter os seguintes valores:

$\mu_A(10,5) = 0$
$\mu_A(100,10) = 0.7$
$\mu_A(100,1) = 0.9$
etc.

A composição de duas relações nebulosas A e B, denotada por $A \otimes B$, é definido como a relação nebulosa cuja função de pertinência está relacionada a μ_A e μ_B por:

$$\mu_{A \otimes B}(x, y) = Sup\ Min[\mu_A(x, z), \mu_A(z, y)] \qquad \text{(A8)}$$

onde *Sup Min* denota o maior de todos os mínimos de $\mu_A(x, z)$ e $\mu_A(z, y)$.

A2.7 Convexidade

Um conjunto nebuloso é convexo (por exemplo, suas funções de pertinência somente tem um pico) se e somente se:

$$\mu_A(\lambda x_1 + (1-\lambda)x_2) \geq min(\mu_A(x_1), \mu_A(x_2)), \qquad x_1, x_2 \in A, \lambda \in [0,1] \qquad \text{(A9)}$$

Uma propriedade básica dos conjuntos nebulosos convexos: a interseção de dois conjuntos nebulosos convexos é também convexo.

A3 Representação de Restrições através da Teoria Possibilística

Devido ao tratamento de problemas de satisfação por restrições nebulosas e ao manuseio de variáveis cujos valores são de natureza incerta, Zadeh realizou pesquisas sobre as relações entre as teorias dos conjuntos nebulosos e as teorias probabilísticas e possibilísticas. Em seu artigo *Fuzzy Sets as a Basis for a Theory of Possibility*, Zadeh declara:

"[...] *uma restrição nebulosa pode ser interpretada como uma distribuição possibilística, com sua função de pertinência realizando o papel de distribuição possibilística, onde é associada uma variável nebulosa a uma distribuição possibilística da mesma maneira como uma variável aleatória é associada a uma distribuição probabilística*"

Pode-se imaginar, por exemplo, um problema de programação de turnos em um terminal marítimo em que alguns de seus parâmetros (como a duração das tarefas) podem ser mal conhecidas ou conhecidas vagamente. Incertezas destes processos podem ser representadas por teoria possibilística. A especificação de uma distribuição possibilística mede o grau na qual a distribuição permite um e somente um elemento como sua manifestação. Como tal, esta é uma medida da quantidade de incerteza ou de informação. Pesquisas detalhadas sobre teoria possibilística tem sido desenvolvidas por Ronald R. Yager R. (YAGER: 1983), L. A Zadeh (ZADEH:1987) e mais recentemente por Dubois D. et al. (DUBOI:1998), Martin S. (MARTIN: 1999), de Campos L. et al. (CAMPOS: 1999a) e (CAMPOS:1999b), entre outros.

Assumindo V como uma variável e seu valor no conjunto X, a proposição da teoria do raciocínio aproximado é uma expressão da forma:

$$V \text{ é } A \tag{A10}$$

onde A é um subconjunto nebuloso de X. Segundo Zadeh (ZADEH:1987), tal proposição induz a uma distribuição possibilística Π definida no espaço X. Em particular, para cada $x \in X$, $\Pi(x)=\mu(x)$, onde $\mu(x)$ é o grau de pertinência de x em A. Neste ambiente, $\Pi(x)$ indica o grau no qual x é um valor possível de V. Um conceito de uma distribuição possibilística está no espírito de uma distribuição probabilística. Os dois tipos de distribuições fornecem informação acerca de uma variável cujo valor atual é incerto. Yager R. (YAGER: 1983) introduz a especificação de Π para medir a quantidade de informação contida em uma distribuição possibilística, análogo à entropia *Shannon* em uma distribuição probabilística.

Para ilustrar a relação entre os conceitos de uma restrição nebulosa e de uma distribuição possibilística apresentar-se-á um breve exemplo tomado de Zadeh (ZADEH: 1987).

Exemplo A3:

Considere uma idade numérica, a dizer u=28, cujo grau de pertinência no conjunto nebuloso *jovem* é aproximadamente 0.7 . Primeiro, interpreta-se 0.7 como o grau de compatibilidade de 28 com o conceito rotulado *jovem*. Então, postula-se que a proposição *"João é jovem"* transforma o significado de 0.7 do grau de compatibilidade de 28 com *jovem* ao grau de possibilidade que João é 28 dada a proposição *"João é jovem"*. Resumindo, a compatibilidade de u com jovem se transforma na possibilidade desse valor de u dado *"João é jovem"*.

Neste trabalho, somente discutir-se-á a motivação para a introdução das teorias probabilísticas e possibilísticas na teoria dos conjuntos nebulosos. Para uma descrição mais detalhada, vide referências anteriormente indicadas. Uma importante questão destes estudos é a existência de diferenças fundamentais entre a possibilidade e a probalidade de um evento. Faz-se $\pi_x(u)$ ser a distribuição possibilística e $p_x(u)$ a distribuição probabilística de uma expressão u, então pode-se interpretar $\pi_x(u)$ facilmente que a realização seja u, enquanto $p_x(u)$ é a probabilidade determinada a partir de uma observação.

A4 Variáveis Lingüísticas

Como já foi mencionado em A1, nas aplicações da vida real, surge a necessidade de manusear variáveis imprecisas; estas vêm sendo representadas o mais próximo possível de sua natureza, posto que as informações numéricas nem sempre são disponíveis. Para permitir o manuseio destas variáveis, Zadeh (ZADEH:1975) introduziu o conceito de *variáveis lingüísticas*. Basicamente, denota-se com este nome uma variável cujos valores podem ser palavras ou frases em vez de apenas dados numéricos.

Definição A2: Variável Lingüística (ZADEH:1975)

Uma variável lingüística é caracterizada por uma quíntupla (x, $T(x)$, U, G, RS), na qual x é o nome da variável, $T(x)$ denota o conjunto de termos de x, isto é o conjunto de nome dos valores lingüísticos de x, sendo cada valor uma etiqueta lingüística denotada genericamente por X com intervalo sobre a extensão de U, ou seja, associada com a variável base u; U é conhecido como universo de discurso, G é a regra sintática para gerar os nomes X de valores x; RS é uma regra semântica para associar a cada X seu significado $RS(X)$, o qual é um subconjunto nebuloso em U. Cada X é chamado de um termo.

Para ilustrar este simbolismo um tanto nebuloso, veja-se um exemplo:

Exemplo A4:

Seja X a variável lingüística de nome *Idade* com seus elementos sendo seus termos lingüísticos {*muito jovem, jovem, velho, muito velho*}. Devido à expectativa de vida humana o universo de discurso é U=[0,100]. A variável base u é a idade em anos de vida. $RS(X)$ é uma regra que assinala um significado, um conjunto nebuloso, aos termos. Por exemplo, no caso do termo *muito jovem* poderia ser:

$$RS(muito\ jovem) = \{(u, \mu_{muito\ jovem}(u))/u \in [0,1]\} \qquad (A11)$$

com:

$$\mu_{muito\ jovem}(u) = \begin{cases} \max(0,\ 1 - m_1(15-u)) & se \quad u > 15 \\ 1 & se \quad u \leq 15 \end{cases} \qquad (A12)$$

$T(X)$ define o conjuntos de termos. Na figura A5, mostra-se uma possível definição dos termos lingüísticos e suas funções de pertinência. Onde m_1 é a inclinação assinalada da mesma figura.

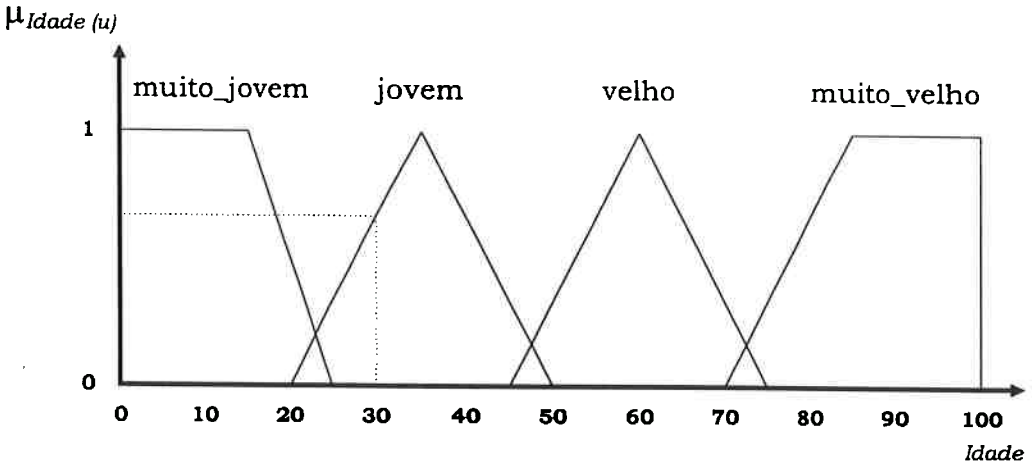


Figura A5 – Funções de Pertinência dos Termos da Variável Lingüística *Idade*

Definição A3: Regras "Se-Então"

Uma regra *"Se-Então"* é uma proposição que assume a seguinte forma:

$$Se\ u\ é\ A\ então\ w\ é\ B \tag{A13}$$

onde A e B são valores lingüísticos definidos por conjuntos nebulosos nos conjuntos universos U e W respectivamente. A expressão *"u é A"* é uma cláusula nebulosa que se chama antecedente ou premissa, e a expressão *"w é B"*, que também é uma cláusula nebulosa, chama-se conseqüente ou conclusão.

Existem duas formas de interpretar uma regra "Se-Então" (JANG:1995). Pode-se interpretá-la como "*A está acoplado com B*", ou seja, que existe uma relação nebulosa entre A e B. Esta relação é definida por um conjunto nebuloso pertencente ao produto cartesiano $U \times W$:

$$R = A \rightarrow B = \int_{UxW} \mu_A(u) \tilde{*} \mu_B(u)/(u,w) \tag{A14}$$

onde $\tilde{*}$ é um operador (t-norma)[6] entre os conjuntos nebulosos A e B. A outra forma é interpretar a regra como "A estabelece B", neste caso $A \rightarrow B$, pode assumir uma forma derivada da lógica clássica, como por exemplo:

$$R = A \rightarrow B = \neg A \cup B \tag{A15}$$

[6] Note-se que o símbolo $\int_{UxW}$ não significa integração como na matemática clássica. Aqui é usado para indicar uma reunião de elementos (u,w) com sua respectiva função de pertinência $\mu_A(u) \tilde{*} \mu_b(w)$.

A5 Inferência Aproximada

A inferência aproximada tem sido um princípio fundamental para o desenvolvimento das aplicações dos modelos nebulosos, pois a partir deles é possível deduzir os resultados da avaliação de um conjunto de regras "Se-Então" em condições determinadas. Este principio baseia-se na Regra Composicional de Inferência (RCI) (ZADEH:1973) e no *Modus Ponens* Generalizado (MPG). A regra composicional de inferência sustenta-se no seguinte princípio intuitivo: dados dois conjuntos nebulosos A e B definidos nos conjuntos universos U e W respectivamente, se existir uma relação nebulosa R entre os conjuntos A e B definidas no espaço $U \times W$, então é possível calcular o conjunto B mediante a composição do conjunto A com a relação entre A e B, isto é:

$$B = A \odot R \tag{A16}$$

onde $\odot$ denota um operador de composição. Por outro lado, o *Modus Ponens* Generalizado permite estabelecer o procedimento de inferência mediante o qual é possível deduzir a verdade de uma proposição com base na verdade de A' e de $A \rightarrow B$, isto é :

Premissa 1:	u é A'
Premissa 2:	Se u é A Então w é B
Conseqüente:	Então w é B'

O princípio anterior estabelece que, na medida em que A' seja similar a A, então B' será similar a B. Observe-se que o *Modus Ponens* é um caso especial do MPG. Baseados no MPG e na RCI pode-se agora definir a inferência aproximada.

Definição A4: Inferência aproximada

Sejam A e A' conjuntos nebulosos definidos em W, e B um conjunto nebuloso definido em V. Seja a implicação $A \rightarrow B$ definida como uma relação nebulosa estabelecida em $U \times W$, então o conjunto nebuloso B' induzido de "u é A' " e a regra *"Se u é A então w é B"* é definida por:

$$B' = A \odot R = A \odot (A \rightarrow B) \tag{A17}$$

A6 Modelos Nebulosos (MN's)

Os modelos ou sistemas nebulosos estabelecem um ambiente computacional que surgiu a partir de conceitos introduzidos pela teoria dos conjuntos nebulosos (as regras "Se-Então") e a inferência aproximada (JANG:1995). Um MN basicamente estabelece um mapeamento entre um espaço de entrada e de saída entre conjuntos nebulosos. Está constituído fundamentalmente por uma base de conhecimento e um mecanismo de

inferência. A base de conhecimento está constituída pelo conjunto de regras "Se-Então" e as definições das funções de pertinência dos termos lingüísticos usados nas regras.

O mecanismo de inferência define o algoritmo pelo qual é possível calcular a resposta do modelo como resultado da introdução de um valor específico nas entradas do modelo (sendo estes conjuntos nebulosos definidos nas entradas).

Um MN com a finalidade exposta denomina-se um modelo nebuloso puro (JANG:1995). Na figura A6 é ilustrada a configuração de um MN puro.

Por outro lado, para usar os MN em aplicações práticas, onde as magnitudes não são conjuntos nebulosos, devem ser introduzidas duas interfaces chamadas *fuzzificador* e *defuzzificador*. A primeira permite a conversão de números reais a conjuntos nebulosos e a segunda a conversão de conjuntos nebulosos a números reais. Chamar-se-ão estes MN de *fuzzificador* e *defuzzificador* (MNFD). Assim um MNFD estabelece um mapeamento entre um espaço de números reais a outro de números reais. Na figura A7 mostra-se a configuração deste modelo. Nas seções seguintes discutir-se-á cada uma das partes de um MNFD em detalhe.

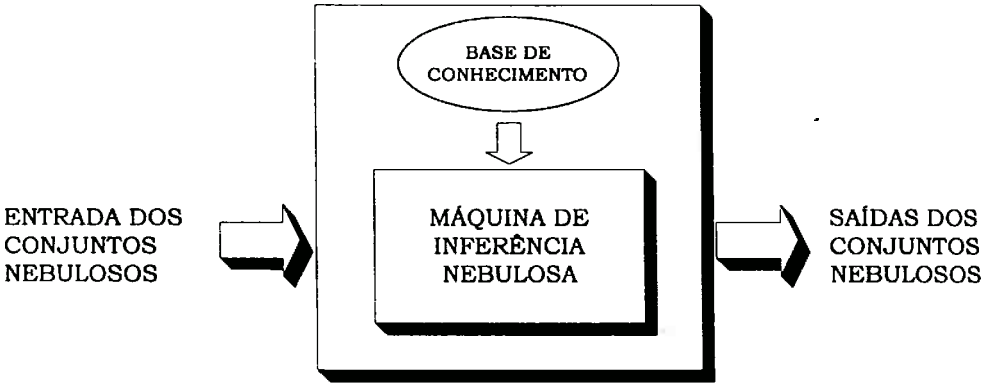


Figura A6 – Modelo Nebuloso Puro (MNP)

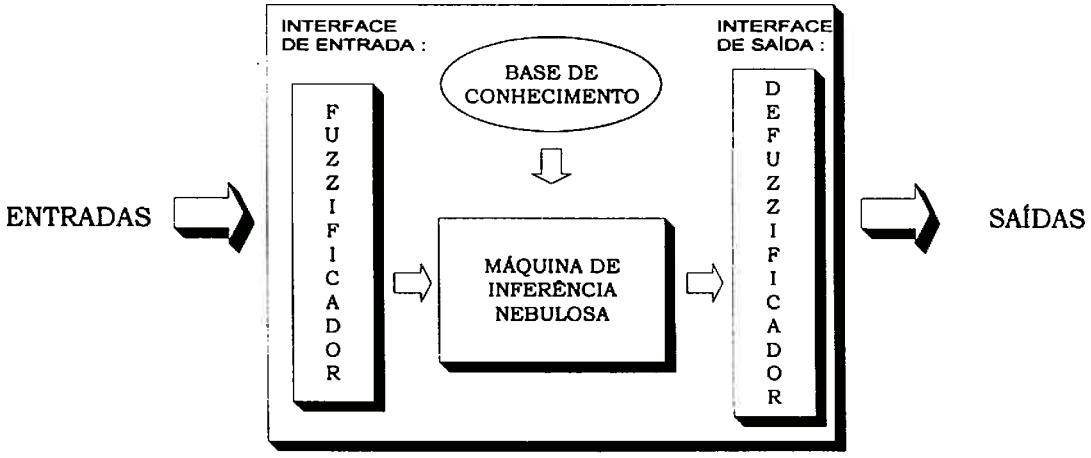


Figura A7 – Modelo Nebuloso com *Fuzzificador* e *Defuzzificador* (MNFD)

A6.1 Conjunto de Regras

Um conjunto de regras define, de acordo com o discutido na seção A4, uma relação entre um espaço de entrada e um espaço de saída. Nas seções seguintes, um MN é abstraído como um sistema com k entradas $x_1, x_2, \ldots, x_k$ e m saídas $y_1, y_2, \ldots, y_m$.

Definição A5: Conjunto de Regras Tipo Mandani

Um conjunto de regras nebulosas tipo *Mandani* com múltiplos antecedentes e múltiplos conseqüentes consiste numa reunião de regras "Se-Então" com a seguinte estrutura:

$$\begin{aligned} R^{(1)} &: Se\ x_1 \text{ é } E_1^1 \text{ e } x_2 \text{ é } E_1^2 \text{ e} \ldots x_k \text{ é } E_1^k \ Então\ y_1 \text{ é } S_1^1 \text{ e } y_2 \text{ é } S_1^2 \ldots y_m \text{ é } S_1^m \\ &\vdots \\ R^{(i)} &: Se\ x_1 \text{ é } E_i^1 \text{ e } x_2 \text{ é } E_i^2 \text{ e} \ldots x_k \text{ é } E_i^k \ Então\ y_1 \text{ é } S_i^1 \text{ e } y_2 \text{ é } S_i^2 \ldots y_m \text{ é } S_i^m \\ &\vdots \\ R^{(n)} &: Se\ x_1 \text{ é } E_n^1 \text{ e } x_2 \text{ é } E_n^2 \text{ e} \ldots x_k \text{ é } E_n^k \ Então\ y_1 \text{ é } S_n^1 \text{ e } y_2 \text{ é } S_n^2 \ldots y_m \text{ é } S_n^m \end{aligned} \quad \text{(A18)}$$

Observe-se que no conjunto de regras representadas em A18, é usado um conectivo "e" para ligar as expressões "x_j é E_i^j" e "y_j é S_i^j" nos antecedentes e conseqüentes respectivamente, onde E_i^j é um termo definido na entrada j e usado na regra i. Alguns outros conectivos poderiam ser usados para ligar os termos nos antecedentes, tais como: *or, xor, not,* etc., o que poderia diminuir o número de regras numa dada aplicação (EICHFELD:1996). Embora não seja explicitamente mostrado, existe um conectivo "também" *(also)* entre cada uma das regras. Este conectivo estabelece uma operação de agregação entre as regras. Um outro aspecto importante, com relação ao conjunto de regras com múltiplas entradas (k) e múltiplas saídas (m), é este ser conversível em um conjunto equivalente de $k.m$ regras de uma saída. Por outro lado, poderia ter-se um sistema com múltiplas entradas e múltiplas saídas, mas com base de conhecimento diferente para cada saída. Para efeito desta revisão, considerar-se-ão apenas sistemas com iguais antecedentes para todas as saídas.

Definição A6: Conjunto de Regras tipo Tsukamoto (TSUKAMOTO:1997)

Um conjunto de regras tipo Tsukamoto é um conjunto similar ao definido em A5 com a diferença de que os termos definidos nos conseqüentes possuem funções de pertinência monotônicas.

Definição A7: Conjunto de Regras tipo Sugeno (SUGENO:1988)

Um conjunto de regras tipo Sugeno possui os antecedentes iguais ao conjunto de regras definido em A5 e como conseqüentes valores reais, os quais são calculados em função dos antecedentes. Uma regra do tipo Sugeno possui a forma:

$$R^{(i)} : Se\ x_1\ é \ldots x_k\ é\ E_i^k\ Então\ y_1 = f_1(x_1 \ldots x_k)\ e \ldots\ y_m = (x_1 \ldots x_k) \quad \text{(A19)}$$

onde f_m é uma função polinomial de $x_1 \ldots x_k$. O grau do polinômio estabelece a ordem do modelo definido pelo conjunto de regras. Se f_m é constante então as regras definem um modelo nebuloso tipo *Sugeno* de ordem zero. Se f_m é de primeiro grau, então o modelo é de primeira ordem e assim sucessivamente.

A6.2 Propriedades do Conjunto de Regras

Em seguida, introduzir-se-ão algumas propriedades úteis de um conjunto de regras (DRIANKOV:1993) (PREDYCZ:1993):

Definição A8: Propriedade de Completeza -"completeness"

Um conjunto de regras é completo se, para qualquer combinação de valores de entrada, resulta em um valor de saída apropriado. Em outras palavras, um conjunto de regras é completo se, para qualquer ponto no espaço de entradas, pelo menos uma regra for ativada. Desta forma não devem existir "regiões" no espaço de entrada que não sejam cobertas pelo conjunto de regras.

Definição A9: Propriedade de Consistência

Um conjunto de regras é consistente se todas as regras com um mesmo antecedente possuam um mesmo conseqüente, em caso contrário o conjunto de regras é inconsistente ou contraditório.

Definição A10: Propriedade de Continuidade

Um conjunto de regras é contínuo se não possui regras "vizinhas" com conjuntos de saída que tenham uma interseção vazia.

Duas regras são "vizinhas" se possuem ao mesmo termo comum no antecedente enquanto os outros termos do antecedente são "vizinhos", isto é suas funções de pertinência se sobrepõem. Se esta propriedade não se cumpre, diz-se que o conjunto de regras é descontínuo.

A6.3 Parâmetros internos e externos de um MN

Um conjunto de regras, como definido em A5, estabelece uma relação entre as entradas e as saídas (isto é, entre os antecedentes e os conseqüentes). Numa aplicação

específica devem ser definidos todos os parâmetros do conjunto de regras, isto é: o número de antecedentes (entradas), o número de conseqüentes, o números de regras, a interpretação dos conectivos, etc. O processo de seleção dos parâmetros pode ser visto como um processo de "sintonização" do modelo. Em seguida, descreve-se cada um destes parâmetros. Para facilitar esta revisão dividem-se os parâmetros em internos (ou estruturais) e externos. Dentro dos parâmetros externos consideram-se:

- O número de entradas $X=x_1 \ldots x_k$ é o número de saídas $Y= y_1 \ldots y_m$
- O número de termos para cada entrada E: $E^i = E^i_1 \ldots E^i_n$ e para cada saída i: $S^i = S^i_1 \ldots S^i_n$
- A forma das funções de pertinência dos conjuntos definidos no espaço de entrada/saída: triangulares, gaussianas, trapezoidais, lineares por trechos, etc.
- O número de regras $R=R_1 \ldots R_n$
- Os fatores de escalonamento de entrada/saída quando se usam domínios normalizados; isto é, as transformações de escala que mapeiam os valores físicos das entradas no domínio normalizado. Igualmente, o escalonamento dos valores de saída será normalizados aos valores reais.

Como parâmetros internos ou estruturais consideram-se:

- A forma da definição matemática do conectivo "e" *(and)* nos antecedentes das regras (ou dos outros operadores se forem usados outros conectivos). Chamar-se-á este operador de T. Para a regra i definida em A5 tem-se:

$$\mu_{I^i}(x_1 \ldots x_k) = T(\mu_{E_i^1}(x_1) \ldots \mu_{E_i^k}(x_k)) \tag{A20}$$

- A forma da definição matemática da implicação no conjunto de regras (ou seja, a forma de interpretar o significado de uma regra "Se-Então"). Chamar-se-á este operador de *"Im"* que representa a relação nebulosa entre o espaço de entrada/saída. Considerando a regra i, por exemplo:

$$\mu_{R^i}(x_1 \ldots x_k, y_1 \ldots y_k) = Im(\mu_{E_i^1}(x_1) \ldots \mu_{E_i^k}(x_k), \mu_{S_i^1}(y_1), \ldots \mu_{S_m^k}(y_m)) \tag{A21}$$

- A forma da definição matemática da composição de relações nebulosas que é usada na regra composicional de inferência. Considerando a regra i uma entrada $A_1 \ldots A_k$ e a função de pertinência do conjunto nebuloso de saída l, $S^{l^i}_i$, que pode ser expressada por:

$$\mu_{S_l^{l^i}}(y_l) = SUP_{(x_1 \ldots x_k)}(T(\mu_{A_1}(x_1), \ldots, \mu_{A_k}(x_k)), \mu_{R^i}(x_1 \ldots x_k, y_1 \ldots y_m)) \tag{A22}$$

- A forma de definição matemática do conectivo também *(also)* para a agregação das regras. Para a saída l o conjunto resultado da avaliação de todas as regras (S^{l^i}) considerando uma entrada $A_1 \ldots A_k$ é:

$$S^{l^i} = \bigcup_{i=1}^{i=n} S_i^{l^i} \tag{A22}$$

A6.4 Interface de Entrada: Fuzzificação

A interface de entrada realiza a operação complementar à interface de saída. Ela mapeia um ponto real $x=(x_1, \ldots, x_n)$ num conjunto nebuloso E. No processo de fuzzificação pode-se distingir dois casos: *singleton* e não *singleton.* O *singleton* é o mais popularmente usado, nele a entrada x é convertida num conjunto nebuloso com suporte[7] num só ponto x: $\mu_E(x') = 1$ em $x' = x$ e $\mu_E(x') = 0$ em $x' \neq x$.

O fuzzificador não *singleton* $\mu_E(x') = 1$ em $x' = x$. $\mu_E(x')$ descreve na medida que x vai-se distanciando de x'. Este método é usado em modelos onde os dados são contaminados com ruído (MENDEL:1995). O fato de usar o *singleton* simplifica o processo de inferência e sua implementação em *hardware.*

A6.5 Interface de Saída: Defuzzificação

O defuzzificador é a interface que mapeia conjuntos nebulosos à números reais. Assim, sendo y um valor real e S um vetor de conjuntos nebulosos, S_i com $i=1 \ldots n$, o objetivo da defuzzificação é determinar um valor y_s (real) que corresponda a S. Existem muitas possibilidades de implementação desta interface. Desafortunadamente não existe uma base teórica que permita escolher qual é a melhor. Defuzzificação tem sido mais uma arte que uma ciência (MENDEL:1995), porém existe interesse na comunidade dos "teóricos da lógica nebulosa" em formalizar o processo de defuzzificação (YAGER:1993) (SAADE:1996) (PEDRYCZ:1996) (RONDEAU:1997). A seguir discutir-se-ão alguns métodos práticos mais usados. Na defuzzificação podem-se distinguir dois casos com relação à interpretação do conectivo *"also"* no conjunto de regras (CORDÓN:1997):

- Agregação primeiro, Defuzzificação depois (AD). Neste método é usado um operador para representar o conectivo "also", o qual realiza a agregação dos conjuntos inferidos. Logo, é usado um método de defuzzificação para calcular a saída a partir do resultado da agregação.

- Defuzzificação primeiro, Agregação depois (DA). Neste método a contribuição de cada conjunto inferido é considerada individualmente, logo, a saída é calculada usando algum critério pré-estabelecido que pode ser: valor médio ponderado, valor máximo, etc. Note-se que este método é uma outra forma de interpretar o conectivo "also".

Com o intuito de mostrar estes métodos considere-se S_i' como o conjunto de saída que resulta da ativação da regra R_i ($i= 1 \ldots N$). Existem vários valores de importância no processo de defuzzificação com relação a este conjunto:

[7] O suporte de um conjunto nebuloso é o conjunto dos pontos do conjunto universo para os quais o valor de pertinência é maior que zero.

- A área definida pela função de pertinência de $S_i^{l'}$ em Y_l :

$$A_{S_i^{l'}} = \int_{Y_l} \mu_{S_i^{l'}}(y)dy \tag{A23}$$

- O centro de gravidade da sua função de pertinência :

$$C_{S_i^{l'}} = \frac{\int_{Y_l} y\mu_{S_i^{l'}}(y)dy}{\int_{Y_l} \mu_{S_i^{l'}}(y)dy} \tag{A24}$$

- Avaliando a função de pertinência neste ponto, define-se:

$$Av_i^{l'} = \mu_{S_i^{l'}}(C_{S_i^{l'}}) \tag{A25}$$

- A altura máxima do conjunto de saída (*A*max.) :

$$A\max_{S_i^{l'}} = Sup\mu_{S_i^{l'}}(y) \forall y \tag{A25}$$

- O valor máximo do conjunto de saída (*V*max.):

$$V\max_{S_i^{l'}} = y \in Y_l | \mu_{S_i^{l'}}(y) = A\max_{S_i^{l'}} \tag{A26}$$

Se existem vários pontos que satisfazem esta igualdade, o valor máximo pode ser calculado considerando um critério adicional, como por exemplo: o menor dos valores máximos, o maior, o valor médio, etc.

Um outro parâmetro que é considerado no processo de defuzzificação em alguns casos é o grau de "casamento" da entrada no antecedente da regra o qual depende da interpretação do conectivo *"and"* (também chamado grau de "ativação" da regra):

$$G_i = T(\mu_{E_l^1}(x_1) \cdots \mu_{E_l^k}(x_k)) \tag{A27}$$

Considerem-se inicialmente os métodos definidos como AD.

- *Defuzzificador baseado no cálculo do centro de gravidade (CG):*

Neste caso é calculado centro de gravidade da função de pertinência do conjunto nebuloso resultante do processo de agregação das regras. Se $S_{j'}$ é o conjunto nebuloso resultante do processo de agregação com suporte Z (os pontos onde a função de pertinência é maior que zero), então, considerado o suporte discretizado em I níveis, o CG $\bar{y}$ é dado por :

$$\bar{y} = \frac{\sum_{i=1}^{I} y_i \mu_{S_{j'}}(y_i)}{\sum_{i=1}^{I} \mu_{S_{j'}}(y_i)} \tag{A28}$$

- *Defuzzificador baseado no valor máximo:*

  Neste método é escolhido o valor máximo do conjunto resultado da agregação.

- *Defuzzificador baseado no Valor Médio dos Máximos (VMM)*:

Neste método o defuzzificador examina o valor máximo do conjunto resultado da defuzzificação. Se só existe um ponto com esta característica o valor de saída é igual ao caso anterior. Se existem vários pontos, então o valor de saída pode ser calculado com o valor médio deles.

Considere-se agora o caso DA para o qual existem muitas propostas (RUNKLER:1993), (HELLENDORN:1993), (WANG:1994) (CORDÓN:1997). Nas expressões a seguir, $\bar{y}$ é o valor resultante do processo de defuzzificação :

- *Defuzzificação baseada no CG ponderado com o grau de casamento das entradas:*

$$\bar{y} = \frac{\sum_{i=1}^{N} G_i C_{S_i'}}{\sum_{i=1}^{N} G_i} \tag{A29}$$

- *Defuzzificação baseada no CG ponderado com a área da função de pertinência :*

$$\bar{y} = \frac{\sum_{i=1}^{N} A_{S_i'} C_{S_i'}}{\sum_{i=1}^{N} A_{S_i'}} \tag{A30}$$

- *Defuzzificação baseada no CG ponderado com a altura máxima:*

$$\bar{y} = \frac{\sum_{i=1}^{N} A\max_{S_i'} C_{S_i'}}{\sum_{i=1}^{N} A\max_{S_i'}} \tag{A31}$$

- *Defuzzificação baseada no CG ponderado com o casamento e o suporte do conjunto nebuloso*:

$$\bar{y} = \frac{\sum_{i=1}^{N} G_i C_{S_i'} Z_i}{\sum_{i=1}^{N} G_i Z_i} \tag{A32}$$

Onde Z_i é proporcional ao suporte do conjunto nebuloso de saída.

- *Defuzzificação baseada no CG ponderado com a avaliação do centróide :*

$$\bar{y} = \frac{\sum_{i=1}^{N} Ev_i^{l'} C_{S_i'}}{\sum_{i=1}^{N} Ev_i^{l'}} \tag{A33}$$

Poder-se-ia usar, também, outros métodos, tais como: defuzzificação baseada no valor máximo ponderado com a área da função de pertinência defuzzificação baseada no valor máximo ponderado com a altura máxima, etc.

Outros métodos de defuzzificação baseiam-se na seleção de um conjunto nebuloso na saída com base num grau de importância assumido a priori, sobre o qual é feita a defuzzificação:

- *Defuzzificação baseada no CG do conjunto nebuloso com maior área :*

$$\bar{y} = C_{S_K''} \tag{A34}$$

onde : $$S_K'' = S_K''|A_{S_j''} = Max(A_{S_j''}), \forall_i \in 1 \cdots N \tag{A35}$$

- *Defuzzificação baseada no CG do conjunto nebuloso com maior altura* :

$$\bar{y} = C_{S_K''} \tag{A36}$$

onde :

$$S_K'' = S_j''|A\max_{S_j''} = Max(A\max_{S_j''}), \forall_i \in 1 \cdots N \tag{A37}$$

Também poderia usar-se outros métodos, tais como: defuzzificação baseada no *CG* do conjunto nebuloso com maior grau de ativação, defuzzificação baseada no valor máximo do conjunto nebuloso com maior área, defuzzificação baseada no valor nebuloso com maior altura, etc.

Alguns dos métodos empregados no caso AD poderiam ser usados no caso DA com algumas modificações. Exemplo:

- O *valor médio dos máximos VMM:*

$$\bar{y} = \left[\sum_{i=1}^{N} V\max_{S_i''}\right] \Big/ N \tag{A38}$$

- O *Centro das somas :*

$$\bar{y} = \frac{\int_{Y_i} y . \sum_i \mu_{S_i''}(y) dy}{\int_{Y_i} \sum_i \mu_{S_i''}(y) dy} \tag{A39}$$

Finalmente, considera-se o método introduzido por Yager (FIGUEIREDO:1993) e o método de Sugeno (SUGENO:1988) em que a_i e b_i são os dois pontos no espaço de saída, tais que $G_i = \mu_{S_i''}(a_i) = \mu_{S_i''}(b_i)$ (sendo $\mu_{S_i''}(x)$ a função de pertinência do conjunto nebuloso de saída definido pela regra *i*), a defuzzificação é feita com base no valor médio destes pontos, isto é $t_i = \frac{a_i + b_i}{2}$, assim a defuzzificação será dada por :

$$\bar{y} = \frac{\sum_i^N G_i t_i}{\sum_i^N G_i} \tag{A40}$$

No caso do método de Sugeno de primeira ordem a saída está definida por uma função polimonial que depende das entradas. Para a regra i e um modelo com k entradas $(x_1, x_2, \ldots, x_k)$, a saída é calculada como :

$$\bar{y} = \sum_{j=1}^{k} P_j^i x_j + Q_s^i \tag{A41}$$

Sendo P_j^i e Q_s^i constantes; desta forma a saída do fuzzificador está dado por :

$$\bar{y} = \frac{\sum_i^N G_i y_i}{\sum_i G_i} \tag{A42}$$

Este vasto panorama de possibilidades no processo de fuzzificação introduz uma maior complexidade no projeto de um MF.

Nota-se que, do ponto de vista da complexidade computacional, os métodos chamados DA são mais simples. Um outro fator importante de notar é, quando da implementação em *hardware* de um defuzzificador flexível que suporte uma grande variedade de procedimentos, deve-se ter acesso aos valores de importância dos conjuntos nebulosos de saída a partir dos quais compute-se a saída do defuzzificador que é realizada pela adição ponderada dos *singletons* de saída (os centros de gravidade), o resultado da saída é calculado através de uma função sigmóide.

Anexo B

Generalidades da Mão de Obra no Processo de Estiva Ideal em Terminais Marítimo de Contêineres

Neste anexo apresenta-se uma proposta de arranjo da mão de obra para um terminal ideal de contêineres focalizado no processo de estiva dos contêineres. Neste processo requerer-se-á uso de equipamento e infra-estrutura especializado, onde a mão de obra deverá realizar uma série de tarefas, desde a operação de maquinaria até as tarefas manuais, organizada em diferentes áreas operacionais do terminal, de acordo com as tarefas que realizam. A classificação da mão de obra fundamenta-se então na sua qualificação, como será apresentado na continuação.

B1 Tarefas e Responsabilidades da Mão de Obra

Em um terminal de contêineres, os empregados deverão ser organizados para cumprir várias funções associadas com o movimento dos contêineres entre o transporte do navio e a terra. A separação das funções de administração, planejamento e supervisão, das funções operacionais varia de terminal em terminal. A figura B1 mostra o fluxograma das funções relativas às operações de estiva para um terminal ideal. Esta seção fornece uma descrição geral dos deveres e/ou tarefas realizadas em cada área funcional.

B1.1 Superintendente do Terminal e Operações

Há usualmente um superintendente do terminal que é responsável pela operação do terminal de contêineres - com a incumbência de não haver problemas - revendo seu planejamento e funções operacionais. O superintendente de operações é responsável pelos aspectos logísticos do terminal e trabalha com os planejadores dos navios. É freqüente haver um superintendente separado para as operações do navio e dos pátios.

B1.2 Planejadores de Navios e de Pátios

O planejador do navio receberá informações na chegada dos navios acerca da localização dos contêineres que necessitam ser descarregados. O planejador do pátio, por sua vez, receberá informações sobre quais contêineres deverão ser descarregados ou coletados pelos caminhões ou trens.

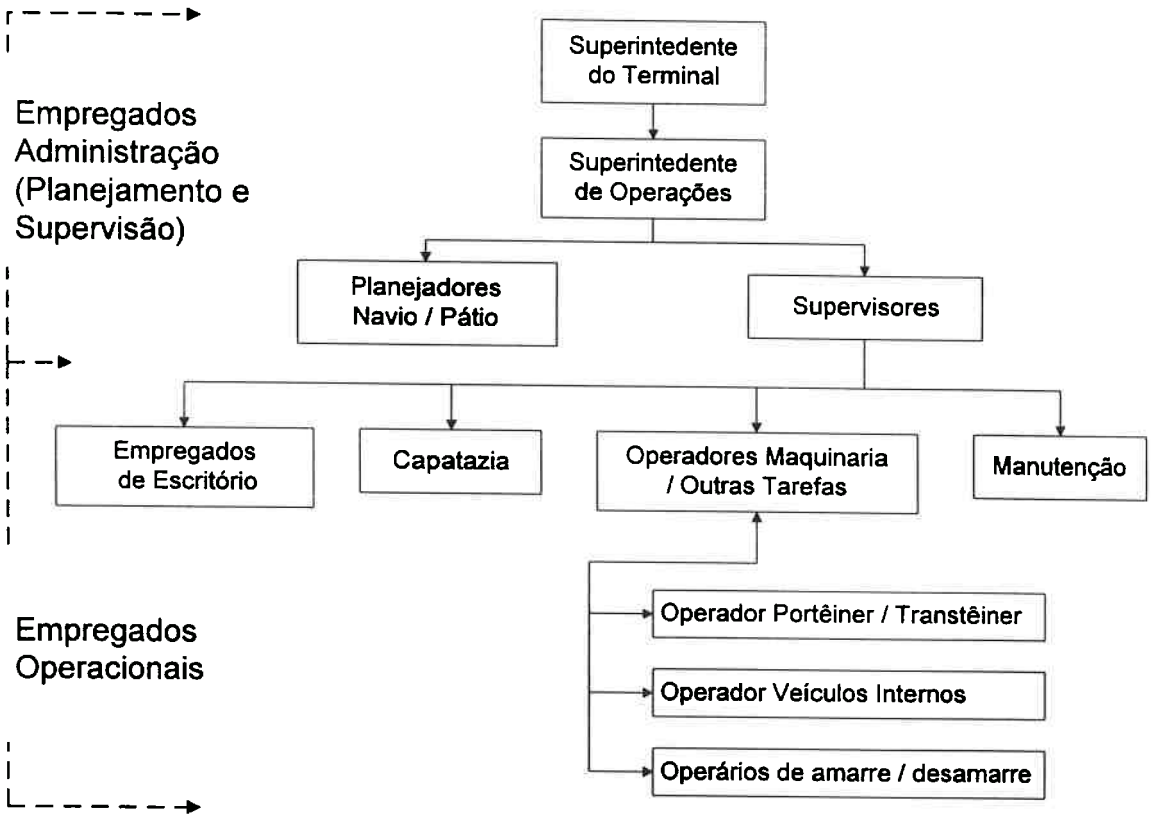


Figura B1 - Fluxograma das Funções Operacionais de um Terminal de Contêineres

Informação acerca de quais contêineres serão carregados nos navios que aportarão deverá ser fornecida ao planejador do navio. Ele determinará onde os contêineres vindos por transporte terrestre deverão ser carregados no navio. O planejador se baseará em vários fatores, incluindo o conteúdo e peso de cada conteiner; determinará onde os contêineres recebidos por transporte terrestre e dos navios deverão ser colocados no pátio. A localização dos contêineres será planejada a fim de que os mesmos sejam deslocados o menor número de vezes, como também seja reduzida a distância dos portêineres.

O trabalho de escritório, com os planejadores, informará aos empregados operacionais aonde os contêineres deverão ser movidos pelos *straddles*, *reach stackers*, *trailers*, caminhões e guindastes. Quando os supervisores ou a capatazia enfrentam problemas inesperados tais como localização indevida de contêineres ou erro na listagem dos navios, a informação será transferida de volta aos planejadores a fim de ajustar as ordens de trabalho.

B1.3 Supervisores

Um supervisor tem a responsabilidade global do trabalho do navio, que pode requerer o esforço de vários turnos para completar a operação. Consulta o superintendente de operações e outros supervisores, decidindo:

- como dispor do capital físico (incluindo o berço, número de guindastes e outros equipamentos e área do pátio);
- quanto de mão de obra é requerida para as operações de carregamento e descarregamento.

O supervisor também deverá coordenar o recebimento e a entrega da carga para o navio. Por exemplo, o desembarque de contêineres chegados no terminal ferroviário poderão requerer de mão de obra e equipamentos extras para as operações de recebimento e entrega para um turno.

Uma vez que os requerimentos de mão de obra tenham sido determinados e os empregados tenham sido alocados a cada turno, o supervisor pode determinar as responsabilidades e tarefas de cada empregado. Supervisores são freqüentemente necessários para a tomada de decisões "sobre as tarefas" quando se enfrentam problemas como recebimentos tardios, quebra de equipamentos e uma precária e incorreta apresentação do navio. Após completado o trabalho do navio, o supervisor deverá reportar à administração sobre o ciclo completo das operações.

B1.4 Empregados Operacionais

Os empregados das operações são definidos como todos os empregados do nível abaixo do supervisor e que desempenham uma gama de papéis e tarefas incluindo as de escritório, manutenção e operação de equipamentos.

B1.4.1 Capatazia

Os capatazes trabalharão em consulta com os supervisores e serão responsáveis por coordenar e monitorar o trabalho dos outros empregados, das operações nos navios e das operações de recebimento e entrega. A capatazia poderá ter alguma sugestão de como o trabalho poderia ser realizado em alguns aspetos da operação, incluindo desde a alocação de empregados e de equipamento entre outras responsabilidades.

A capatazia poderá trabalhar sobre o cais ou o no navio com os outros empregados das operações, dirigir guindastes e equipamentos do pátio, supervisionar turmas de apeação/desapeação e verificar os contêineres entre as listas de seqüenciamento. Um empregado operacional com algum treinamento, como o capataz, poderá ser promovido a realizar estas atividades durante um turno.

B1.4.2 Empregados de Escritório

Na estiva dos contêineres, várias funções de escritório têm sido computadorizadas. Contudo, empregados de escritório continuam realizando tarefas associadas ao rastreamento do movimento de contêineres dentro e fora do terminal. Alguns empregados podem estar envolvidos diretamente com o equipamento, verificando as operações do pátio com as listas de seqüenciamento. Empregados do escritório poderão estar também envolvidos na documentação do processamento do recebimento e de descarregamento. Tarefas adicionais tais como alocação de mão de obra e funções de pagamento de salários, são também realizadas pelos empregados de escritório; em certos terminais, alguns empregados têm sido treinados para trabalhar em áreas operacionais.

B1.4.3 Operadores de Maquinaria / Outras Tarefas

A operação de maquinaria pesada, tal como portêineres, transtêineres e *straddle carries*, e outras tarefas manuais gerais, são realizadas pelos empregados operacionais. As tarefas podem ser empreendidas por um turno completo ou, no caso de equipamento pesado, motoristas poderão realizar outras tarefas durante seus períodos de parada. Alguns exemplos de tarefas manuais que empregados poderão realizar incluem:

- **trabalhos a bordo dos navios:** desaperto de tampas que cobrem as baías, apeação/desapeação dos contêineres, colocação e remoção de *cones* e *twistlocks* usados para assegurar e situar os contêineres;
- **trabalhos no pátio:** ligação/desligação e monitoramento dos contêineres refrigerados e localização dos contêineres; e
- **operações auxiliares:** limpeza de contêineres, da maquinaria e das acomodações e reabastecimento da maquinaria.

As condições sob as quais as tarefas são realizadas variam de dia para dia, por isto, poderão afetar o seqüenciamento e a velocidade das operações de carregamento e descarregamento. Alguns destes fatores incluem avarias nas células guias nos navios, que podem dificultar a estiva do conteiner, pouco equipamento de amarre, avarias dos contêineres e inclemência do tempo.

Operadores dos Portêineres

São usados até três portêineres do cais para o carregamento e descarregamento de contêineres em uma embarcação. Eles são o equipamento pesado de maior custo operando no terminal; um novo porteiner, por exemplo, pode custar aproximadamente US$10 milhões. Dependendo do tipo de guindaste, eles podem levantar de 20-40 toneladas a uma altura de aproximadamente 45 metros.

O operador subirá (por escada ou elevador) até a cabine de controle localizada próximo ao topo do guindaste, em uma altura aproximada de 30 metros. A habilidade para trabalhar nas alturas é, portanto, um critério essencial para o desenvolvimento da proficiência dos operadores dos guindastes.

Competentes operadores de guindastes também requerem excelente coordenação motora e óptica, traduzidas em habilidade de concentração e boa visão.

Uma vez na cabine, o operador do guindaste realizará seu trabalho sentado, observando ao redor e embaixo, através de paredes e chão transparentes. A exposição visual da unidade oferece ao operário direções específicas para planejar sobre que conteiner descarregar do navio ou, inversamente, sobre onde colocar um conteiner que esteja sendo carregado a bordo.

Para descarregar um navio desde o convés ou porão do navio, o operário deverá ajustar o *spreader* (o qual está preso na parte superior por cabos de arame) ao comprimento correto do conteiner de 20 ou 40 pés. O operário, então, descerá o *spreader,* deslocando-o sobre o topo do conteiner para ser erguido. O *spreader* será corretamente alinhado com os quatro cantos do conteiner e é baixado na posição de modo que as travas do *spreader* entrem nos extremos do conteiner. O conteiner, após seu içamento, será deslocado desde o convés ou porão do navio até o pavimento e liberado para ser recolhido por um *straddle carrier* ou colocado sobre um caminhão ou *trailer*. O processo inverso ocorrerá levantando-se um conteiner e situando-o no navio.

Devido à altura em que operador do *spreader se* encontra, faz-se necessário o emprego de um operador habilidoso e experiente, capaz de minimizar o balanço dos cabos de arame, reduzindo o número de tentativas para assegurar o *spreader* ao conteiner. Um operador inexperiente pode levar um tempo significativamente maior para levar um conteiner. Além disso, outros problemas podem agravar as dificuldades tais como ventos fortes e chuvas.

Discussões entre operários mostram que, mesmo trabalhando em cabines, muitas vezes com ar condicionado, boa visibilidade e cadeiras ergonômicas, a natureza da tarefa - observar constantemente para baixo - pode causar esforços prejudiciais às costas, pescoço e olhos.

Operadores de Straddle Carriers

Terminais que operam com *straddle carries* freqüentemente tem 20 ou mais em uso ao mesmo tempo - cada nova máquina custa cerca de US$ 1 a 2 milhões. Dirigir uma *straddle carrier* requer um talento e qualificação similares àqueles necessários para operar guindastes, dado que seu propósito é similar, ou seja, coletar e depositar contêineres. Os *straddle carries* se diferenciam dos guindastes por moverem contêineres ao redor do pátio e os situa e tira dos caminhões.

Mesmo que sejam substancialmente menores que um guindaste, *straddle carries* são, no entanto, altos e estreitos de modo a permitirem o deslocamento ao longo das filas de contêineres estivadas de 2 e 3 unidades de altura no pátio. Assim como nos guindastes, o motorista do *straddle* senta-se numa cabine alta (cerca de 10 metros), recebendo instruções via computador, coletando e depositando contêineres de maneira similar.

A manobra dos *straddle carries* em relação aos guindastes completa um conjunto de problemas potenciais. O projeto de uma *straddle carrier,* alta e estreita, é inerentemente instável. Os motoristas deverão tomar cuidado de modo a manter o centro

de gravidade baixo, levando os contêineres tão baixo quanto possível. Os giros nas esquinas devem ser realizados com especial cuidado, assegurando-se de que não haverá tombamento do veículo; o sistema computarizado da cabina fornece informações sobre a inclinação do veículo quando faz os giros. A visibilidade desde a cabina é geralmente boa, mas há pontos cegos. Os motoristas portanto, precisam ter cuidado com os outros veículos e com o pessoal em terra quando realizam manobras ao redor do pátio. A habilidade de um experiente motorista é também necessária para manobrar o *straddle carrier* sobre a estiva dos contêineres, devido ao mínimo espaço entre as fileiras, evitando com isto, danos na máquina ou nos contêineres.

Apeação/Desapeação

É uma tarefa manual que requer um maior grau de esforço físico que o necessário para operar equipamentos pesados. As obrigações envolvem manusear pesadas varinhas que asseguram os contêineres ao convés do navio. Estas varinhas são projetadas para prevenir o deslocamento enquanto o navio está navegando.

A turma de apeação começa seu trabalho logo após o navio atracar no terminal. Eles abordam o navio pelo portaló ou são içados no interior do navio pelo porteiner. Os membros do grupo se movimentam ao redor do convés do navio e desatam e removem as varinhas de aço dos contêineres que vão ser levantados. Uma vez que a operação é completada, a turma retorna ao navio para atar as varinhas de aço nos novos contêineres carregados.

As condições a bordo do navio podem aumentar a dificuldade da tarefa e também afetar a segurança. Devido a sua demanda de natureza física, a apeação é considerada a tarefa mais popular nas operações de trabalho do navio.

B1.4.4 Manutenção

Os empregados de manutenção são usualmente mecânicos e eletricistas qualificados que realizam a manutenção do equipamento de estiva. Geralmente, os empregados de manutenção se responsabilizam de uma ampla faixa de funções de manutenção incluindo a prevenção, revisão periódica, grandes reparos e substituições. A manutenção é normalmente empreendida em oficinas localizadas nos terrenos do terminal. Contudo, os empregados da manutenção são freqüentemente chamados para reparar ou servir equipamentos em qualquer lugar do terminal onde sejam requeridos.

Alguns trabalhos de manutenção podem serem realizados por terceirizados. Por exemplo, o trabalho de manutenção sobre equipamentos pode ser realizada fora do terminal. Em alguns terminais, empregados de manutenção podem estar disponíveis para trabalhar em outras áreas operacionais para os quais foram treinados.

B2 Classificação da Mão de Obra

Na maioria dos terminais, as classificações baseiam-se no nível do conjunto de qualificações e nos acordos empresariais, incorporando as tarefas realizadas pelos empregados nas diferentes áreas funcionais; a tabela B1 detalha uma classificação por níveis de empregados para um terminal ideal de contêineres.

Os empregados são classificados em níveis de acordo com sua qualificação. Esta é, geralmente, adquirida ou desenvolvida através de programas de treinamento oferecidos pelo empregador em cada terminal.

Empregados podem ser ascendidos a outro nível quando demonstrarem capacidade de a qualificação requerida para esse nível.

A classificação das qualificações não determina estritamente o tipo de tarefa que um empregado realiza ou da antigüidade do empregado sob um mesmo serviço.

De qualquer modo, os aspectos da operação, que requere um alto nível de qualificação, tais como operar uma porteiner, são geralmente realizados por empregados no nível de qualificação de 4 ou acima. Outros equipamentos pesados, tais como transtêineres e *straddle carries* (se os houver), são geralmente operados por empregados classificados com nível 3 ou maior.

A maioria dos empregados novos são treinados em todas as funções e tarefas dentro de uma classificação. Empregados recentemente recrutados são classificados no nível de qualificações 2 seguindo um inicial treinamento introdutório. Estes empregados deverão ter a oportunidade de serem classificados em um nível superior com treinamento adicional e a experiência adquirida no exercício das tarefas.

B3 Arranjos da Mão de Obra

Nesta seção serão examinados os arranjos da mão de obra nos terminais de contêineres. Antes de entrar em detalhes é necessário distinguir a importante entre uma *escalação de operários* e o *nível de operários*.

- Uma ***escalação de operários*** especifica o número de trabalhadores por grupo requeridos para realizar definidas tarefas dentro de um específico precesso, tais como a operação de equipamento pesado, por exemplo dois motoristas para uma portêiner, ou outras, como da apeação/desapeação dos contêineres.
- O ***nível de operários*** refere-se ao total de trabalhadores empregados em um terminal. Isto varia de terminal para terminal e geralmente inclui os empregados permanentes e empregados suplementares ou avulsos.

Tabela B1 – Níveis de Classificação dos Empregados para um Terminal de Contêineres

	Qualificação usada em cada nível de classificação			
Nível	*Escritório*	*Operacional*	*Outros*	*Manutenção*
1	• Introdução e treinamento inicial;			
2	• Uso de equipamentos pequenos; funções diversas no cais e no navio; revisão e conserto de equipamentos básicos; e tarefas básicas de escritório;			
3	• Competência em habilidades operacionais e do escritório, além de:			
	• trato dos recebimentos e das entregas, carregamentos e descarregamentos, etc.;	• Uso de equipamento pesado, operação guinches no navio, revisões básicas;	• Primeiros auxílios;	• Manutenção semi-qualificada (equipamentos e reparos) e uso de ferramentas manuais;
4	• Processo de informações relativas à carga, alocação do pessoal e pagamento;	• Operação de navios especializados e complexos e equipamento de terra;	• Monitoramento, manutenção e controle de plantas frigorificas;	• Pessoal que trata da manutenção;
5	• Auxílio à coordenação do trabalho de pessoal dos níveis inferiores e relacionamento com os supervisores;			
6	• Coordenação e supervisão das áreas de escritório;	• Coordenação e supervisão das operações de estiva;		• Categoria especial do pessoal da manutenção;
7	• Planejamento e coordenação das operações de estiva integradas (incluindo manutenção) e força de trabalho e conexão com os navios e/ou a carga.			

Disso resulta haver uma relação direta entre o nível de operários, o tamanho dos ternos e a escalação dos operários - um aumento na escalação dos operários traduz-se em grandes ternos e altos níveis de operários, mantidos constantes os outros arranjos.

Além da escalação dos operários, que em um princípio deve ser em função da tecnologia, considerando principalmente saúde e segurança, a chave determinante dos níveis dos operários em qualquer terminal inclui:

- demanda dos usuários pelos serviços de estiva, isto é, a movimentação de contêineres;
- arranjo nas alocações dos turnos;
- combinação do uso dos empregados permanentes e avulsos (que deverá ser baseada, em princípio, nos custos da mão de obra e na flexibilidade associada a cada forma de emprego).

Mudanças na demanda e/ou fatores de subministro, observados acima, criam um ímpeto para ajustar os níveis de operários no exercício da estiva em um terminal. Inabilidade ou falhas no ajuste dos níveis de operários (quando há mudanças notáveis nestes fatores e nos arranjos do trabalhos) geralmente atrapalham o desempenho do terminal.

A escalação dos operários e o tamanho dos ternos nos terminais de contêineres selecionados serão examinados na seção B3.1. Os principais fatores e os arranjos de trabalho que afetam os níveis dos operários serão apresentados na seção B3.2.

B3.1 Tamanhos dos Ternos

Um terno é uma unidade básica de trabalho em um terminal de contêineres. Geralmente há ternos separados por navio trabalhado, e para as operações de carregamento e de descarregamento. Nas operações de atendimento a um navio, cada terno é designado a uma porteiner.

Os arranjos de trabalho que afetam o tamanho do terno na operações do navio podem ser:

- Existência ou inexistência de turnos de trabalho normais e/ou contínuos;
- Escalação da mão de obra para equipamento pesado;
- Escalação da mão de obra para outras tarefas como a de apeação/desapeação dos contêineres.

B3.1.1 Turnos de Trabalho Normais e/ou Contínuos

O conceito de turnos contínuos de trabalho tem sido desenvolvido e adotado com sucesso nos últimos dez anos, permitindo a operação dos equipamento durante os tempos

de paradas. Este arranjo de turnos afeta o tamanho e a produtividade dos ternos no trabalho do navio e nas operações de recebimento e descarregamento.

A principal vantagem dos turnos contínuos de trabalho é a redução dos tempos de giro dos navios e dos caminhões. Contudo, com este arranjo, o tamanho dos ternos é maior (ternos deste tipo de turno geralmente possui um capataz extra e dois operários a mais nos trabalhos gerais), isto é, um tamanho médio de 13 empregados comparados com 10 empregados em média para turnos de trabalho normais. Existe também nos turnos contínuos certa dificuldade na programação das trocas e designação das outras tarefas em períodos de curto tempo. Na maioria dos terminais em uso de turnos de trabalho contínuos, os operadores de equipamentos pesados esperam realizar outras tarefas durante os períodos de parada. A administração destes terminais tem confirmado um uso mais efetivo dos motoristas do que no passado, reduzindo ao mesmo tempo o pagamento do tempo não trabalhado.

Acordos empresariais listam as várias atividades que os empregados podem realizar nos tempos inativos. Por exemplo, os operadores inativos das transtêineres podem realizar uma faixa de funções alternativas, incluindo tarefas de apeação/desapeação dos contêineres, atendimento de contêineres refrigerados, tarefas de lavagem e limpeza, reabastecimento e operação de equipamento leves. Operadores inativos dos portêineres não realizam tarefas de apeação/desapeação, a menos que a troca tenha sido completada. Durante os períodos inativos, os operadores dos portêineres, geralmente, permanecem em um lugar da estiva e guiam o guindaste.

Em qualquer terminal, o tamanho dos termos pode variar de "navio para navio" dependendo de alguns fatores, incluindo o tipo de arranjo de turno, normal ou contínuo, o número e tipo de equipamentos disponíveis e a dificuldade ou extensão da tarefa de desapeação.

A quantidade típica de um terno sobre um turno contínuo é mostrado na tabela B2. O número de operadores dos equipamentos pesados, excluindo veículos de transferencia interna, estão numa faixa de 5 a 7, dependendo da configuração do equipamento.

Tabela B2 – Quantidade típica de um Terno por Portêiner para Turnos de Trabalho Contínuos

Categoria	Terminal Típico nº.
Operadores Portêineres[a]	2
Operadores transtêineres e Straddle Carrier[b]	5
Trabalhadores Gerais[c]	4
Capatazia	2
Trabalhadores de Escritório	1
Total	**14**

[a] Os operadores dos Portêineres quando não se encontram operando os guindastes podem trabalhar como capatazia de convés.
[b] Configuração corresponde a uma operação típica de 2 RTG com 3 *Straddle Carrier* por Portêiner.
[c] Motoristas de veículos de transferência interna são incluídos nesta categoria.

B3.1.2 Escalação da Mão de Obra para Equipamentos Pesados

A escalação de operários para equipamentos pesados indica o número de motoristas por terno, designados para operar portêineres, transtêineres, *straddle carries, reach stackers* e veículos de transferência interna.

De um modo geral, mudanças no arranjo têm contribuído significativamente para diminuir o tamanho dos ternos. A escalação de operários para portêineres (dois homens por uma portêiner) tem-se mantido constante. Contudo, a escalação de operários aplicados a *straddle carries, reach stackers,* transtêineres e veículos de transferência interna foram reduzidos de dois operários por uma máquina a três operários por duas máquinas e cinco operários por três máquinas.

B3.1.3 Escalação de Operários para Tarefas de Apeação/Desapeação

A escalação típica de operários para as tarefas de apeação/desapeação é de três ou quatro empregados (ver item B2.4.3 para uma descrição desta tarefa). Acordos empresariais especificam por exemplo que estas tarefas devem ser realizadas por um mínimo de três empregados sob a supervisão geral do contramestre do navio o qual está sendo desamarrado.

Existe uma certa flexibilidade para ajustar o número de empregados designados para este tipo de tarefas. Por exemplo, acordos sindicato-empresa estabelecem que o número de empregados requeridos para apeação/desapeação deverão estar em conformidade com os requerimentos reais da operação.

A dificuldade de uma tarefa de desapeação é freqüentemente desconhecida até que o navio chegue ao terminal; deste modo, a administração do terminal pré-aloca três ou quatro empregados para realizar a tarefa. Se esta, uma vez que o navio chega, é considerada dificultosa, a administração indicará uma re-alocação de empregados trabalhando em outro lugar do terminal para a tarefa de desapeação.

B3.2 Nível de Operários

O nível de operários tem declinado fortemente no cenário mundial. A fim de atender uma demanda que apresenta características variáveis, os estivadores enfrentam um número de restrições para determinar o nível ou combinação de mão de obra necessária. Em termos de empregados permanentes, o número de arranjos de trabalhos, que contribui aos altos níveis de operários, temos:

- tamanho prescrito e composição da mão de obra;
- a ordem de chamada;
- a extensão da saída e dos tempos de folga;

- os custos e dificuldades de implementar redundâncias;
- restrições contratuais.

Estas restrições resultam em altos níveis de operários de mão de obra permanente e subutilização de empregados suplementares. Por sua vez, eles podem conduzir a uma redução da produtividade e altos custos associados à mão de obra.