Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica e de Computação

Departamento de Engenharia de Computação e Automação Industrial

Sistemas Inteligentes para Planejamento de Escalas de Equipagens em Sistemas de Transporte : aplicação a Sistemas Ferroviários

Autor: Rodrigo Almeida Gonçalves

Orientador: Prof. Dr. Fernando Antônio Campos Gomide

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de DOUTOR EM ENGENHARIA ELÉTRICA

Julho 2000

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

G586s

Gonçalves, Rodrigo Almeida
Sistemas inteligentes para planejamento de escalas de equipagens em sistemas de transporte: aplicação a sistemas ferroviários / Rodrigo Almeida Gonçalves.--Campinas, SP: [s.n.], 2000.

Orientador: Fernando Antônio Campos Gomide.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Inteligência artificial. 2. Pesquisa operacional. 3. Algoritmos difusos. 4. Algoritmos genéticos. 5. Heurística. 6. Transporte ferroviário. 7. Trabalhadores do transporte. 8. Maquinistas. I. Gomide, Fernando Antônio Campos. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Agradecimentos e dedicatória:

Agradeço à minha esposa Gina, aos meus pais Paulo e Haydée, aos meus primos Paulo Márcio e Moacir (Léo) e aos demais amigos pelo apoio e compreensão; ao Gomide pela orientação e amizade; ao Romilto pela amizade e paciência; ao Luis Elesbão pela motivação; a todos os outros amigos da MRS Logística S.A e da CVRD pelo apoio intelectual e a FAPESP pelo apoio financeiro.

Dedico este trabalho a todos os brasileiros que, assim como eu, "trazem no peito o cheiro e a cor de nossa terra, a marca de sangue de nossos mortos e a certeza de luta de nossos vivos"[1].

[1] François Silvestre, Cantador

Resumo

Este trabalho apresenta abordagens baseadas em inteligência computacional para um problema de alocação de recursos humanos, mais especificamente, para a geração de escalas de trabalho para equipagens ferroviárias. Esta abordagem leva em consideração uma visão ampla do problema de gerenciamento de equipagens ferroviárias, onde questões que são normalmente negligenciadas na literatura, são avaliadas e levadas em consideração.

Para que isto seja possível, foram desenvolvidos métodos de geração de escalas em dois paradigmas diferentes: o das escalas cíclicas e o das escalas individualizadas. Ambos os casos foram avaliados e testados, com dados reais, por especialistas de ferrovias do país através de um sistema computacional que implementa os algoritmos desenvolvidos.

Dentro do paradigma das escalas cíclicas, foram desenvolvidos dois métodos para criação de seqüenciais de tarefas: um baseado em algoritmos de busca e outro baseado em algoritmos genéticos. O seqüencial de tarefas é posteriormente utilizado para a criação de escalas através de um algoritmo de atribuição, baseado em programação matemática, que distribui as tarefas (os passos do seqüencial) levando em consideração o passado dos funcionários.

Dentro do paradigma das escalas individualizadas, foram desenvolvidos métodos para a geração de escalas levando em consideração não só o passado mas também as necessidades individuais de cada funcionário, bem como as necessidades da empresa como treinamentos e exames médicos.

Abstract

Crew management problems are highly important for many transportation systems such as airlines, railways and public bus transportation. Despite recent advances, scheduling methodologies and decision support systems still need improvement, especially their computational efficiency, practical feasibility and use. This thesis presents methods and algorithms based on computational intelligence for railways crew management.

All developments presented take into account a global view of the crew management and problems often neglected in the literature are considered.

To make it possible, we present methods based into two different paradigms. In the first of them, schedules are generated using crew rostering techniques and, in the other, crew schedules are created in a non-cyclic and more flexible approach.

Computational results and experiences with actual data and real world situations are also reported.

Conteúdo

Listas de figuras, tabelas e algoritmos

Figuras

Tabelas

Algoritmos

1. Introdução

Quando é necessário operar alguma atividade 24 horas por dia, 7 dias por semana, surge a necessidade de se realizar um planejamento adequado da alocação dos recursos humanos. Este não é um problema simples uma vez que inúmeras restrições, muitas vezes complexas, ligadas à qualidade de vida dos funcionários e legislação, devem ser obedecidas.

Embora não haja uma terminologia padrão na literatura, este problema é geralmente denominado de "*manpower scheduling*", aqui considerado como escalonamento de recursos humanos. Alguma confusão é feita porque alguns autores utilizam este termo também para designar subproblemas do problema geral. Além disto, a grande maioria dos trabalhos se concentram em domínios muito particulares, tornando-se difícil o estabelecimento de um jargão único.

Dentro deste contexto podemos destacar duas principais correntes de pesquisa e desenvolvimento: alocação de trabalho para trabalhadores em locais fixos e alocação de trabalho para condutores de veículos em empresas de transporte [Con97]. Na primeira a localização dos trabalhadores é fixa. Exemplos clássicos são hospitais, fábricas e mais recentemente, "*call centers*" [War76], [Hol76]. Neste caso um determinado período de trabalho é dividido em partes denominadas turnos, aos quais são associados os trabalhadores a fim de atender uma determinada demanda.

O horizonte de tempo utilizado para se definir os turnos depende do domínio da aplicação mas normalmente é um dia. Este problema de definição de turnos é conhecido na literatura por "*shift allocation*" ou "*shift scheduling*" [Ayk96], [Bur85], [Bur87]. O problema neste caso é formulado a partir de um conjunto de turnos possíveis e um perfil de demanda. O resultado desta formulação normalmente é um problema de programação inteira.

Uma vez definidos os turnos, torna-se necessário associá-los aos funcionários. Este subproblema é chamado de "*shift assignment*" na literatura. [Tie82], [Koo88], [Lau96]. Um subproblema que surge naturalmente é a definição do mínimo de trabalhadores necessários para atender a uma determinada demanda, ou seja, a redução de custos. Na prática o que normalmente é feito é realizar um procedimento de minimização inicial e posteriormente uma análise de sensibilidade através de relaxações em restrições de qualidade e modificação de parâmetros como intervalo mínimo de descanso e outros.

Alguns autores sugerem formas alternativas de se abordar o mesmo problema, como é o caso de [Tie82] que propõe uma abordagem de 5 estágios para resolver o problema de alocação de trabalho para trabalhadores em locais fixos.

A segunda corrente de pesquisa e desenvolvimento em alocação de recursos humanos (a alocação de trabalho para condutores de veículos em empresas de transporte) se caracteriza pelo fato de que a localização espacial do trabalhador (tripulação de avião, trem ou ônibus) deve ser levada em consideração na modelagem do problema. Outra característica marcante deste problema é que as

atividades realizadas pelos trabalhadores, diferentemente do caso anterior, têm duração variável, chegando em alguns casos a se diferenciar em ordem de grandeza.

A literatura sobre este problema é vasta e concentrada em domínios específicos. São exemplos de domínios diferentes para este problema: escalonamento de motoristas de ônibus urbanos [Fre97], [Por98]; escalonamento de pilotos e comissários de bordo de aviões [Day96], [Van97]; e escalonamento de maquinistas de trens [Con97], [Cap97], [Cap98], [Cap99].

Essencialmente, o problema de alocação de trabalho para condutores de veículos em empresas de transporte tem duas variações possíveis: cíclica e individualizada [Kho94]. Na abordagem cíclica as atividades são atribuídas aos funcionários de forma seqüencial e cíclica através de um núcleo comum denominado de seqüencial de tarefas. Na abordagem individualizada a programação de cada empregado é única e depende somente do(s) período(s) anterior(es) de trabalho e das restrições (legais e sociais). Assim cada empregado tem uma programação totalmente diferente dos demais (e não somente deslocada no tempo como no caso cíclico).

Alguns autores [Con97][Kho94] ressaltam que não é possível a utilização das duas abordagens ao mesmo tempo e destacam as vantagens e desvantagens entre elas. Resumidamente, a abordagem cíclica é mais fácil de ser gerenciada e mais estável (no caso em que não há muitas reprogramações ou que o número de trabalhadores é inferior ao tamanho do período de programação). Já a abordagem individualizada, por outro lado, é mais flexível e atende melhor situações onde há variações de demanda. Esta análise será efetuada com mais detalhes nas seções 2 e 3.

1.1 Planejamento de escalas de equipagens de trens

Em uma empresa de transporte, o gerenciamento de recursos humanos é na verdade parte de um problema de planejamento operacional maior, resumido na Figura 1.1, (adaptado de [Fre97]). Devido ao seu tamanho e complexidade, este problema de planejamento é particionado em subproblemas para torná-lo tratável. Em geral não é factível resolver matematicamente modelos globais do problema de planejamento. A abordagem normalmente utilizada é a de dividi-lo nas seguintes partes: *gerenciamento de veículos* e *gerenciamento de pessoal*. O gerenciamento de veículos, por si só, é um problema de grande complexidade, multi-critério e que envolve inúmeros fatores de difícil quantificação. Neste trabalho não se considera o problema de gerenciamento de veículos. Detalhes referentes a este problema podem ser obtidos em [Bod83], [Bok80], [Fre97] e [Tot93].

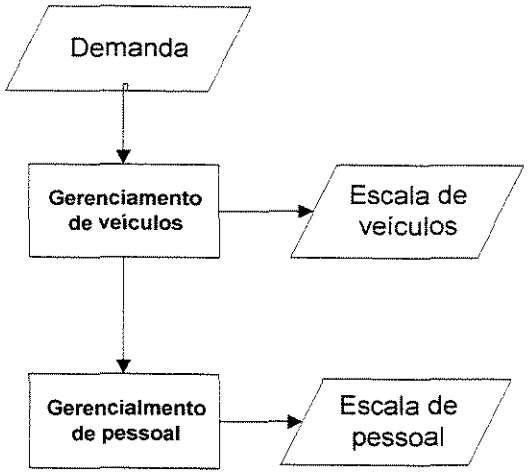

Figura 1.1 - Problema de planejamento

O problema de gerenciamento de pessoal, por sua vez, é composto por vários outros subproblemas, onde a alocação de trabalho (que é o foco desta tese) é somente um deles. São exemplos de outros problemas que constituem o gerenciamento de pessoal: apoio à realização da escala; gerenciamento de exceções; controle de hora-extra; gerenciamento de alocação de férias; gerenciamento de treinamento e aptidões; realização de estudos de dimensionamento; etc... Todos estes problemas afetam de forma direta ou indireta a alocação de trabalho.

No caso específico das empresas de transporte ferroviário, os trabalhadores são denominados de *equipagens* e suas escalas de trabalho são denominadas de *escalas de equipagens*. O termo equipagem pode ter qualquer gênero. Normalmente as empresas do sul e sudeste do país utilizam o termo com o gênero masculino ("o/um equipagem") enquanto as empresas do norte e nordeste utilizam o termo no gênero feminino ("a/uma equipagem"). Uma equipagem ferroviária não é necessariamente um maquinista de trem. Várias outras categorias se encaixam neste termo, como por exemplo, auxiliares, técnicos ferroviários, inspetores (quando os mesmos realizam viagens), etc...

1.2 Inteligência computacional em alocação de recursos humanos

Na resolução da maioria dos problemas do mundo real, ou não dispomos de todas as informações necessárias ou não as temos com precisão. Nesses casos, a lógica e a computação tradicionais são limitadas para representar e resolver estes problemas. No entanto os seres humanos, mesmo com informações incompletas e imprecisas, conseguem resolver problemas complexos. Isto se dá através da capacidade de generalização, aproximação e, através dos erros cometidos, da aprendizagem [Ter92].

A computação tradicional é restrita em sua eficácia porque não explora o poder de generalização, aproximação e aprendizagem que nós, os humanos. Os esforços da inteligência artificial vêm neste sentido, ou seja, dar à máquina parte do poder de generalização, aproximação e aprendizagem dos seres humanos.

Atualmente, podemos dividir sistemas de computação em duas grandes categorias [Zad94], [Gon97]:

- Computação Rígida - visão tradicional do tratamento de informações. Esta categoria apresenta dificuldades no tratamento de informações imprecisas e/ou incertas;
- Computação Flexível ("*Soft Computing*") - visão moderna do tratamento de informações. Esta categoria tem tolerância à imprecisão e incerteza. Com isto se consegue maior robustez, maleabilidade, menores custos e maior eficiência no tratamento de informações.

Uma vez que os pensamentos humanos incluem elementos ilógicos como intuição e inspiração, é virtualmente impossível exprimí-los utilizando-se a lógica convencional. Por isso, a computação rígida tem se mostrado cada vez mais limitada na resolução de problemas complexos.

Os sistemas inteligentes e de computação flexível têm se mostrado promissores em classes de problemas que eram difíceis, ou até impossíveis de serem resolvidos através da computação rígida. São exemplos destes problemas:

- reconhecimento de escrita à mão;
- reconhecimento de fala;
- processamento de linguagem natural;
- reconhecimento de imagens e padrões;
- diagnósticos;
- gravação e recuperação de informação de forma desestruturada;
- aprendizagem;
- raciocínio;
- resolução de problemas.

Até meados da última década do milênio, os paradigmas da computação flexível eram vistos conforme ilustrado na Figura 1.2, onde: LN representa a lógica nebulosa; RN representa as redes neurais e RP representa o raciocínio probabilístico. Esta abordagem fazia com que a computação flexível, apesar de ter cada um dos seus paradigmas já consolidados como áreas de pesquisa e aplicação, ainda fosse discretamente apartada de outros paradigmas mais estabelecidos na grande área de pesquisa denominada genericamente de inteligência artificial.

Hoje se nota uma tendência a uma visão mais global e generalista onde os métodos de computação flexível se integram mais fortemente com outros, anteriormente tidos como clássicos. Esta abordagem é denominada de *inteligência computacional* e é ilustrada pela Figura 1.3. Nesta figura IA representa os métodos tradicionais de inteligência artificial; LN representa a lógica nebulosa; RN representa as redes neurais; SS representa sistemas semióticos; CC representa as ciências cognitivas e finalmente CP computação probabilística (ou evolutiva).

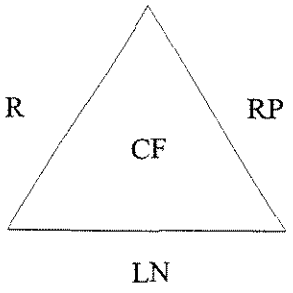

Figura 1.2 - Visão clássica da computação flexível

Os sistemas semióticos (SS) são aqueles baseados na semiótica computacional [Alb97][Gud96]. A semiótica é a disciplina das ciências humanas que estuda os aspectos básicos dos fenômenos da cognição e comunicação. O estudo da cognição visa entender como ocorre a captura e o processamento dos fenômenos do ambiente, enquanto o estudo da comunicação visa compreender como o conhecimento produzido no processo cognitivo é transmitido entre os seres inteligentes. A semiótica se aproxima muito da semiologia [Not95] em seus objetivos porém se distingue no enfoque utilizado uma vez que a semiologia é direcionada à lingüística.

Sendo recente a utilização da semiótica na computação, é natural que alguma confusão com relação à terminologia adotada ocorra. Assim, o termo semiótica computacional tem sido utilizado para várias abordagens diferentes dos mesmos conceitos básicos. Vários trabalhos recentes mostram que a semiótica computacional é uma alternativa interessante para aplicações de agentes (autônomos ou não) e na computação "com emoções" ("*affective computing*") [Gon99], [Pic97].

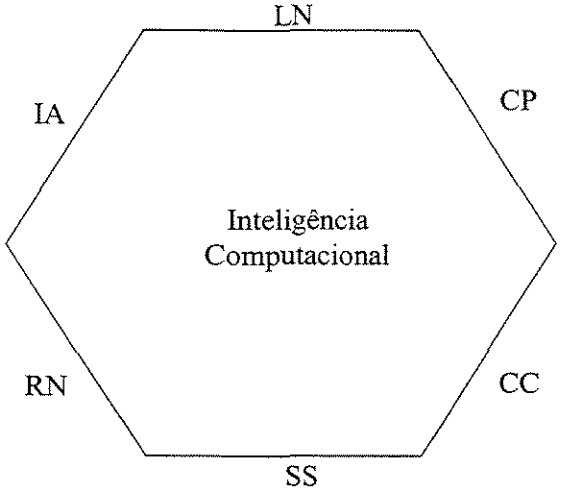

Figura 1.3 – Inteligência computacional

Dentro do contexto do problema de alocação de recursos humanos, a maioria absoluta dos trabalhos se concentram em métodos de pesquisa operacional, mais especificamente em métodos de programação inteira. Apesar disto é cada vez maior a utilização da inteligência computacional e outros métodos alternativos. De acordo com [Por98], umas das razões para este tipo de abordagem é o fato de que a necessidade atual das companhias de transporte está mais ligada a flexibilidade do que a otimalidade. Assim, o objetivo passa a ser encontrar soluções boas (e não necessariamente

ótimas) em um curto espaço de tempo para ajudar na tomada de decisões. Deste modo, métodos heurísticos se tornam interessantes por permitir o uso de qualquer tipo função objetivo. Métodos como busca tabu [Por98], algoritmos genéticos [Cle93][Por98] e algoritmos evolutivos [Lui00], já estão sendo explorados.

1.3 Relevância

Essencialmente, a programação de escalas de equipagens consiste na geração de períodos de trabalho que atendam vários requisitos e restrições. Normalmente o objetivo principal é minimizar o número de equipagens utilizadas para realizar todas as tarefas e viagens necessárias durante um dado horizonte de tempo. Em outras palavras, o objetivo principal é minimizar custos. Porém, com a modernização das ferrovias e os acordos de negociação sindical, outros objetivos tornaram-se importantes. Um destes é a qualidade de vida dos trabalhadores. Em alguns países (e.g. EUA), as ferrovias não geram escalas de trabalho mensal. Nestes casos os maquinistas são funcionários autônomos que prestam serviços. Quando um maquinista retorna de uma atividade, ele entra em uma lista de plantão, podendo ser chamado a qualquer momento para trabalhar novamente. É dada a ele a opção de recusa da tarefa, porém ele só é remunerado pelas horas trabalhadas. A utilização de escalas de trabalho nas empresas brasileiras foi, em alguns casos, uma conquista sindical da década de 80.

A modernização das ferrovias também trouxe a preocupação com a qualidade dos serviços e com a flexibilidade do *centro de escalas*[2] frente aos imprevistos e à sazonalidade típica deste ramo de atividade. Assim tornou-se importante não só a qualidade da escala, mas também a rapidez com que ela pode ser gerada. Até o presente momento, os centros de escala das principais ferrovias brasileiras criam escalas manualmente, num processo demorado e sujeito a erros. Normalmente, uma escala é gerada em 2 ou 3 dias, até 15 dias antes de ser colocada em produção. Assim mudanças de requisitos que ocorrem na véspera do início do mês podem não ser consideradas, causando transtornos e prejuízos. O tempo gasto na confecção da escala também é um limitante que impede sua modificação e manutenção durante sua realização.

Uma escala bem feita e ajustada com a realidade é ponto fundamental para a satisfação da demanda de transporte, redução dos custos e também para o aumento da satisfação dos funcionários. Na maioria das ferrovias, é freqüente ocorrerem paradas de trem por falta de equipagem, levando a prejuízos.

Acredita-se que a flexibilização do centro de escalas leva a confecção de escalas mais próximas da realidade, reduzindo-se os prejuízos por falta de equipagem. Como o crescimento e a modernização das empresas de transporte ferroviário são inegavelmente um ponto essencial para o desenvolvimento do país, o gerenciamento adequado de equipagem traz benefícios para a economia e a sociedade como um todo.

[2] Centro de escalas é o departamento de uma ferrovia que é responsável pela geração e manutenção de escalas de trabalho e outras atividades do gerenciamento de equipagens

1.4 Objetivos e organização

Este trabalho propõe-se a desenvolver novas abordagens para a criação de escalas de equipagens, tanto no paradigma das escalas cíclicas, quanto no paradigma das escalas individualizadas. Sempre que possível os métodos são testados e validados com dados reais fornecidos por ferrovias brasileiras. As metodologias e os algoritmos desenvolvidos contém heurísticas originárias de conhecimento especialista, ingrediente indispensável na solução de problemas de natureza combinatorial como o aqui enfocado.

Após esta introdução, o Capítulo 2 apresenta os desenvolvimentos realizados dentro do paradigma cíclico (também chamado de método seqüencial). Em seguida são apresentados os desenvolvimentos realizados no paradigma individualizado (também chamado de método direto). No Capítulo 4 é apresentado um sistema computacional que implementa os algoritmos e métodos das duas seções anteriores. Por último apresenta-se uma análise dos resultados, algoritmos e métodos, as conclusões e trabalhos futuros.

2. Método do seqüencial de tarefas

2.1 Introdução

Conforme exposto no capítulo anterior, o método baseado em escalas cíclicas é o método de geração de escalas de equipagem tradicionalmente utilizado nas empresas de transporte aéreo, urbano e ferroviário e também muito comum na literatura. Neste método são criadas seqüências de tarefas que posteriormente são transformadas em escalas. Vem daí o nome "método do seqüencial de tarefas", utilizado nas ferrovias do país e que é utilizado neste trabalho. Neste nome, o adjetivo seqüencial foi transformado em substantivo, preservando o mesmo significado: "em que há uma seqüência" [Aur94].

No método do seqüencial de tarefas, o gerenciamento de equipagens pode ser dividido em quatro etapas ou subproblemas que serão mais detalhados nas seções posteriores:

- seqüenciamento ("*scheduling*");
- escalonamento ("*rostering*")[3];
- atribuição ("*pairing*");
- instanciação.

Esta divisão em 4 etapas (ilustrada na Figura 2.1) é uma adaptação dos métodos encontrados na literatura uma vez que não há um consenso nos vários artigos que discutem formas de realizar esta partição, e os melhores modelos para cada uma das partes (e.g. [Van97], [Cap97]). Acreditamos que esta divisão é bastante genérica e engloba as outras anteriores.

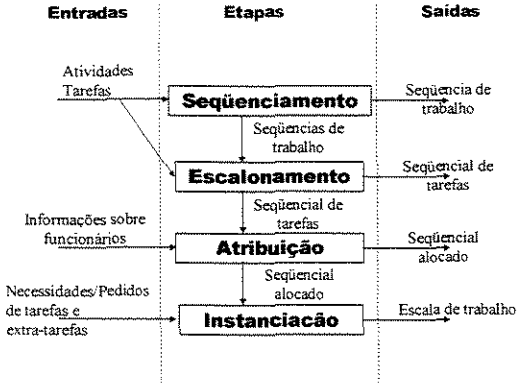

Figura 2.1 – Etapas do método do seqüencial de tarefas

[3] Apesar de acreditarmos que a melhor tradução para "scheduling" é escalonamento e de "rostering" é seqüenciamento, vamos adotar a nomenclatura inversa para manter compatibilidade com o jargão da indústria e algumas outras literaturas em português.

2.2 Conceitos básicos

Antes de iniciar a descrição dos subproblemas que compõem o método do seqüencial de tarefas, se faz necessária a introdução dos seguintes conceitos básicos: tarefa, atividade, segmento de trabalho, seqüencial de tarefas, pernoite e pegada.

Uma tarefa é uma ação de trabalho para uma ou mais equipagens. Normalmente é uma viagem, uma manobra de pátio, uma prontidão, etc. Em alguns casos uma tarefa pode ser subdividida em atividades. Por exemplo, nas ferrovias, uma tarefa de longa duração (e.g. viagem) normalmente é dividida nas seguintes atividades: prontidão ou sobreaviso (tempo em que o funcionário fica no aguardo do trem), passe (tempo de deslocamento até o local do trem), trabalho (viagem), descanso fora da sede, prontidão (sobreaviso) fora da sede, passe fora da sede, trabalho fora da sede (retorno). Uma atividade também é chamada de subtarefa.

Uma atividade pode ser produtiva ou não. Por exemplo, uma atividade de descanso fora da sede não é uma atividade produtiva. Assim uma única tarefa pode ser decomposta em partes produtivas e não produtivas. Somente as partes produtivas devem ser levadas em consideração ao se calcular a carga de trabalho de uma determinada tarefa.

Uma tarefa pode ser fixa ou não-fixa. Uma tarefa não fixa é uma tarefa que caracteriza uma viagem, caso contrário é uma tarefa fixa. Por exemplo, uma tarefa de manobra de pátio é fixa e uma tarefa de transporte de minério é não-fixa.

"Seqüência de trabalho" é a expressão em português utilizada para designar o conceito expresso pelo termo em inglês "*duty*", dentro do escopo de gerenciamento de equipagens. Uma seqüência de trabalho é uma seqüência válida de atividades que podem ser realizadas por uma equipagem, ou seja, é uma tarefa.

Para se determinar a validade de uma seqüências de tarefas, são levados em consideração vários parâmetros e restrições. Entre eles:

- regras da CLT (leis trabalhistas brasileiras);
- tempos das atividades que compõem a tarefa;
- tempos de descanso entre tarefas.

Para exemplificar estes conceitos, a Figura 2.2 ilustra um conjunto de atividades. A Figura 2.3 mostra um conjunto de tarefas (seqüências de trabalho) que contém todas as atividades da Figura 2.2 (sem repetição).

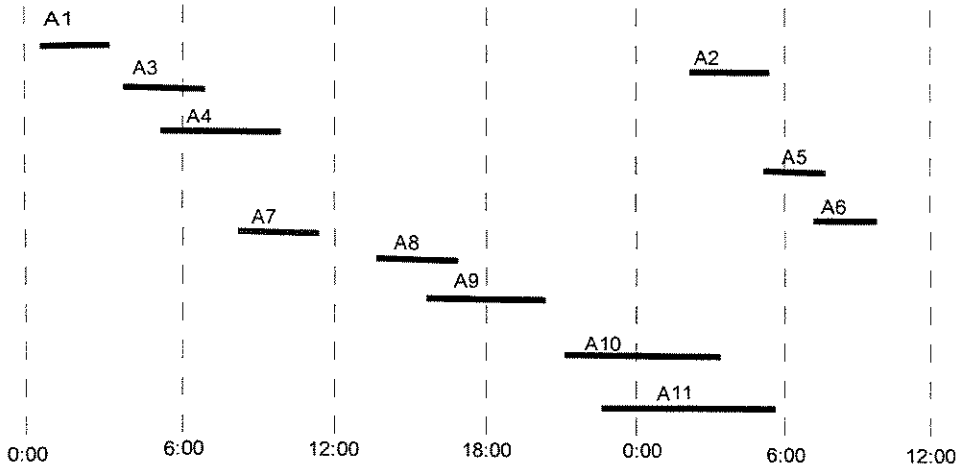

Figura 2.2 – Conjunto de tarefas que necessitam ser cumpridas todos os dias

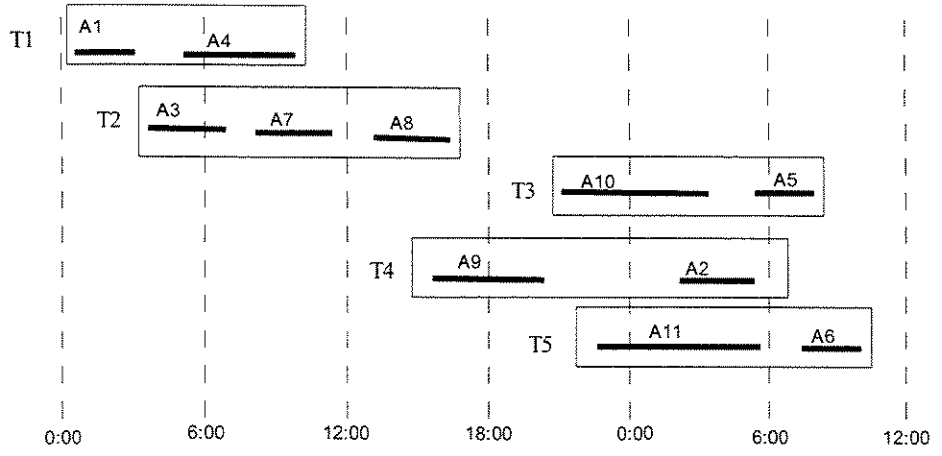

Figura 2.3 – Seqüências de trabalho que contém todas as tarefas

Um "seqüencial de tarefas" é uma tabela auxiliar utilizada na criação de escalas através do paradigma das escalas cíclicas, sendo o responsável pelo nome "método do seqüencial de tarefas". Esta tabela contém sete colunas, uma para cada dia da semana, e quantas linhas forem necessárias. Cada célula da tabela, também denominada de "passo do seqüencial", contém uma tarefa a ser realizada no primeiro dia da escala por um determinado funcionário. Por exemplo, no caso ilustrado na Figura 2.4 (vide legenda na Figura 2.5), o funcionário (ou equipe) 37 irá executar no primeiro dia de sua escala, a tarefa RET que se inicia às 13:00hs. A escala de n dias de um funcionário é o conjunto das n células consecutivas, iniciando-se na célula em que ele foi atribuído para o primeiro dia da escala. No exemplo anterior, a escala do funcionário 37 (atribuído ao passo 1 do seqüencial) é a seguinte: RET 13:00, FES, AUX 09:00, RET 09:00, FOL 09:00, RET 20:00, etc... Por outro lado, a escala do funcionário 36 (atribuído ao passo 2 do seqüencial) é a seguinte: FES, AUX 09:00, RET 09:00, FOL 09:00, RET 20:00, FES, etc...

A	B	C	D	E	F	G
1.4 **RET** **13:00** 37	2.3 **FES** 36	3.2 **AUX** **09:00** 35	4.1 **RET** **09:00** 33	5.0 **FOL** **09:00** 32	6.5 **RET** **20:00** 31	7.4 **FES** 34
8.3 **RET** **03:00** 29	9.2 **RET** **07:00** 26	10.1 **AUX** **18:00** 30	11.0 **FOL** **16:00** 24	12.5 **RET** **21:00** 25	13.4 **FES** 27	14.3 **AUX** **06:00** 22
15.2 **RET** **08:00** 18	16.1 **RET** **15:00** 17	17.0 **FOL** **15:00** 23	18.5 **AUX** **21:00** 5	19.4 **FES** 15	20.3 **RET** **06:00** 16	21.2 **RET** **19:00** 21
22.1 **FES** 19	23.0 **FOL** **00:00** 14	24.5 **PRO** **14:00** 20	25.4 **PRF** 28	26.3 **FES** 13	27.2 **RET** **00:01** 9	28.1 **RET** **11:00** 11
29.0 **FOL** **11:00** 10	30.6 **RET** **22:00** 6	31.5 **FES** 12	32.4 **RET** **06:30** 8	33.3 **RET** **17:00** 7	34.2 **FES** 4	35.1 **RET** **05:00** 3
36.0 **FOL** **05:00** 2	37.5 **MA1** **07:00** 1					

Figura 2.4 – Seqüencial de tarefas

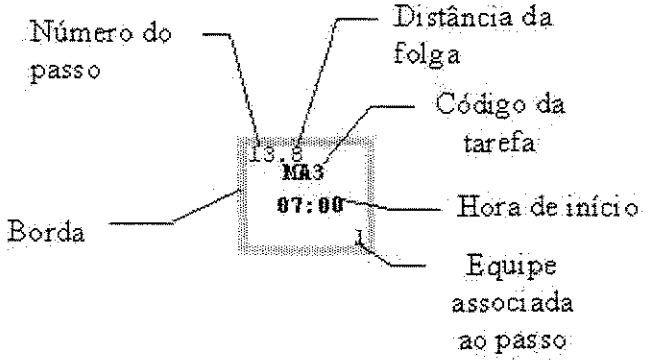

Figura 2.5 – Legenda utilizada no seqüencial de tarefas

Mais adiante, ao analisar escalas, usaremos o conceito de *pernoite*. Um pernoite corresponde à duas noites consecutivas onde um funcionário não dorme em casa. Esta definição é um pouco vaga porque não deixa explícito o que significa "dormir em casa". Esta definição pode variar de destacamento[4] para destacamento, dependendo da sua distância para o centro da cidade mais próxima, formas de transporte, acomodações disponíveis, etc.... No caso geral, considera-se uma noite fora de casa quando o funcionário realiza qualquer atividade de trabalho entre 0:00hs e 5:00hs.

[4] Um destacamento é um local ferroviário onde se concentram as operações de um subconjunto das equipagens de uma empresa ferroviária. Também chamado de sede.

exemplificado na Figura 2.6 onde h_1 é a hora de início da tarefa t_1, J é o tamanho da jornada (duração da tarefa + tempo de descanso) e h_2 é a hora de início da tarefa t_2.

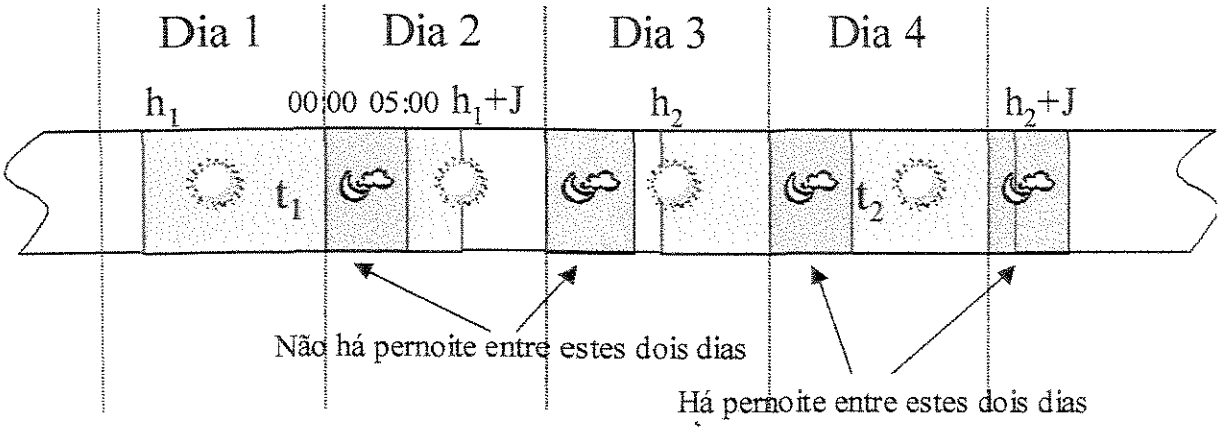

Figura 2.6 – Pernoites

Um outro conceito que será utilizado mais adiante é o de *pegadas noturnas* e *pegadas diurnas*. Pegada é o termo utilizado para designar o instante em que o funcionário inicia uma tarefa. Normalmente uma pegada é noturna se ela se inicia a noite e diurna se ela se inicia de dia. "À noite" e "de dia" aqui não tem relação nenhuma com o conceito de trabalho noturno e diurno estabelecido nas leis trabalhistas e sim com aspectos psicológicos. Portanto uma pegada é noturna se, ao iniciar uma atividade, o sol já tiver se posto e diurna no caso contrário. Este valor é portanto diferente de destacamento para destacamento. Por exemplo, em São Luis (MA), uma tarefa que se inicia às 18:00 tem uma pegada noturna, enquanto em Juiz de Fora (MG) uma tarefa que se inicia às 18:00 pode ter uma pegada diurna (no horário de verão).

2.3 Seqüenciamento e escalonamento via programação matemática e heurísticas

A primeira etapa do gerenciamento de equipagens no método do seqüencial de tarefas é a de seqüenciamento. Esta etapa consiste em encontrar um conjunto conveniente de seqüências de trabalho que contenha todas as tarefas. Conforme mencionado na seção anterior, não há um consenso na literatura para esta divisão e muito menos para a nomenclatura adotada. Por exemplo, a etapa de seqüenciamento também é chamada de planejamento ("*planning*") [Day96], "*pairing*" [Day96] [Fre97], etc.

A segunda etapa é a de escalonamento. Nesta etapa são formados seqüenciais de tarefas a partir das seqüências de trabalho criadas na etapa de seqüenciamento.

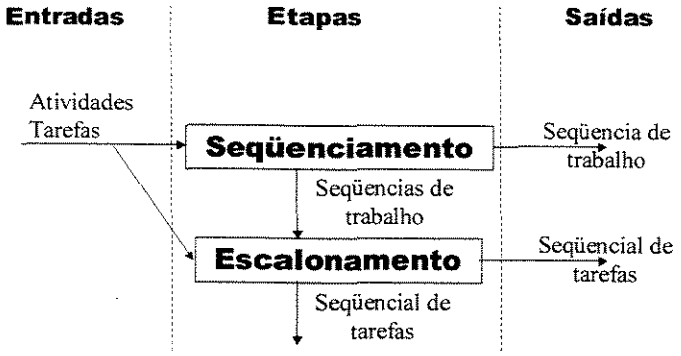

Figura 2.7 – Seqüenciamento e escalonamento

A razão pela qual as etapas de seqüenciamento e escalonamento são reunidas aqui em uma única seção é que elas são muito semelhantes. Ambas têm como objetivo determinar uma seqüência de mínimo custo a partir de um conjunto de itens. No caso do seqüenciamento os itens são atividades e no caso do escalonamento os itens são tarefas (vide Tabela 2.1).

Etapa	Item	Saída
Seqüenciamento	Atividades e tarefas curtas[5]	Tarefas (seqüências de trabalho)
Escalonamento	Tarefas	Seqüencial de tarefas

Tabela 2.1 – Diferença entre seqüenciamento e escalonamento

Caprara et. al. [Cap97] menciona várias razões práticas que justificam esta decomposição. Uma delas é que uma equipagem sempre deve executar seqüências de trabalho que se iniciam e terminam no mesmo local. Outra razão é que tarefas de curta duração devem ter um tratamento diferenciado das tarefas de longa duração.

Conforme já se poderia esperar, os modelos matemáticos existentes na literatura para estas duas etapas são muito semelhantes. Ambos resultam em uma formulação clássica de "*set-covering*" ou de "*set-partitioning*". Pode-se encontrar inúmeras variações de formulações na literatura, dependendo das características específicas de cada caso. As teses de doutorado [Con97] e [Fre97] são referências importantes sobre este assunto. Em particular a tese [Con97] faz uma ampla revisão bibliográfica, mostrando detalhadamente alguns modelos matemáticos existentes na literatura. Aqui, por questão de simplicidade, e para evitar redundâncias com estas referências, somente serão expostos os modelos de maior relevância.

Para o desenvolvimento deste trabalho, foram estudados aproximadamente 25 destacamentos diferentes de pelos menos 3 ferrovias nacionais. Em todas elas se verificam os seguintes fatos:

- há um conjunto muito reduzido de atividades e as tarefas já são pré-definidas pelo usuário.

[5] Em alguns casos, tarefas muito curtas podem ser consideradas como sendo atividades e ser utilizadas para compor outras tarefas

- não é possível trabalhar com o conceito de turnos – as tarefas das empresas estudadas são longas e não uniformes o suficiente para inviabilizar a utilização da idéia de trabalho em turnos, utilizado em um grande número de publicações;
- todas as tarefas se iniciam e acabam no mesmo ponto – mesmo quando uma tarefa implica em deslocamento da equipagem, ela já incorpora tanto a ida quanto a volta ao destacamento de origem. Somente para ilustrar, vamos tomar como exemplo uma tarefa longa. Neste caso, a tarefa tem duração de 38 horas. Destas 38 horas, 4 são de prontidão na sede, 10 são de viagem até outro destacamento, 10 são de descanso no destacamento remoto, 4 são de prontidão no destacamento remoto e 10 são de viagem de retorno ao destacamento original;
- tanto as tarefas longas quanto as tarefas curtas tem duração total (incluindo o tempo de descanso) na ordem de grandeza de dias.

Estas observações nos levam a concluir que não há a necessidade da etapa de seqüenciamento uma vez que ela é implicitamente realizada pelo usuário durante a criação das tarefas, ou seja, quando as subtarefas são reunidas em uma única tarefa.

Assim sendo, no caso particular das ferrovias brasileiras, as duas etapas iniciais são reduzidas à etapa de escalonamento.

2.3.1 Modelo matemático

Conforme mencionamos na seção anterior, grande parte dos artigos que elaboram soluções para os problemas de seqüenciamento e escalonamento baseiam-se no modelo clássico de "*set-covering*" ou "*set partitioning*" [Orl93]. O maior problema encontrado nestas formulações não é a formulação propriamente dita, e sim sua implementação. Normalmente estes modelos resultam em problemas de programação matemática com milhares de restrições e milhares (em alguns casos centenas de milhares) de variáveis inteiras [Cer98]. Muitos trabalhos discutem somente técnicas de resolução para estes problemas. Normalmente são utilizadas técnicas de geração de colunas ("*column generation*") [Fre97]. Outras técnicas podem ser encontradas em [Ken95][Ash92][Niz96][Bea96].

Caprara et. al. [Cap97], faz um resumo dos modelos mais usuais na literatura. Segundo ele, a formulação natural tanto para o problema de seqüenciamento, quanto para o problema de escalonamento, é através de um grafo onde os nós representam os itens do problema (vide Tabela 2.1) e os arcos representam as possíveis transições entre estes itens. Mais especificamente, o problema pode ser interpretado via um grafo direcionado $G=(V,A)$ com um nó $j \in V$ para cada item e um arco $(i,j) \in A$ se e somente se o item j pode preceder o item $i \in V$ em uma seqüência factível. Nesta representação, os dois problemas podem ser vistos como o problema de busca por uma coleção de caminhos em G que minimizem uma função de custo e que cubra todos os nós de G somente uma vez. Conforme a Tabela 2.1, um caminho no modelo do seqüenciamento é uma seqüência de trabalho enquanto um caminho no modelo do escalonamento é uma escala.

No caso geral, a única diferença entre modelos de seqüenciamento e de escalonamento é o fato de que no primeiro deles um circuito somente é válido se iniciar e terminar no mesmo ponto (destacamento), normalmente se adiciona nós no modelo para representar os destacamentos. Assim,

o grafo se torna G=(V,A) onde D (D⊂V) é um conjunto de nós que representam os destacamentos ("*depots*" em inglês).

Há duas formas básicas de modelar o problema de busca no grafo G através de programação inteira. O primeiro modelo associa uma variável de decisão x_{ij} a cada arco (i,j)∈A, onde x_{ij}=1 se o arco (i,j) pertencer a solução e x_{ij}=0 caso contrário. O segundo modelo associa uma variável de decisão y_i para cada caminho factível do grafo G (y_i=1 quando o caminho está presente na solução e y_i=0 caso contrário). O primeiro modelo resulta em uma formulação do tipo "*set covering*" enquanto o segundo modelo resulta em uma formulação do tipo "*set partitioning*". Ambos serão mais detalhados a seguir.

No primeiro caso o modelo matemático é:

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{2.1}$$

sujeito a

$$\sum_{(i,j)\in\delta^+(v)} x_{ij} = \sum_{(i,j)\in\delta^-(v)} x_{ij} = 1, \quad \forall v \in V \setminus D \tag{2.2}$$

$$\sum_{(i,j)\in\delta^+(v)} x_{ij} = \sum_{(i,j)\in\delta^-(v)} x_{ij} = 1, \quad \forall v \in D \tag{2.3}$$

$$\sum_{(i,j)\in P} x_{ij} \leq |P| - 1, \quad \forall P \in \Pi \tag{2.4}$$

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A \tag{2.5}$$

onde:

- V é o conjunto de nós do grafo;
- A é o conjunto de arcos do grafo;
- D é o conjunto de nós relacionados com os destacamentos;
- x_{ij} é uma variável de decisão associada ao arco (i,j)∈A, onde x_{ij}=1 se o arco (i,j) pertence a solução e x_{ij}=0 caso contrário;
- $\delta^+(v)$ e $\delta^-(v)$ são os conjuntos dos arcos de G que entram e saem do nó $v \in V$, respectivamente;
- c_{ij} é o custo de cada arco (i,j)∈A – normalmente uma função do tempo entre os itens (para se minimizar a duração da seqüência);
- D⊂V é o conjunto de nós de destacamentos (na etapa de escalonamento D=∅);
- Π é uma família de subconjuntos de arcos (P) que não podem fazer parte de nenhuma solução factível (por algum motivo de ordem prática qualquer).

Neste modelo as restrições (2.2) e (2.3) obrigam que haja o mesmo número de arcos entrando e saindo de cada nó e que cada nó não associado com um destacamento (v∉D) seja coberto exatamente uma vez. A restrição (2.4) evita que sejam escolhidas algumas seqüências que são infactíveis devido a condições operacionais. Caprara et. al. [Cap97] expõe uma variação deste modelo levando em consideração outras condições operacionais.

O segundo modelo assume que $\Delta=\{C_1,\ldots,C_n\}$ é o conjunto de todos os circuitos factíveis de G (seqüências de trabalho ou escalas, dependendo do caso), com $n = |\Delta|$. Cada circuito C_j tem um custo c_j associado e cobre um conjunto I_j de nós.

Neste caso o modelo matemático é:

$$\min \sum_{j=1}^{n} c_j y_j \tag{2.6}$$

sujeito a

$$\sum_{j:v\in I_j} y_j = 1, \quad v \in V \setminus D \tag{2.7}$$

$$\sum_{j\in S} y_j \leq |S| - 1, \quad S \in \Theta \tag{2.8}$$

$$y_j \in \{0,1\}, \quad j = 1,\ldots,n \tag{2.9}$$

onde:

- V é o conjunto de nós do grafo;
- A é o conjunto de arcos do grafo;
- D é o conjunto de nós relacionados com os destacamentos;
- y_j é uma variável de decisão que indica se o circuito Cj está ou não presente na solução ótima;
- Θ é uma família de conjuntos $S \subseteq \{1,\ldots,n\}$ com a propriedade de que, para cada conjunto S, não são permitidas as soluções que contenham todos os caminhos C_j para $j \in S$.
- c_j é um custo associado com o caminho C_j

Neste modelo, a restrição (2.7) impõe que todo nó, não associado a um destacamento, seja coberto somente uma vez. Já a restrição (2.8), da mesma forma que a restrição (2.4), representa limitações operacionais.

Cada um dos modelos apresentados tem suas vantagens e desvantagens. O primeiro modelo somente pode ser aplicado quando o custo de uma solução pode ser expresso como a soma dos custos associados a cada item. No segundo modelo os custos podem ser associados a seqüências de nós e não aos nós em separado. Por outro lado, é praticamente impossível enumerar todos os possíveis circuitos C_j utilizados no modelo, sendo necessário utilizar uma abordagem do tipo geração de colunas, o que dependendo da forma como são calculados os custos, pode não ser factível. Já o primeiro modelo tem, em relação ao segundo, um número de variáveis bastante reduzido. Mesmo assim o número de variáveis é muito grande, exigindo técnicas sofisticadas e heurísticas para sua solução.

Na prática, o que determina qual dos modelos é utilizado é a forma como os custos podem ser gerados e analisados. De acordo com Caprara et. al. [Cap97], no seu caso específico (*Ferrovie dello Stato SpA* [FerHP]), o segundo modelo é eficiente para resolver o problema de seqüenciamento porque os circuitos são formados de poucos nós. Já o problema de escalonamento é resolvido com o primeiro modelo.

No caso das ferrovias do país, conforme dito na seção anterior, somente há a necessidade de resolver o problema de escalonamento. Da mesma forma que na ferrovia italiana, o primeiro modelo é o modelo natural para este caso. Infelizmente este modelo exige um grande esforço computacional para ser resolvido. Este problema de ordem prática levou a mesma equipe da *Ferrovie dello Stato SpA* a procurar métodos alternativos. Um deles (o mais bem sucedido) será descrito na próxima seção (2.3.2).

2.3.2 Método heurístico

Os algoritmos exatos para resolver os problemas de "*set covering*" são eficientes para instâncias de até alguns poucos milhares de variáveis inteiras. Para instâncias maiores é necessário utilizar algoritmos heurísticos. O algoritmo "*greedy*" é muito utilizado na prática, mas raramente produz soluções de qualidade [Bal80], [Bal90]. Os métodos mais eficientes são os baseados em *relaxação Lagrangeana*. A relaxação Lagrangeana vem sendo extensivamente utilizada desde os anos 70 na resolução de problemas de programação inteira. Em 1991 Beasley [Bea91] detectou que 6% dos trabalhos publicados nas revistas *Operations Research*, *Management Science* e *European Journal of Operational Research* naquele ano eram sobre relaxação Lagrangeana. Para detalhes sobre a teoria de relaxação Lagrangeana o leitor pode se referir a [Geo74]. Para um tutorial resumido da teoria de relaxação Lagrangeana (com exemplos práticos) e referências adicionais, o leitor pode se referir a [Bea91].

A idéia básica da relaxação Lagrangeana é encarar o problema de otimização como sendo um problema que a princípio seria "facilmente resolvido" se não fosse "complicado" por algumas restrições. Assim um problema do tipo "*set covering*" (SCP) pode ser visto como [Fre97] [Cap99]:

$$\min \sum_{j \in N} c_j x_j \tag{2.10}$$

sujeito a

$$\sum_{j \in N} a_{ij} x_j \geq 1, i \in M \tag{2.11}$$

$$x_j \in \{0,1\}, j \in N \tag{2.12}$$

onde:

- A = (a_{ij}) é uma matriz mxn
- c = (c_j) é um vetor n-dimensional de inteiros
- M = {1,...,m}
- N = {1,...,n}

O valor c_j (j∈N) representa o custo da coluna j da matriz A. Assume-se também que c_j>0. Diz-se que a coluna j∈N de A cobre a linha i∈M de A se a_{ij}=1.

A solução normalmente adotada é relaxar todas as restrições exceto a binária (2.12). Assim o problema se torna:

$$L(\lambda) = \min \sum_{j \in N} c_j x_j + \sum_{i \in M} \lambda_i \left(1 - \sum_{j \in N} a_{ij} x_j \right) = \min \sum_{j \in N} \tilde{c}_j x_j + \sum_{i \in M} \lambda_i \quad (2.13)$$

sujeito a

$x \in \{0,1\}, j \in N$

onde:

- λ_j é o multiplicador de Lagrange;
- $\tilde{c}_j = c_j - \sum_{i \in M : a_{ij}=1} \lambda_i$ é o custo Lagrangeano associado com a coluna j∈N

Em 1994 a *Ferrovie dello Stato SpA* [FerHP], juntamente com a *Italian Operational Research Society*, organizou uma competição chamada FASTER, com o intuito de promover o desenvolvimento de algoritmos capazes de produzir bons resultados para algumas instâncias do problemas de alocação de pessoal. O algoritmo descrito em [Cap97] e posteriormente, com mais detalhes, em [Cap99] foi capaz de encontrar as soluções ótimas em 92 dos 94 casos estudados. Este algoritmo é a base para o algoritmo que será apresentado na próxima seção e é a inspiração para o algoritmo proposto neste trabalho.

A idéia por trás do algoritmo heurístico proposto por Caprara et. al. [Cap99] é bastante simples e pode ser aplicada tanto para o problema de *set covering* quanto para o problema de *set partitioning*. Ela se baseia no fato de que para multiplicadores de lagrange λ_i próximos do ótimo, os custos Lagrangeanos $\tilde{c}_j$ fornecem uma informação sobre a utilidade da coluna j na solução do problema. Assim o algoritmo busca rapidamente, através de formulações heurísticas, λ_i próximos do ótimo e iterativamente constrói a seqüência de itens selecionando aqueles com melhores avaliações baseadas nos respectivos custos Lagrangeanos.

O algoritmo em questão é resumido a seguir (será detalhado em seguida):

1) O problema é formulado através da técnica de relaxação Lagrangeana, o dual é resolvido (achar os multiplicadores de Lagrange próximos do ótimo) e os custos Lagrangeanos são armazenados;
2) Escolhe-se um item (vide Tabela 2.1) para ser o item inicial. Observe que, como a escala é cíclica, na verdade não existe um início;
3) Loop:
    4) escolha o melhor item para ser seqüenciado após o último item escolhido
    5) atualiza os custos Lagrangeanos
    6) tenta finalizar uma subseqüência
    7) volta para o passo 5 (fechando o loop)
8) Fim

A formulação do problema utilizada por Caprara et. al. [Cap99] não será descrita aqui com detalhes por não ser necessária ao entendimento do algoritmo. Para maiores detalhes o leitor pode se referir a [Cap97] [Cap98] e [Cap99]. Resumidamente, ela é semelhante à colocada em (2.1) a (2.5).

A diferença está no fato de que ao invés de usar somente um conjunto de arcos, ele usa três. O primeiro (A_1) contém os arcos que ligam um item a outro (como os do modelo (2.1) a (2.5)). O segundo conjunto (A_2) contém os arcos que ligam um item a um descanso simples (um dia de folga). O terceiro conjunto (A_3) contém os arcos que ligam um item a um descanso duplo (dois dias de folga).

A escolha do primeiro item não é aleatória. Dá-se preferência aos itens que tem poucos arcos saindo dele para outros itens. Assim minimiza-se a chance de se chegar a uma situação onde não há tarefas compatíveis com a última escolhida.

A escolha do melhor item para ser seqüenciado após o último item escolhido se dá através de uma análise dos custos reduzidos do problema dual. Escolhe-se um item que minimize uma determinada função utilidade que leva em conta heurísticas de ordem prática e principalmente o incremento na função objetivo (custo lagrangeano).

A atualização dos custos Lagrangeanos é realizada modificando-se o modelo e recalculando o problema dual. A modificação no modelo é realizada de forma simples, colocando como $+\infty$ o custo de todos os arcos relacionados com itens já utilizados na seqüência. Segundo Caprara et. al. [Cap99] não é necessário resolver novamente o problema dual ($O(n^3)$ no pior caso). Pode-se usar técnicas paramétricas para obter a atualização dos custos reduzidos em $O(n^2)$.

Neste modelo, uma seqüência é formada de subseqüências. Uma subseqüência é um conjunto de tarefas seguidas por uma folga semanal (simples ou dupla). Em cada iteração do loop do algoritmo é considerada a hipótese de fechar a subseqüência inserindo-se uma folga. Isto é feito levando-se em consideração a factibilidade e a conveniência conforme algumas restrições operacionais.

O resultado deste algoritmo é um seqüencial de tarefas. Segundo Caprara et.al., foram obtidos ótimos resultados, em tempo computacional muito inferior aos algoritmos tradicionais.

2.3.3 Método heurístico baseado em princípios de ergonomia e preferência declarada

Constantino [Con97] elabora um algoritmo baseado no mesmo princípio de utilização dos custos Lagrangeanos de Caprara et. al. apresentado na seção anterior. É notável neste trabalho a revisão bibliográfica realizada e a metodologia utilizada para se criar os critérios de avaliação e restrições do problema. O autor utilizou conceitos de ergonomia e preferência declarada (PD) para criar uma função utilidade e medir o nível de satisfação de um seqüencial de tarefas. Porém, como este estudo foi realizado com somente um funcionário de um destacamento e de uma única companhia, os critérios levantados e a função obtida não são válidos para o caso geral.

Resolvemos então utilizar critérios válidos para o caso geral (todos os destacamentos de todas as ferrovias) e deixar como trabalho futuro a aplicação de técnicas de PD para customizar os desenvolvimentos obtidos para cada um dos destacamentos particulares (vide seção 6).

2.4 Escalonamento baseado em algoritmos de busca

Seja Ω o conjunto de todos os seqüenciais factíveis. Os métodos propostos por Caprara et.al. [Cap97] [Cap98] [Cap99] e por Constantino [Con97] podem ser vistos como uma busca informada do tipo "*best-first*", sem "*backtracking*" [Lug98] [Nil80], no conjunto Ω. Assumindo que o custo da solução ótima do problema é o somatório dos custos de cada item da escala e que os custos são monotonicamente crescentes com a profundidade da árvore, uma busca informada sem "*backtracking*" encontra a solução ótima. Observe que estas condições são dificilmente observadas na prática. Uma melhoria natural sob esta ótica é a criação de um algoritmo de busca informada com "*backtracking*". Neste trabalho propõe-se um algoritmo do tipo "*branch-and-bound*" (BNB) [Rus95] [Win92] para realizar a busca.

A Figura 2.8 apresenta os conceitos que serão utilizados para descrever o algoritmo.

O algoritmo será primeiramente apresentado e depois detalhado. Sejam:

- *t* uma tarefa representada pela ênupla (h, J, Hd, Hn), onde:
  - h é a data/hora de ínicio da tarefa
  - J é a duração total da tarefa
  - Hd é o número de horas diurnas trabalhadas
  - Hn é o número de horas noturnas trabalhadas
- PDT o conjunto de tarefas a serem utilizadas na escala (programação diária de tarefas).
- AUX o conjunto de tarefas auxiliares utilizadas na escala. AUX = {FOL, FES} onde:
  - FOL é uma tarefa que representa uma folga semanal
  - FES é uma tarefa chamada de "fora de escala". Na verdade FES não é uma tarefa propriamente dita mas sim a indicação de que não é atribuída nenhuma tarefa a um determinado passo do seqüencial de tarefas. Muitas vezes isto ocorre porque naquele passo o funcionário ainda está executando a tarefa do passo anterior.
- S um nó da árvore de busca, representado pela ênupla $S=(E,T_f,Av)$, onde
  - E é uma lista ordenada de tarefas (seqüencial de tarefas). Um seqüencial de tarefas é dito terminal quando todas as tarefas de PDT estão presentes em E, e não-terminal no caso contrário;
  - T_f é o subconjunto de PDT das tarefas $t \notin E$
  - Av é um escalar que fornece uma avaliação para o nó

O algoritmo é o seguinte (vide seção 8.3 para detalhes da notação utilizada):

```
1)    //Procura por uma solução do problema
2)    Procedure acharSolucao()
3)    BEGIN
4)       diaInicial := dia do inicio da escala;
5)       noInicial := Criar_nó_raiz(diaInicial)
6)       resposta = BranchNBound(noInicial,∅)
7)    END
8)
9)    //Verifica se um nó é uma solução
10)   Procedure ehFolha(S) retorna verdadeiro ou falso
11)   BEGIN
12)      Se o nó S for uma folha da arvore, retorna verdadeiro
13)      caso contrário, retorna falso.
14)   END
15)
16)   //Procedimento recursivo de busca
17)   // S é o nó atual
18)   // sol é a melhor solução encontrada até o momento
19)   //Retorna um nó da árvore (a solução) ou vazio (sem solução)
20)   Procedure BranchNBound(S, sol) retorna um nó
21)   BEGIN
22)      //Verifica condição de parada (I)
23)      IF (ehFolha(S)) // Se S for uma folha
24)         IF (S.Av ≤ sol.Av)
25)            retorna S como sendo a solução
26)         ELSE
27)            retorna ∅
28)         ENDIF
29)      ENDIF
30)      //Verifica condição de parada (II)
31)      IF (Tf=∅)
32)         retorna ∅     //(ou seja, chegou a uma folha
33)                              //que não é uma solução)
34)      ENDIF
35)      //Expande o nó atual (Branch)
36)      Criar uma lista de nós Φ inicialmente vazia
37)      Para cada tarefa tar_i∈(T_f∪AUX) verificar se é possível criar
38)         um novo nó F_i:=(E,T_f,Av) onde:[6]
39)            F_i.E = S.E|tar_i  //(tar_i.d = diaAtual)
40)            F_i.T_f = S.T_f - tar_i;
41)      BEGIN
42)         Se for possível:
43)            o nó Fi é criado
44)            uma avaliação para Fi é calculada (Fi.Av)
45)            //Bound
46)            IF ((Fi.Av ≤ sol.Av) && !Bound2(Fi,sol))
47)               o nó Fi é inserido na lista Φ,
48)               mantendo a lista ordenada pelas avaliações
49)            ENDIF
50)      END
51)      //Busca recursivamente a solução
52)      Para todo elemento Fi∈Φ,
53)      BEGIN
54)         aux = BranchNBound(Fi, diaAtual+1, sol);
55)         IF (aux!= ∅)
56)            sol = aux;
57)      END
58)   END
59)   //Verifica a possibilidade de extinguir o ramo
60)   Procedure Bound2(Fi,sol)
61)   BEGIN
62)      Este procedimento será detalhado na seção 2.4.5
63)   END
```

Algoritmo 2.1- Branch and bound

[6] $F_i.E = S.E|tar_i$ significa que a escala associada ao nó F_i é a escala do nó S concatenada com a tarefa tar_i
$F_i.T_f = S.T_f - tar_i$ significa que a tarefa tar_i é retirada do conjunto de tarefas fora do seqüencial associado ao nó F_i

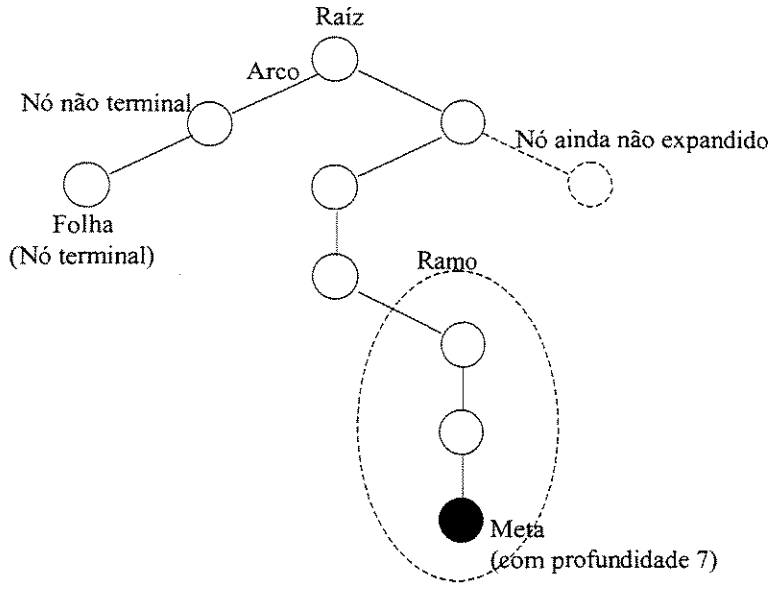

Figura 2.8 - Conceitos utilizados nos algoritmos de busca

Na linha (4) do algoritmo é criado o nó raiz da árvore de busca. Diferentemente de Caprara et. al., não é feita nenhuma análise para se escolher a tarefa inicial. Ela é escolhida aleatoriamente. Observe que o termo "inicial" aqui se aplica ao algoritmo e não ao seqüencial em si porque ele é cíclico, ou seja, não tem início nem fim. Portanto o procedimento `Criar_nó_raiz()` pode simplesmente escolher uma tarefa aleatoriamente ou deixar que o operador o faça.

Para simplificar a aritmética de datas, todos os tempos são manipulados como variáveis inteiras. Estas acumulam o número de segundos desde 01/01/1970 00:00:00 (GMT). Portanto o parâmetro passado para a criação da raiz especifica o dia de início do seqüencial de tarefas para que a primeira tarefa tenha seu atributo h ajustado convenientemente. Para as outras tarefas o atributo h é ajustado levando-se em consideração o dia da primeira tarefa e a profundidade da árvore de busca (cada nível da árvore corresponde a um dia no seqüencial de tarefas, ou seja, a um passo).

Na linha (37) é realizada a ramificação do nó corrente da busca. Neste ponto são testadas as condições de factibilidade e aplicadas algumas regras de seqüenciamento. Todo nó F tem um conjunto T_f de tarefas que não estão presentes no seqüencial de tarefas E. Ramificar um nó é em sua essência, retirar tarefas de T_f e inseri-las em E, observando restrições. A primeira restrição é a factibilidade. Uma tarefa somente pode ser inserida na lista de tarefas se obedecer as seguintes regras:

- $h_3 > h_2$ (vide Figura 2.9), hora de término da primeira tarefa deve ser menor do que a hora de início da tarefa consecutiva;
- o tempo de descanso (Td) deve ser maior do que um valor especificado Td_{min} (vide a Figura 2.9).

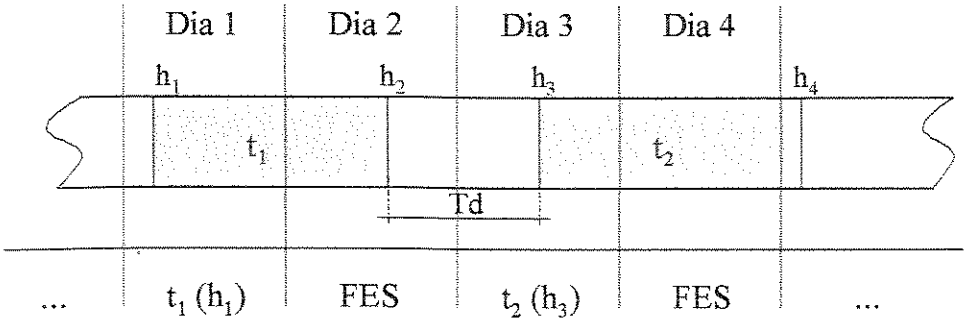

Figura 2.9 - Detalhamento de um seqüencial de tarefas

Em geral, o valor do tempo de descanso mínimo (Td_{min}) varia de destacamento para destacamento. A legislação trabalhista em vigor no país obriga que haja um intervalo maior do que 10 horas entre as tarefas, porém é comum adotar intervalos de 16 horas.

As outras restrições utilizadas são bastante simples uma vez que grande parte das regras de seqüenciamento serão incluídas na função de avaliação, conforme veremos mais adiante. Os conhecimentos utilizados aqui e na função de avaliação foram fruto de um longo trabalho interativo com especialistas de três ferrovias brasileiras. Apesar de serem restrições simples, mesmo em uma única companhia ferroviária elas variam de destacamento para destacamento. Portanto a sua utilização deve ser analisada pelo usuário do algoritmo em cada caso. As restrições mais simples são as seguintes:

- proibir duas tarefas consecutivas com o mesmo código;
- proibir duas tarefas consecutivas do mesmo tipo (fixa ou não-fixa);
- proibir duas tarefas consecutivas com aproximadamente o mesmo horário;

Observe que a linha (37) do algoritmo pressupõe que na ramificação também são consideradas as tarefas do conjunto AUX. Isto significa que podem ser consideradas as tarefas "especiais" folga (FOL) e fora-de-escala (FES). Contudo, estas tarefas não são consideradas em todas as ramificações realizadas.

A folga somente é utilizada em intervalos fixos de passos, normalmente 6 ou 8. O valor 7 não é desejável porque todas as folgas ocorrerão no mesmo dia da semana no decorrer da escala. Quando o intervalo de 8 dias é utilizado, em alguns casos, a cada 4 semanas, uma folga dupla deve ser alocada. A folga também é obrigatória no final do seqüencial de tarefas. Portanto um seqüencial gerado por este algoritmo sempre se inicia com uma tarefa e termina com uma folga.

O fora-de-escala somente é considerado quando o conjunto (lista) T_f não é vazio, porém não é possível encontrar nenhuma tarefa factível para o dia corrente (nível da árvore de busca).

Na linha (44) do algoritmo, uma avaliação é calculada. O algoritmo utiliza uma medida de avaliação para um nó que tem o seguinte formato genérico:

$$f(S) = g(S) + h(S) \qquad (2.14)$$

onde g(S) é a avaliação do nó e h(S) é a estimativa da melhor solução que pode ser encontrada a partir do nó S. Mais especificamente, a avaliação é calculada conforme a equação a seguir, onde cada parcela da soma pode ser também dividida em duas partes: uma avaliando o nó e outra fornecendo uma estimativa da melhor solução que pode ser encontrada.

$$f(S) = k[Entropia(S)] + l[Pernoites(S)] + o[Pegadas(S)] + p[Tarefas(S)] + (1-k-l-o-p)[Tamanho(S)] \quad (2.15)$$

onde

- k,l,o,p são constantes entre 0 e 1;
- *Entropia*(S) é uma função que estima a entropia do seqüencial de tarefas (este conceito será mais detalhado adiante);
- *Pernoites*(S) é uma função que estima o número de pernoites do seqüencial de tarefas;
- *Pegadas*(S) é uma função que estima o número de pegadas repetidas no seqüencial de tarefas;
- *Tarefas*(S) é uma função que estima o número de tarefas repetidas no seqüencial de tarefas;
- *Tamanho*(S) é uma função que estima o tamanho do seqüencial de tarefas (número de passos).

Os valores de todas as funções estão entre zero e um e quanto menor, melhor. Para um determinado nó não terminal S, todas estas parcelas fornecem uma medida do seqüencial de tarefas gerado até o momento e um limite inferior da melhor solução possível de ser obtida a partir do nó S. Assim, na linha (46) é aplicado o princípio da otimalidade de Belman[7] [Lug98], utilizado na programação dinâmica, excluindo-se aqueles nós que sabidamente levarão a soluções piores do que a já encontrada.

As próximas seções detalham cada uma das parcelas da avaliação utilizada no algoritmo proposto.

2.4.1 A medida de entropia: *Entropia*(S)

Conforme mencionado anteriormente, a função *Entropia*(S) mede a entropia de um seqüencial de tarefas. A entropia é uma medida de como a carga de trabalho das tarefas está distribuída dentro do seqüencial de tarefas. Um seqüencial de tarefas com baixa entropia tem sua carga de trabalho uniformemente distribuída ao longo de sua extensão[8]. Já um seqüencial de tarefas com alta entropia tem "concentrações" de trabalho em alguns pontos de sua extensão.

---

[7] Princípio da otimalidade de Belman – o melhor caminho entre um nó inicial e uma meta, passando por um nó particular intermediário, é o melhor caminho do nó inicial até este, seguido pelo melhor caminho deste nó até a meta. Não é necessário considerar nenhum outro caminho que passa por este nó partícular.

[8] Considerando que esta uniformidade é uma conseqüência de uma organização e que o caso contrário seria conseqüência de desorganização

Uma medida de entropia para um seqüencial de tarefas deve ter as seguintes propriedades:

- um seqüencial de tarefas é cíclico. Portanto, independentemente da tarefa escolhida para ser o "início", o valor da entropia deve ser o mesmo. Assim, por exemplo, a entropia de [A,B,C,D] é a mesma de [B,C,D,A];
- um seqüencial de tarefas incompleto deve ter uma entropia que é um limite inferior a todas as possíveis soluções que possam resultar dele. Assim, por exemplo, a entropia de [A,B,_,_] é menor ou igual às entropias de [A,B,C,D] e [A,B,D,C];

Uma forma de medir a entropia de um seqüencial de tarefas é através da seguinte formulação (será detalhada adiante):

$$Entropia(S) = \frac{\sum_{ij} \delta_{ij}}{n^2} \tag{2.16}$$

$$\delta_{ij} = abs\left(\frac{\xi_{ij} - med_i}{med_i}\right) \tag{2.17}$$

$$med_j = \frac{\sum_{i=1}^{n} \xi_{ij}}{n} \tag{2.18}$$

$$\Xi = \begin{bmatrix} \xi_{1,1} & \cdots & \xi_{1,n} \\ \vdots & \ddots & \vdots \\ \xi_{n,1} & \cdots & \xi_{n,n} \end{bmatrix} \tag{2.19}$$

$$\xi_{ij} = \sum_{k=j-1}^{i-1+j-1} x_{(k\%n)+1} \tag{2.20}$$

$$x_i = \begin{cases} \tau_i, & \text{se } i \le m \\ \hat{x}, & \text{se } i > m \end{cases} \tag{2.21}$$

$$\hat{x} = \frac{\sum_{i=1}^{m} \tau_i}{m} \tag{2.22}$$

onde:

- o operador % é utilizado para designar a operação que retorna o resto de uma divisão de inteiros. Por exemplo, 5%2 = 1;
- τ_i é a carga de trabalho da i-ésima tarefa do seqüencial de tarefas E, isto é, τ_i é a soma das horas noturnas e horas diurnas de uma determinada tarefa;
- E é o seqüencial de tarefas (parcial ou terminal) do nó S;

- $n = |\mathrm{PDT}|$, ou seja, o número total de tarefas que devem estar presentes no seqüencial de tarefas;
- $m = |\mathrm{PDT}| - |\mathrm{T_f}|$, ou seja, o número de tarefas presentes no seqüencial de tarefas;

Esta medida de entropia tornar-se-á clara quando se apresentar o algoritmo tabular que permite seu cálculo.

Seja um seqüencial de tarefas terminal E (m=n) correspondente a um nó S da árvore de busca e formada por m tarefas $\{t_1, t_2, ..., t_m\}$. Considere a matriz Ξ $n \times n$ (2.19) inicializada da seguinte forma:

$$\Xi = \begin{bmatrix} \tau_1 & \tau_2 & \cdots & \tau_n \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{2.23}$$

Podemos dizer que cada elemento da primeira linha de Ξ dá a distribuição da carga de trabalho no seqüencial de tarefas no seu nível mais elementar, ou seja, por cada tarefa. Efetuando a operação ilustrada em (2.24), cada elemento da segunda linha dará a distribuição da carga de trabalho em um nível mais alto, mais especificamente, a cada duas tarefas.

$$\Xi = \begin{bmatrix} \tau_1 \downarrow & \swarrow \tau_2 & \cdots & \tau_n \\ \tau_1 + \tau_2 & \tau_2 + \tau_3 & \cdots & \tau_n + \tau_1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{2.24}$$

Agora, repetimos a mesma operação sobre a tabela, para a terceira linha (usando três tarefas da última linha ao invés de duas). Assim cada elemento da terceira linha dará a distribuição da carga de trabalho no seqüencial a cada três tarefas.

Se esta operação for realizada para todas as linhas, com cada linha levando em consideração mais um elemento, obteremos a matriz em (2.25):

$$\Xi = \begin{bmatrix} \tau_1 & \tau_2 & \cdots & \tau_n \\ \tau_1 + \tau_2 & \tau_2 + \tau_3 & \cdots & \tau_n + \tau_1 \\ \vdots & \vdots & \ddots & \vdots \\ \tau_1 + \tau_2 + \cdots + \tau_n & \tau_2 + \tau_3 + \cdots + \tau_n + \tau_1 & \cdots & \tau_n + \tau_1 + \cdots + \tau_{n-1} \end{bmatrix} \tag{2.25}$$

onde todos os elementos da última linha são iguais e fornecem a carga de trabalho do seqüencial como um todo.

A partir da matriz Ξ podemos construir outra matriz Δ (2.26) onde cada elemento δ_{ij} (dado pela equação (2.17)) é o desvio percentual absoluto com relação a média das cargas de trabalho do mesmo nível.

$$\Delta = \begin{bmatrix} \delta_{1,1} & \cdots & \delta_{1,n} \\ \vdots & \ddots & \vdots \\ \delta_{n,1} & \cdots & \delta_{n,n} \end{bmatrix} \tag{2.26}$$

O somatório de todos os elementos de Δ dá uma medida de como está a distribuição da carga de trabalho em todos os níveis, ou seja, a entropia. Portanto uma possível medida de entropia é dada pela equação (2.16). A divisão por n^2 é realizada somente para normalizar o valor, isto é, *Entropia*(S) $\in [0,1]$.

Na página 33 foram consideradas duas propriedades que uma medida de entropia deve atender. A primeira delas estabelece que um seqüencial de tarefas, por ser cíclico, deve ter uma medida de entropia que independa do ponto escolhido como início. Esta propriedade, chamada de comutatividade rotacional, é facilmente verificável (ver sua prova em anexo, seção 7.1).

A segunda propriedade mencionada na página 33 estabelece que um seqüencial de tarefas incompleto deve ter uma medida de entropia que é um limite inferior a todas as possíveis escalas que dele resultam. Até o momento supôs-se que o seqüencial é completo. Suponha que desejamos calcular a entropia de um seqüencial E' com p passos ($p < n$). Se a formação inicial da matriz Ξ for dada conforme (2.27) e (2.28), pode-se provar que o valor de entropia obtido é o limite inferior desejado. A prova está em anexo, seção 8.2.

$$\Xi = \begin{bmatrix} \tau_1 & \cdots & \tau_p & \widetilde{\tau} & \cdots & \widetilde{\tau} \\ \vdots & \ddots & 0 & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \ldots & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{2.27}$$

$$\widetilde{\tau} = \frac{\sum_{i=1}^{p} \tau_i}{p} \tag{2.28}$$

2.4.2 A medida de pernoites: Pernoites(S)

Outra parcela da avaliação utilizada pelo algoritmo é a dada pela função *Pernoites(*S*)* que fornece o número de pernoites em um seqüencial. A definição de pernoite foi colocada na seção 2.2. É comum ocorrer, no dia a dia dos condutores de trens, alguns pernoites. Porém, o aumento do número de períodos nos quais as noites são passadas fora de casa, é bastante indesejável. Dizemos que um pernoite seguido ocorre quando o período no qual as noites são passadas fora de casa é maior do que 2 dias. Se d for o número de dias consecutivos em que as noites são passadas fora de casa, o número de pernoites (*pn*) é dado por (2.29). Já o número de pernoites seguidos (*ps*) é dada pela equação (2.30).Esta definição está ilustrada na Figura 2.10 onde h_1, h_2, h_3 e h_4 são respectivamente as horas de início e fim das tarefas t_1 e t_2.

2.4.2 A medida de pernoites: Pernoites(S)

Outra parcela da avaliação utilizada pelo algoritmo é a dada pela função *Pernoites(S)* que fornece o número de pernoites em um seqüencial. A definição de pernoite foi colocada na seção 2.2. É comum ocorrer, no dia a dia dos condutores de trens, alguns pernoites. Porém, o aumento do número de períodos nos quais as noites são passadas fora de casa, é bastante indesejável. Dizemos que um pernoite seguido ocorre quando o período no qual as noites são passadas fora de casa é maior do que 2 dias. Se d for o número de dias consecutivos em que as noites são passadas fora de casa, o número de pernoites (pn) é dado por (2.29). Já o número de pernoites seguidos (ps) é dada pela equação (2.30).Esta definição está ilustrada na Figura 2.10 onde h_1, h_2, h_3 e h_4 são respectivamente as horas de início e fim das tarefas t_1 e t_2.

$$pn = \min(0, d-1) \tag{2.29}$$

$$ps = \min(0, d-2) \tag{2.30}$$

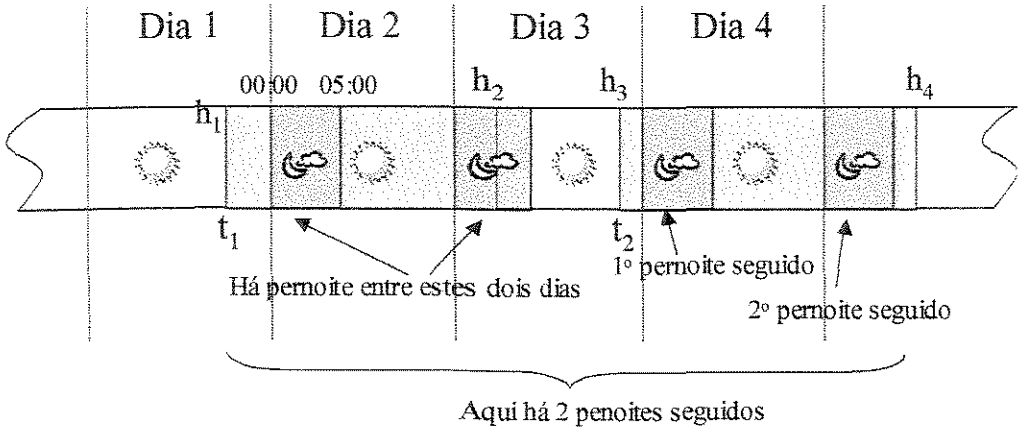

Figura 2.10 - Pernoites

Quando o seqüencial de tarefas é completo, o número de pernoites é facilmente calculado contando-se o número de noites consecutivas fora de casa e aplicando (2.29). Porém, quando o seqüencial de tarefas é incompleto, deseja-se que o valor medido seja um limite inferior para todos os seqüenciais que podem ser geradas a partir dele. Uma possível estimativa é a seguinte:

$$Pernoites(S) = \frac{Pernoites'(S) + In(T_f)}{MAX} \tag{2.31}$$

onde:

- $Pernoites'(S)$ é o número de pernoites do seqüencial incompleto;
- $In(T_f)$ é o número de pernoites inevitáveis em T_f. Um pernoite inevitável ocorre quando ele é implícito na definição de uma tarefa que está no conjunto T_f do seqüencial incompleto (nó não terminal) S. Um exemplo de um pernoite inevitável ocorre na tarefa T2, Figura 2.10;
- MAX é o número máximo de pernoites que pode ocorrer: $MAX = |PDT| - 1$

Levando-se em consideração que só existem dois tipos de pegadas, o melhor caso em um seqüencial incompleto ocorre quando as tarefas em T_f com pegadas diferentes são intercaladas entre si. Pode-se, portanto, medir o pior caso de repetições de pegadas através da diferença do número de pegadas noturnas e diurnas. A expressão que fornece a medida de pegadas repetidas é a seguinte:

$$Pegadas(S) = \frac{Rpp(E) + abs(Pn(T_f) + Pd(T_f))}{MAX} \qquad (2.32)$$

onde:

- *Rpp*(E) é o número de repetições de pegadas no seqüencial parcial E do nó S;
- *Pn*(T_f) é o número de tarefas com pegadas noturnas em T_f;
- *Pd*(T_f) é o número de tarefas com pegadas diurnas em T_f;
- *MAX* é o número máximo de repetições possíveis: $MAX = |PDT| - 1$

2.4.4 A medida de tarefas repetidas: Tarefas(S)

Em muitos casos não é desejável ter seqüencias de tarefas com o mesmo código no seqüencial de tarefas. Por questões operacionais, deseja-se que o funcionário alterne os tipos de serviços prestados. Há alguns casos porém que esta medida não faz sentido uma vez que todas as tarefas do destacamento tem o mesmo código. Estes casos são contornados fazendo o fator multiplicador desta parcela na função de avaliação valer zero (vide equação (2.15)).

A parcela da avaliação dada pela função *Tarefas(*S*)*, de maneira análoga a função *Pegadas(*S*)*, mede a ocorrência de repetição de tarefas com o mesmo código no seqüencial. Em muitos casos porém, devido a ocorrência de vários códigos diferentes no mesmo seqüencial, não se pode fazer a mesma estimativa do número inferior de repetições das tarefas em T_f para os seqüenciais incompletos que foi realizada em *Pegadas(*S*)*. A única forma de fazê-lo seria por enumeração de todas as possibilidades ou através da introdução de um passo de otimização onde o caso ótimo seria encontrado (independentemente da factibilidade). Assim o zero é adotado como limite inferior do número de repetições de tarefas com o mesmo código no conjunto de tarefas restantes para um seqüencial incompleto. Portanto a formulação fica assim:

$$Tarefas(S) = \frac{Rpt(E)}{MAX} \qquad (2.33)$$

onde:

- *Rpt*(E) é o número de repetições de tarefas com o mesmo código no seqüencial parcial E do nó S;
- *MAX* é o número máximo de repetições possíveis: $MAX = |PDT| - 1$

2.4.5 A medida do tamanho: Tamanho(S)

A forma de se obter a avaliação pode depender do objetivo desejado. Apesar da maioria absoluta dos trabalhos utilizarem funções objetivo relacionadas com o tamanho do seqüencial, na

prática muitas vezes é mais importante achar um seqüencial com um número de passos pré-definidos, maior do que o mínimo possível. Isto ocorre porque a programação diária de tarefas pode ser sazonal e um valor médio de trabalhadores necessários já é pré-estabelecido ao passar dos anos. Freqüentemente o custo de demitir um funcionário em um mês e readmitir em um futuro próximo é muito alto uma vez que as equipagens necessitam de treinamento longo e especial. Na medida do possível, os funcionários são remanejados entre os vários destacamentos da empresa, mas mesmo assim sempre há destacamentos com sobra de funcionários. Nestes casos o objetivo é a criação de um seqüencial com um número pré-determinado de passos, ao invés de um seqüencial com o número mínimo de passos. A vantagem de um seqüencial maior é que se pode melhorar a qualidade de outros objetivos por se ter mais "espaço de manobra".

Se o objetivo da busca não é achar o menor seqüencial de tarefas possível e sim achar um que tenha um tamanho pré-determinado, então a função *Tamanho(*S*)* não é calculada e o parâmetro p da equação (2.15) é determinado da seguinte forma:

$$p = (1 - k - l - o) \tag{2.34}$$

Neste caso a linha (37) do algoritmo deve permitir que o FE seja introduzido no meio do seqüencial como uma tarefa comum para preencher "os espaços" existentes em um seqüencial que não tem o tamanho mínimo. Observe porém que o FE é uma "tarefa" que sempre é compatível com todas as tarefas. Portanto, na prática, o FE pode degradar a performance do algoritmo se alguns cuidados não forem tomados, como por exemplo:

- não permitir FE antes ou depois de folgas;
- não permitir dois FE consecutivos.

Cabe aqui também a especificação da função *Bound2* presente na linha 46 do algoritmo. Suponha um nó não terminal S=(E,T_f,Av). Se o tamanho do seqüencial (profundidade máxima da árvore de busca) desejado for h, então se $|S.E| + |T_f| > h$ não há soluções factíveis em nenhum ramo da árvore de busca que tem o nó S como raíz. Assim, quando o objetivo é encontrar um seqüencial com um tamanho predeterminado, pode-se realizar uma "poda" adicional em ramos infactíveis. A função *Bound2* retorna verdadeiro quando o ramo deve ser podado e falso caso contrário. Quando o objetivo é encontrar o menor seqüencial possível, pode-se também realizar esta "poda". Basta comparar com a profundidade da melhor solução encontrada até o momento (*sol*). Assim, se $|S.E| + |S.T_f| > |sol.E|$, o ramo deve ser podado.

Os algoritmos heurísticos apresentados na seção 2.3 utilizam um sofisticado cálculo do limite mínimo de passos do seqüencial para ter uma estimativa da avaliação de cada nó. Aqui usaremos uma medida mais simples baseada na idéia de que a *solução de menor tamanho é aquela que admite menos tempo entre as tarefas*. Assim uma medida do quão boa é uma solução com relação ao tamanho é o somatório do tempo de descanso entre as tarefas do seqüencial. Em um nó terminal (seqüencial completo) a medida ficaria assim:

$$Tamanho(S) = \sum_{i=1}^{n} Td(t_i, t_{(i+1)\%n}) \tag{2.35}$$

onde:

- t_i é a i-ésima tarefa do conjunto E;
- $Td(t_a,t_b)$ é uma função que retorna o tempo de descanso entre duas tarefas consecutivas (vide Figura 2.9).

Note que a somatória da equação (2.35) e o operador '%' garantem que o intervalo entre a última e a primeira tarefa também seja levado em consideração.

Da mesma forma que todas as outras parcelas da avaliação, é necessário que esta parcela forneça um limite inferior para os nós não terminais (soluções incompletas). A equação (2.35) já é um limite inferior, porém este limite pode ser melhorado para aumentar a eficiência do algoritmo. Se o tempo de descanso entre as tarefas tem um valor mínimo td_{min}, então para um nó não terminal $S=(E,T_f,Av)$, uma estimativa do melhor caso (limite inferior) é a seguinte:

$$Tamanho(S) = \sum_{i=1}^{n-1} Td(t_i, t_{i+1}) + td_{\min}\left(|T_f| + 1\right) \qquad (2.36)$$

Observe que em (2.36), os intervalos entre as tarefas restantes, o intervalo entre a última tarefa do seqüencial parcial e o intervalo entre a última tarefa restante e a primeira tarefa do seqüencial parcial são considerados como sendo o mínimo possível (td_{min}).

2.5 Escalonamento baseado em computação evolutiva

Vários trabalhos da literatura sugerem métodos baseados em computação evolutiva para o problema de escalonamento de recursos humanos [Zup99] [Mur98] [Wre95] [Pet95] [Tan95] [Mib93]. Para o problema de alocação de trabalho para condutores de veículos em empresas de transporte os mais relevantes são [Lui00][Por98b][Wre95].

[Por98b] apresenta uma aplicação de algoritmos genéticos para o problema de escalas para condutores de ônibus. A codificação realizada e os operadores genéticos aplicados para resolver o problema são baseados em trabalhos em que algoritmos genéticos são utilizados para resolver problemas de "*set covering*" e "*set partitioning*" [Als96][Bea96b]. Bons resultados foram obtidos e comparados com outros métodos tais como busca tabu [Glo97].

Dentro do contexto de equipagens ferroviárias, até onde temos conhecimento as únicas publicações são [Lui00] e [Lui00b]. Nestes trabalhos, o autor desenvolve um método baseado em computação evolutiva e teoria de conjuntos nebulosos para a geração de escalas. Diferentemente de [Por98b], o autor propõe uma codificação alternativa, sem modelar o problema como sendo um caso de "*set covering*" ou "*set partitioning*". Também foi utilizada uma avaliação baseada em conjuntos nebulosos que se mostrou bastante adequada para o problema. Apesar disto, a codificação utilizada impõe algumas restrições no tamanho máximo das tarefas utilizadas e na distribuição das folgas.

O algoritmo evolutivo aqui proposto não utiliza os problemas equivalentes de "*set covering*" ou "*set partitioning*" para sua codificação e elimina as restrições existentes em [Lui00] e [Lui00b].

Pode-se encontrar inúmeras variações para os algoritmos genéticos na literatura, variando alguns parâmetros e a ordem com que os operadores são aplicados. Destas variações, podemos destacar duas: o GA padrão[9] (conforme [Gol89]) e o GA modificado [Gud99].

A Figura 2.11 ilustra o GA padrão. Neste caso, primeiramente (a) o melhor indivíduo é copiado da população inicial para a população final para preservar a melhor solução. Posteriormente os outros indivíduos são copiados de duas formas diferentes, (b) e (c), de acordo com a probabilidade de "*crossover*", isto é cruzamento. Em (c) os indivíduos são copiados como eles são e em (b) os indivíduos são copiados através de cruzamentos com outros indivíduos. Finalizando um ciclo do algoritmo, alguns indivíduos sofrem mutações (d) conforme a probabilidade de mutação.

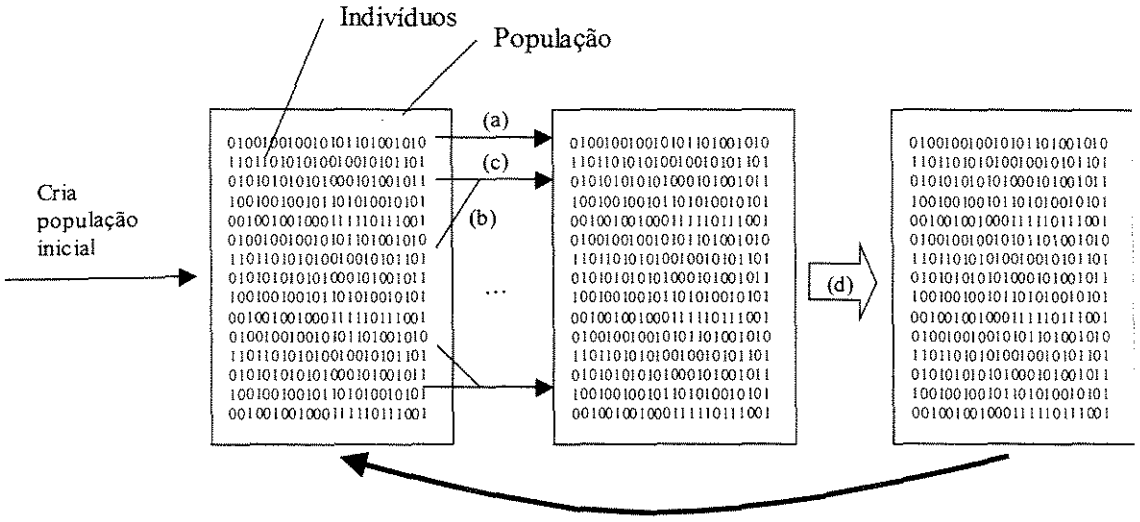

Figura 2.11 – GA Standard

O GA modificado (ilustrado na Figura 2.12) é uma combinação de vários algoritmos encontrados na literatura [Gud99]. Neste caso, vários operadores genéticos são aplicados à população inicial, gerando populações intermediárias em uma fase denominada de reprodução. Após a reprodução é realizada uma fase de seleção onde as populações intermediárias são reduzidas a uma única população final que será utilizada em uma outra geração (ciclo) do algoritmo.

[9] Na realidade não existe uma variação do GA que podemos dizer que é um padrão. O termo padrão aqui é utilizado para enfatizar que o algoritmo apresentado é semelhante ao originalmente proposto por Goldenberg [Gol89]

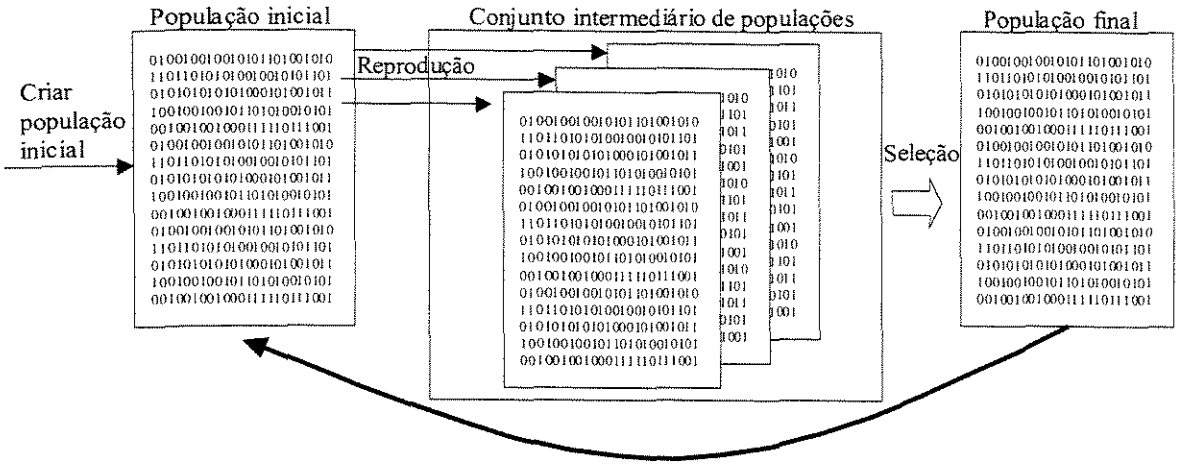

Figura 2.12 - GA modificado

O algoritmo aqui proposto é baseado no algoritmo GA modificado. Portanto, para descrevê-lo é necessário detalhar os seguintes itens:

- codificação utilizada;
- função de avaliação;
- operadores genéticos de reprodução;
- estratégia de seleção.

O algoritmo genético proposto resulta em um seqüencial de tarefas.

2.5.1 Codificação utilizada

Um indivíduo é representado por um vetor de inteiros $V=[v_1, \ldots, v_n]$ que descreve a seqüência de tarefas que forma um seqüencial sem levar em consideração as tarefas auxiliares FOL e FES. A componente v_i indica que a i-ésima tarefa do seqüencial é a v_i-ésima tarefa do conjunto *PDT*. Por exemplo, dado o conjunto PDT da Tabela 2.2, o indivíduo V=[1,4,3,2] representa o seqüencial da Tabela 2.3. Observe que de acordo com esta representação, um indivíduo pode representar vários seqüenciais. Por exemplo, o mesmo indivíduo apresentado no exemplo anterior também representa o seqüencial da Tabela 2.4.

Número	Código	Descrição	Hora de início	Horas noturnas	Horas diurnas
1	MA	Manobra de pátio	8:00	0	7
2	MA	Manobra de pátio	12:00	0	7
3	PRO	Prontidão	16:00	4	4
4	RET	Viagem e retorno	20:00	2	6

Tabela 2.2 – PDT (programação diária de tarefas)

DIA 1	DIA 2	DIA 3	DIA 4	DIA 5	DIA 6	DIA 7
MA 8:00	RET 20:00	FES	PRO 16:00	MA 12:00	FOL	

Tabela 2.3 – Um indivíduo representado pelo vetor V=[1,4,3,2]

DIA 1	DIA 2	DIA 3	DIA 4	DIA 5	DIA 6	DIA 7
FOL	MA 8:00	RET 20:00	FES	PRO 16:00	FES	MA 12:00

Tabela 2.4 – Outro indivíduo representado pelo vetor V=[1,4,3,2]

Para se estabelecer uma unicidade da correspondência entre um indivíduo e um seqüencial de tarefas, utiliza-se os seguintes critérios para construir um seqüencial:

- um seqüencial sempre se inicia com uma tarefa do conjunto PDT;
- a tarefa auxiliar que indica folga semanal (FOL) é inserida no seqüencial a cada intervalo de 6 ou 8 dias;
- entre o início de duas tarefas consecutivas em um seqüencial deve haver um intervalo de tempo suficiente para executar a primeira tarefa e descansar até o início da segunda. Este tempo é configurável e pode variar de acordo com os tipos das tarefas;
- a tarefa auxiliar que indica fora-de-escala (FES) somente é inserida entre duas tarefas quando a condição acima não é satisfeita. Em alguns casos o FES pode ser substituído por códigos que indicam que a tarefa anterior ainda se encontra em andamento no dia seguinte. Por exemplo, em algumas ferrovias a tarefa VIAG (viagem) sempre vem seguida do código REVI (retorno da viagem) ao invés do código FES.

2.5.2 Função de avaliação

Um indivíduo, quando representa um seqüencial válido, é avaliado pela equação (2.15) da seção anterior (página 32) com a ressalva de que a função *Tamanho*(S) passa a ser a representada pela equação (2.37) ao invés da equação (2.36).

$$Tamanho(S) = \begin{cases} \dfrac{tamanho}{TmMAX}, \text{ quando o objetivo é minimizar o tamanho} \\ \dfrac{tamanho - TmD}{TmMAX}, \text{ quando o obj. é um seq. de tamanho fixo} \end{cases} \quad (2.37)$$

onde: *TamD* é o tamanho desejado e *TmMAX* é uma constante utilizada para normalizar o resultado, ou seja, para que os valores estejam entre 0 e 1.

Algumas vezes um indivíduo pode não corresponder a um seqüencial válido. Neste caso a avaliação utilizada é determinada por:

$$f(S) = CTE + PF \quad (2.38)$$

onde *PF* é o número do passo onde houve falha na criação do seqüencial a partir de um indivíduo (vetor de inteiros). A constante *CTE* é um número grande o suficiente para ser maior do que qualquer avaliação de um indivíduo válido (quanto menor a avaliação, melhor o indivíduo). Por exemplo, CTE=10.

2.5.3 Operadores genéticos

Os seguintes operadores genéticos foram implementados:

- CX1 – cruzamento com o melhor indivíduo
- CX2 – cruzamento com um indivíduo escolhido aleatoriamente
- MU1 – mutação do melhor indivíduo
- MU2 – mutação de um indivíduo escolhido aleatoriamente

O cruzamento utilizado em CX1 e CX2 é o conhecido na literatura como OX ("*ordered crossover*"), bastante utilizado na literatura [Gol89][Lad96]. Este crossover foi preferido a outros como o PMX ("*partially-matched crossover*") por preservar a ordem relativa das tarefas no indivíduo. Observe que como o seqüencial é cíclico, a ordem absoluta das tarefas não é relevante.

A mutação utilizada foi a inversiva [Gol89][Lad96]. Neste operador, dois genes são aleatoriamente escolhidos na representação do indivíduo e trocados.

Para se selecionar indivíduos aleatoriamente em CX2 e MU2 foi utilizada a estratégia conhecida como "*roulette-wheel*", onde a probabilidade de um indivíduo ser escolhido é inversamente proporcional a sua avaliação, pois quanto menor a avaliação, melhor o indivíduo, e portanto maior deve ser a probabilidade de ser escolhido.

2.5.4 Estratégia de seleção

O GA modificado cria uma população final a partir de um conjunto de populações intermediárias. Para tal determina-se primeiro a união de todas as populações intermediárias. A seguir, o resultado da união é amostrado usando a estratégia de "*roulette-wheel*", mas preservando o indivíduo correspondente a melhor solução encontrada.

2.6 Atribuição

Uma vez criado o seqüencial de tarefas, o próximo passo consiste na etapa de *atribuição* (veja a Figura 2.1). Esta etapa é essencialmente a atribuição de funcionários para cada um dos passos do seqüencial de tarefas. Esta atribuição não é trivial uma vez que, devido ao trabalho realizado pelos funcionários no passado, nem todos os funcionários estarão aptos a executar todas as tarefas. Além disto, é necessário haver um equilíbrio do trabalho realizado pelos funcionários.

Das poucas referências encontradas na literatura utilizando a abordagem cíclica, a atribuição somente é mencionada em [Con97]. Isto ocorre porque normalmente assume-se que após um período de tempo suficientemente grande, todos os funcionários terão executado as mesmas tarefas, levando assim a uma distribuição igualitária da carga de trabalho. Porém nos casos reais isto não se verifica por três razões: primeiro porque o seqüencial freqüentemente muda antes que os funcionário possam executar um ciclo completo; segundo porque o que é planejado raramente é executado com precisão; terceiro porque na etapa de instanciação a escala sofre várias modificações (vide seção 2.7). Portanto, diferentemente do que é proposto em [Con97], é utilizado não só o passado planejado para cada funcionário mas também o efetivamente realizado.

Este problema pode ser facilmente formulado através de um modelo de "*set-covering*", de forma semelhante ao proposto em [Con97].

Seja:

- F o conjunto de m funcionários $\{f_1, \ldots, f_m\}$;
- f_i o i-ésimo funcionário, caracterizado pela tupla (*Hnf*, *Hdf*, *Pf*, *Df*) onde:
  - Hnf_i é o número de horas noturnas trabalhadas pelo funcionário no passado;
  - Hdf_i é o número de horas diurnas trabalhadas pelo funcionário no passado;
  - Pf_i é o próximo instante em que o funcionário estará pronto para pegar uma tarefa;
  - Df_i é o número de dias que o funcionário está sem folgar;
- E o conjunto $\{t_1, t_2, \ldots, t_n\}$, isto é, um seqüencial de n tarefas
- p_j é o j-ésimo passo do seqüencial E representada pela ênupla (*hs*, *Hdp*, *Hnp*, *Dfp*), onde:
  - hs_j é a hora de inicio da próxima tarefa do seqüencial que se inicia neste passo;
  - Hdp_j é o número de horas diurnas de uma escala que inicia no passo p_j;
  - Hnp_j é o número de horas noturnas de uma escala que inicia no passo p_j;
  - Dfp_j é o número de dias do passo p_j à próxima folga.
- G=(V,A) um grafo onde V é um conjunto de nós e A é um conjunto de arcos.

Hnf_i e Hdf_i devem ser contabilizados através do trabalho efetivamente realizado para um período passado de interesse. Normalmente adotam-se 2 ou 3 meses passados utilizando a escala planejada para cobrir do realizado até o início da escala. Por exemplo, se a escala for criada no dia 10 do mês para o mês seguinte, é então utilizada a programação do dia 11 ao dia 31 para cobrir a lacuna dos dados realizados ainda inexistentes.

Os nós do grafo são associados aos funcionários e aos passos, isto é, $V=F \cup E$. Para cada possível atribuição de um funcionário $f_i \in F$ a um passo do seqüencial $p_j \in E$ são criados um arco $a_{ij} \in A$ e uma variável de decisão x_{ij} e calculado um valor c_{ij}. Uma atribuição é possível quando o próximo instante em que o funcionário f_i pode iniciar uma atividade (Pf_i) é menor ou igual ao início da primeira atividade da escala gerada a partir do passo p_j. Além disso, o número de dias que o funcionário f_i está sem folgar (Df_i) somado com a distância que o passo atual está do passo de folga (Dfp_i) deve ser menor ou igual ao número máximo de dias consecutivos que um funcionário pode ficar sem folga.

Assim o modelo fica:

$$\min \sum_{ij} c_{ij} x_{ij} \tag{2.39}$$

As

$$\sum_{i} x_{ij} = 1 \tag{2.40}$$

$$\sum_{j} x_{ij} = 1 \tag{2.41}$$

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A \tag{2.42}$$

A restrição dada pela equação (2.40) assegura que cada funcionário será atribuído a somente um passo do seqüencial. Por outro lado a equação (2.41) garante que cada passo do seqüencial será alocado para somente um funcionário.

O valor c_{ij} associado a uma atribuição é determinado levando-se em consideração o passado do funcionário f_i. Assim, para cada passo p_i do seqüencial de tarefas E é contabilizado o total de horas noturnas Hnp_i e o total de horas diurnas Hdp_i da escala de d dias que se inicia em p_i. Também é contabilizado, para cada passo, a distância Dfp_i que este passo está da próxima folga.

$$c_{ij} = 1 - e^{-\left(\gamma_{1ij}^2 + \gamma_{2ij}^2\right)} \tag{2.43}$$

$$\gamma_{1ij} = \frac{\left(Hnp_j + Hnf_i\right) - Mn}{Mn} \tag{2.44}$$

$$\gamma_{2ij} = \frac{\left(Hdp_j + Hdf_i\right) - Md}{Md} \tag{2.45}$$

$$Mn = \frac{\sum_j Hnp_j}{|E|} + \frac{\sum_i Hnf_i}{|F|} \tag{2.46}$$

$$Md = \frac{\sum_j Hdp_j}{|E|} + \frac{\sum_i Hdf_i}{|F|} \tag{2.47}$$

onde:

- Mn e Md são as médias de horas noturnas e diurnas das escalas formadas pelo seqüencial de tarefas;
- γ_{1ij} e γ_{2ij} são os desvios com relação a média se o trabalhador i for atribuído ao passo j.

A idéia desta formulação é implementar a superfície de decisão mostrada na Figura 2.13. Nesta superfície, a melhor avaliação (menor valor c_{ij}) ocorre quando os desvios do trabalho noturno e diurno são ambos zero. A medida em que os valores absolutos destes desvios vão aumentando, a avaliação vai piorando, sempre de forma simétrica.

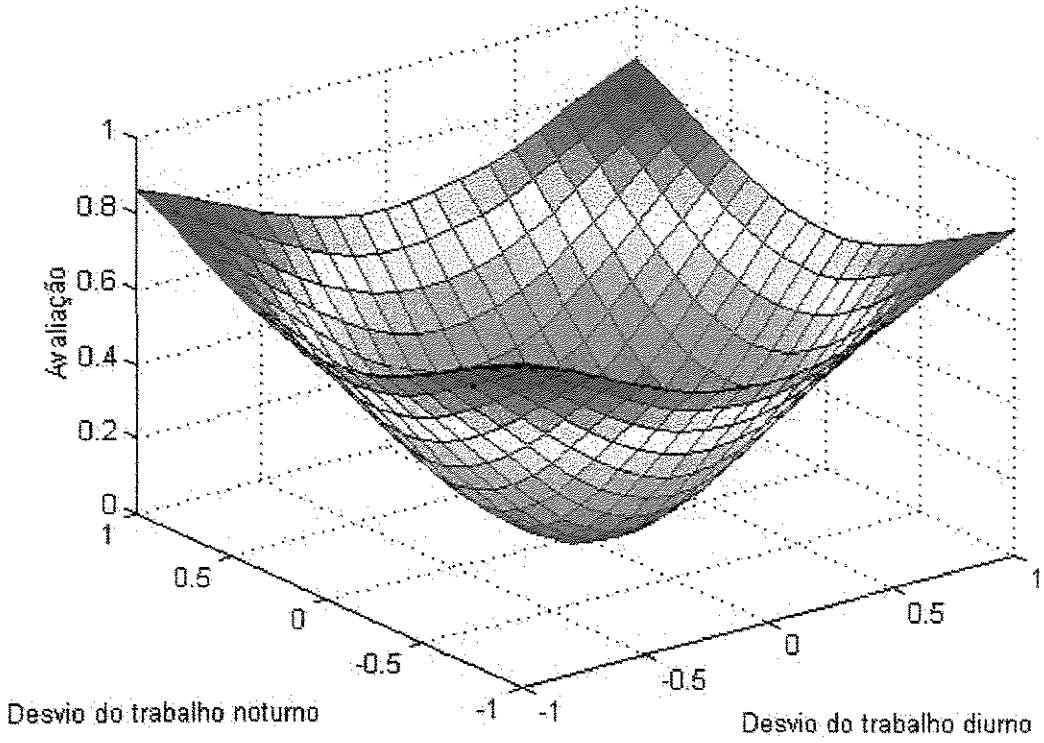

Figura 2.13 – Superfície de decisão para calcular a avaliação c_{ij}

2.7 Instanciação

Depois de criado o seqüencial de tarefas e feita as atribuições dos passos aos funcionários, dá-se início à etapa de instanciação. Esta etapa consiste em gerar as escalas para todos os funcionários e ajustá-la para atender algumas restrições operacionais. Normalmente esta etapa não é referenciada na literatura. Apesar disto ela também tem um papel extremamente importante pois, na maioria das vezes, demanda um tempo considerável do pessoal responsável pela geração de escalas. Questões práticas são normalmente resolvidas nesta etapa e incluem:

- atender pedidos de liberação de funcionários para treinamento;
- atender pedidos de férias;
- atender pedidos de troca de passo (mudança da posição no seqüencial);
- atender outras solicitações de funcionários (folga extra, etc...);
- cobrir funcionários que ficam indisponibilizados (doença, acidente de trabalho, etc).

No método de seqüencial de tarefas abordado neste capítulo é difícil considerar estas questões práticas. Normalmente gera-se o seqüencial de tarefas desconsiderando propositalmente alguns funcionários, que serão utilizados nos ajustes posteriores. Observe que esta solução pode comprometer os ganhos conseguidos na etapa de atribuição tais como a distribuição adequada da carga de trabalho levando-se em consideração o passado de cada funcionário.

2.8 Resumo

Neste capítulo foi apresentado um dos paradigmas utilizados na programação de equipagens chamado de "método do seqüencial de tarefas". As principais abordagens existentes na literatura foram expostas e duas novas abordagens propostas. A primeira delas é baseada em algoritmos de busca, mais especificamente no algoritmo "*branch-and-bound*". A segunda abordagem é baseada em algoritmos genéticos.

Observe que o algoritmo de busca proposto, implementa o princípio da otimalidade de Belman e utiliza avaliações com estimativas de limites inferiores. Portanto ele é um algoritmo do tipo A* [Lug98].

Os algoritmos propostos para as etapas de seqüenciamento e escalonamento têm as seguintes vantagens com relação aos outros encontrados na literatura:

- são explicitamente multicritério (2.15), podendo o usuário ponderar a importância de cada critério (constantes de combinação linear k,l,o,p). A maioria dos algoritmos encontrados na literatura, e.g. Caprara et. al., se preocupam em minimizar custos. Outros, como Constantino, utilizam outros critérios porém não permitem ao usuário ponderar sua importância.
- podem ser facilmente estendidos para utilizar outros objetivos além daqueles aqui considerados;
- podem ser utilizados para gerar seqüências de tamanho mínimo ou de tamanho pré-determinado;
- podem levar em consideração objetivos locais e globais na avaliação de uma solução sem aumentar a complexidade dos algoritmos. Os objetivos locais são aqueles que resultam da análise de cada atribuição em separado, e os objetivos globais são aqueles que resultam da análise do seqüencial como um todo.

O algoritmo aqui proposto para a atribuição somente encontra um similar em [Con97] onde o autor propõe uma abordagem baseada no fato de que o passo de cada funcionário é conhecido. Isto porém é de pouca utilidade prática uma vez que despreza a etapa de instanciação onde a escala final pode sofrer várias alterações e ajustes. Além disto, os funcionários que não estão presentes no seqüencial do mês anterior (sobra) não são levados em consideração. Além de resolver estas questões, o modelo aqui proposto leva em consideração a trabalho efetivamente realizado em adição ao planejado para o período anterior.

Os resultados obtidos com os algoritmos propostos neste capítulo serão discutidos no capítulo 5.

3. Método direto

3.1 Introdução

O paradigma de geração de escalas apresentado no capítulo anterior é o utilizado atualmente em todas as ferrovias brasileiras que emitem escalas mensais de trabalho e, até onde se sabe, não se encontra alternativa a este paradigma na literatura para este ramo de atividade. Conforme mencionado no capítulo 1, em alguns casos, onde se aplica o conceito de turnos de trabalho, existem publicações que descrevem métodos de criação de escalas individualizadas, sem a utilização de seqüenciais de tarefas [War76][Bur85][Hol76]. Todavia, estes métodos não se aplicam à geração de escalas de equipagens ferroviárias.

Apesar de ser o mais utilizado, o método do seqüencial de tarefas apresenta um grande problema de ordem prática: a instanciação da escala (vide seção 2.7) é complicada. Observe que, apesar da instanciação da escala ser praticamente ignorada na literatura, ela é de grande importância prática pois é necessário cada vez mais atender as necessidades pessoais e sociais dos funcionários (e da empresa) e ganhar em produtividade. Alguns exemplos que ilustram estas necessidades:

- um determinado funcionário sabe, com seis meses de antecedência, que no dia 13/06/2000 terá um compromisso social. Neste caso ele solicita uma folga ao centro de escalas neste dia ou que o coloque numa tarefa (por exemplo, numa manobra) que inicie bem cedo e que termine (sem correr o risco de atrasar) antes do compromisso;
- o sindicato da classe, para abrir mão de algum benefício, exige como compensação que a empresa dê folga para seus funcionários no dia de seus aniversários;
- através das avaliações realizadas constantemente na ferrovia, detecta-se que um determinado funcionário não opera o trem adequadamente em um determinado trecho, causando riscos e gastos desnecessários, além de um consumo elevado de combustível. Neste caso o centro de escalas recebe um pedido de um período de treinamento para o funcionário em questão;
- os funcionários de um determinado destacamento solicitam períodos de férias menores do que 30 dias.

Conforme exposto na seção 2.7, a solução normalmente encontrada para acomodar as necessidades dos equipagens (e da empresa) é gerar um seqüencial de tarefas sem a participação de algumas equipagens (geralmente uma ou duas)[10]. Desta forma resolve-se parte dos problemas através de trocas de tarefas. Por exemplo, para resolver o primeiro problema exemplificado anteriormente, bastaria passar a tarefa alocada para o equipagem no dia 13/06/2000 para a equipagem que foi deixada de fora da escala. Esta solução tem os seguintes problemas:

[10] No jargão utilizado nas ferrovias, estas equipagens ficam "na sobra".

- o fato de haver equipagens disponíveis para realizar trocas de tarefas não garante que todas as trocas possam ser realizadas;
- apesar do método do seqüencial de tarefas buscar uma equalização na distribuição do trabalho durante a atribuição (vide seção 2.6), o resultado obtido é prejudicado pelos ajustes realizados na instanciação da escala. Além disto, nada garante a qualidade da escala de trabalho das equipagens não consideradas na etapa de atribuição;
- o trabalho realizado na instanciação é complexo e dispendioso, tomando grande parte do tempo necessário para a confecção da escala mensal de trabalho.

Para tentar resolver estes problemas, foi desenvolvido um método de alocação de escalas que não utiliza o conceito de seqüencial de tarefas, ou seja, um novo paradigma de alocação de escalas. Este método foi chamado de *método direto*. Não foi utilizada a expressão "alocação individualizada" porque ela é comum quando uma escala é gerada sem o uso do seqüencial de tarefas, mas utilizando-se o conceito de turnos de trabalho [Con97].

A seguir serão apresentados alguns conceitos básicos que serão utilizados no decorrer do capítulo. Posteriormente, é apresentado um algoritmo genérico para o método direto e um exemplo real utilizado como teste do algoritmo. A seguir são apresentadas várias instâncias do algoritmo genérico, incluindo o detalhamento de alguns passos. Por último apresentam-se uma análise e comparação dos algoritmos propostos.

3.2 Conceitos básicos

Além dos conceitos já descritos no capítulo anterior na seção 2.2, são necessários ainda os seguintes: extra-tarefa, pré-alocação, equipe, força de trabalho, disponibilidade diária de força de trabalho e unimodularidade.

Extra-tarefa é uma tarefa que, ao contrário das tarefas fixas e não fixas, não tem um trabalho associado. É um termo utilizado para designar folgas, licenças remuneradas, etc... Uma extra-tarefa tem normalmente a duração de 24 horas e não é considerada como trabalho realizado apesar de ser normalmente remunerada.

Uma pré-alocação é uma atribuição de uma tarefa (ou extra-tarefa) na escala de uma equipagem em um determinado dia. É uma "alocação *a priori*".

Uma equipe é um conjunto de equipagens que trabalham juntas. Em algumas ferrovias todas as tarefas são executadas por duas equipagens (um maquinista e um auxiliar) devido a acordos. Em outras companhias, o conceito de equipe somente existe quando há um funcionário que necessita ser treinado, formando assim equipes de instrutores e alunos. Uma equipe aqui é portanto um conjunto de um ou dois funcionários. Quando uma equipe tem somente um funcionário é chamada de equipe unitária. Quando ela tem dois funcionários é chamada de equipe dupla.

Força de trabalho é o conjunto de funcionários aptos a trabalhar em um determinado momento. A disponibilidade diária de força de trabalho pode ser representada por um gráfico que mostre a aridade da força de trabalho em cada um dos dias de um determinado período.

Uma matriz é dita unimodular se ela possui somente elementos inteiros e tem determinante igual a ±1. As matrizes unimodulares também são chamadas de matrizes unitárias. A inversa de uma matriz unimodular também é unimodular. Uma matriz unimodular positiva tem determinante igual a +1.

3.3 Algoritmo genérico

O método direto é caracterizado por uma coleção de algoritmos que se baseiam no algoritmo genérico resumido na Figura 3.1. Este algoritmo é baseado em abordagens heurísticas desenvolvidas para problemas de escalonamento em sistemas de manufaturas (*multi-machine scheduling systems*) [Tho93]. A idéia que está por trás deste algoritmo é que uma escala de N dias pode ser vista como a concatenação de N escalas de um dia.

O algoritmo genérico possui dois laços principais: o maior (a) é executado uma vez para cada dia do horizonte de planejamento da escala; o menor (b) pode existir ou não, dependendo do algoritmo implementado. Quando ele existe, ele é executado uma vez para cada tarefa pertencente a uma lista criada na etapa "criar e ordenar lista de tarefas". O funcionamento deste algoritmo ficará claro quando forem apresentadas as suas especializações.

A diferenciação entre os algoritmos está na forma em que a avaliação é executada. Foram estudadas 4 formas diferentes de se realizar esta etapa, sendo que, para cada uma delas, há outras 4 variações que serão vistas mais adiante. Portanto, a rigor, foram estudados 16 variações do algoritmo genérico de alocação dentro deste novo paradigma.

A seguir descreve-se, de forma conceitual, as etapas. Posteriormente as suas variações serão detalhadas.

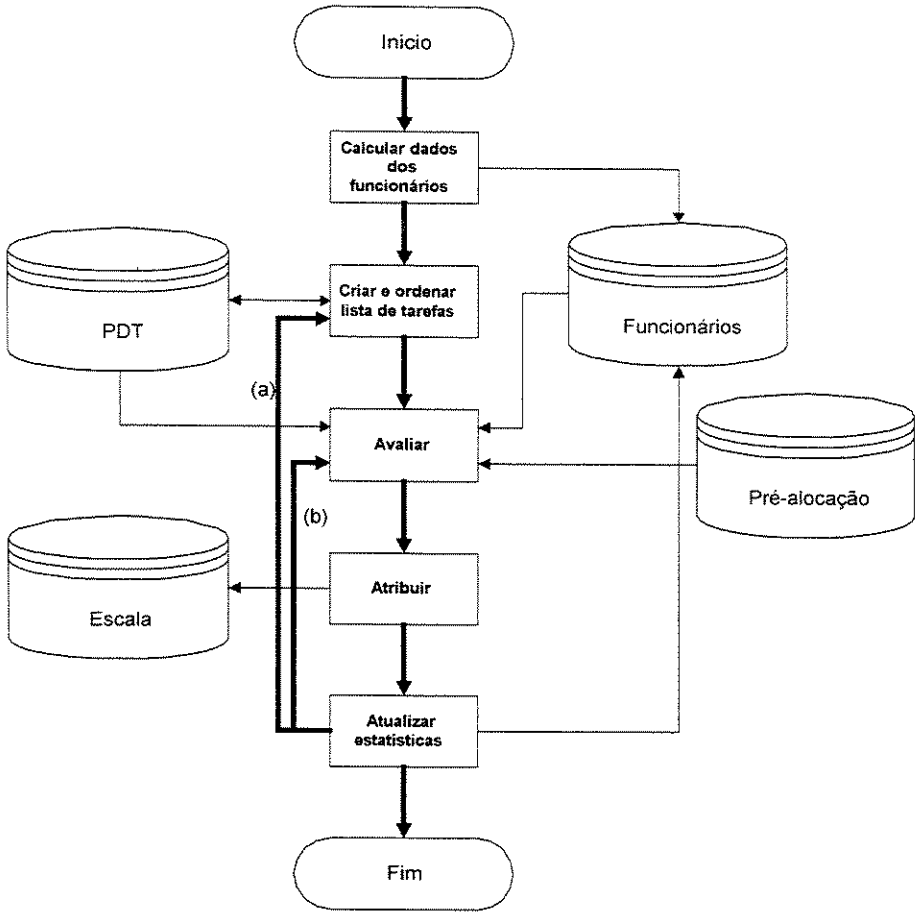

Figura 3.1 – Algoritmo genérico do método direto

3.3.1 Calcular dados dos funcionários

A primeira etapa do algoritmo tem por finalidade coletar e organizar informações sobre os funcionários que participarão da escala. As seguintes informações são levantadas para cada um dos funcionários:

- data da última folga;
- carga de trabalho no passado (usualmente nos últimos 2 ou 3 meses);
- próxima disponibilidade para trabalhar de acordo com o passado (última escala planejada);
- pré-alocações;
- formação de equipes e relacionamento instrutor-aluno;
- tipo da última tarefa executada (ou planejada);
- feriados contemplados com folga nos últimos cinco anos;
- data de aniversário.

Alguns dos dados não são diretamente utilizados no algoritmo. Por exemplo, os feriados contemplados com folga nos últimos cinco anos servem somente para o operador (usuário) planejar

melhor as pré-alocações que serão utilizadas. Normalmente é inviável atender todas as pré-alocações, necessitando por isso um processo de seleção. Por exemplo, normalmente todos os funcionários desejam folgas no Natal. Pré-alocar folga no Natal para todos os funcionários é inviável. Sendo assim, necessita-se selecionar as pré-alocações, priorizando (por exemplo) aqueles que não folgaram no Natal nos últimos cinco anos.

Algum cuidado deve ser tomado com a formação de equipes duplas. Quando uma equipe dupla é formada, normalmente os funcionários que a compõe têm escalas diferentes, folgas em datas diferentes e com tarefas diferentes no dia imediatamente anterior ao início da escala que está sendo planejada. Assim, para que possam ser "sincronizados", sem que saiam prejudicados, a data da última folga da equipe deve ser considerada como sendo a mais antiga entre as datas dos funcionários. Além disto, a próxima disponibilidade para trabalhar deve ser a mais recente. Formalmente, seja:

$Pdisp_i$ o instante da primeira disponibilidade do funcionário i

UF_i a data da última folga do funcionário i

Então, para a equipe $Eqp = \{f_1, f_2\}$:

$$Pdisp_E = \max(Pdisp_1, Pdisp_2)$$

$$UF_E = \min(UF_1, UF_2)$$

Como para o método seqüencial, a carga de trabalho será medida em horas noturnas e horas diurnas.

3.3.2 Criar e ordenar lista de tarefas

A segunda etapa do algoritmo é a primeira de um laço que se repete uma vez para cada dia da escala. Nesta etapa do algoritmo é criada e ordenada uma lista de tarefas a serem cumpridas em um determinado dia a partir da programação diária de tarefas (PDT). Esta etapa é necessária porque, diferentemente do método do seqüencial de tarefas, pode haver tarefas sazonais na PDT. Por exemplo, pode haver uma manobra extra somente aos sábados e/ou uma viagem a menos aos domingos. Observe que é difícil considerar esta situação no método de seqüencial de tarefas uma vez que por definição, todas as tarefas do seqüencial serão executadas todos os dias da semana.

3.3.3 Avaliar

A avaliação é uma etapa onde são verificadas as possibilidades de atribuição de tarefas para funcionários (para equipes, sendo mais preciso). Dependendo do método utilizado, esta etapa pode ser executada somente uma vez ou ser o início de um laço que é executado uma vez para cada tarefa da lista criada na etapa anterior. Esta etapa é a mais importante de todas elas e será descrita com detalhes mais adiante.

3.3.4 Atribuir

Da mesma forma que a etapa anterior, a etapa de atribuição pode ser executada somente uma vez ou quantas vezes forem o número de tarefas da lista produzida na etapa "criar e ordenar lista de tarefas", dependendo do método. Nesta etapa são efetivamente realizadas as atribuições examinadas na etapa anterior.

3.3.5 Atualização de estatísticas

Após cada atribuição realizada, a carga de trabalho dos funcionários é recalculada. Esta etapa poderia ser considerada como parte integrante da etapa anterior mas, devido a sua importância, ela foi separada.

3.4 Um exemplo real

Apresenta-se a seguir uma descrição detalhada dos dados e requisitos que caracterizam uma escala. Os dados e requisitos proporcionam um exemplo prático típico, pois espelham situações presentes nos casos reais. Além de servir como ilustração e testes dos algoritmos propostos, o exemplo também fornece um "*benchmark*" pois na literatura aberta não se encontra dados e requisitos para testes comparativos.

Suponha um destacamento D, com um conjunto $F=\{f_1, f_2, \ldots, f_{48}\}$ de 48 funcionários com uma programação diária de tarefas $PDT= \{t_1, \ldots, t_{11}\}$com 11 tarefas distribuídas conforme a Tabela 3.1.

			Tempo em horas das atividades que compõe a tarefa						
Nome	Descr.	Horário	Prontidão	Passe	Trabalho	Descanso fora da sede	Prontidão	Passe	Trabalho
RET	Viagem	02:00	2	2	8				
RET	Viagem	04:00	2	2	8				
RET	Viagem	06:00	2	2	8				
MA1	Manobra de pátio	07:00			8				
PRO	Viagem	08:00	2	2	8	10	2	2	8
RET	Viagem	11:00	2	2	8				
RET	Viagem	14:00	2	2	8				
RET	Viagem	17:00	2	2	8				
MA1	Manobra de pátio	18:00			8				
RET	Viagem	20:00	2	2	8				
RET	Viagem	23:00	2	2	8				

Tabela 3.1 - Programação diária de tarefas

Na tabela de programação diária PDT, todas as tarefas devem ser realizadas todos os dias por duas pessoas. Observe que, apesar de não ser o caso neste exemplo, pode haver tarefas para ser

executadas somente em dias específicos da semana. Este é o caso, por exemplo, do trem de passageiros, em algumas ferrovias brasileiras.

Suponha também uma escala a ser programada para o período de 01 a 29/fevereiro/2000 e um conjunto $P=\{ p_1, \ldots, p_n \}$ com n pré-alocações distribuídas conforme a Tabela 3.2.

#	Func.	Tarefa	Descrição	Hora início	Data início	Data fim
1	f_1	AAT	Afastado por acidente de trabalho	-	01/02/2000	29/02/2000
2	f_2	DEM	Demitido	-	01/02/2000	29/02/2000
3	f_8	FER	Férias	-	01/03/2000	31/03/2000
4	f_{13}	FER	Férias	-	01/02/2000	01/03/2000
5	f_{25}	FER	Férias	-	01/02/2000	01/03/2000
6	f_{11}	TRN	Treinamento no simulador	08:00	02/02/2000	04/02/2000
7	f_{11}	FOL	Folga	-	05/02/2000	06/02/2000
8	f_{11}	TRN	Treinamento no simulador	08:00	08/02/2000	11/02/2000
9	f_{11}	FOL	Folga	-	12/02/2000	
10	f_{11}	RET	Viagem	23:00	13/02/2000	
11	f_{16}	AFD	Afastado por doença	-	01/02/2000	
12	f_{17}	EXM	Exame médico	08:00	10/02/2000	
13	f_{17}	MAN	Manobra	07:00	09/02/2000	
14	f_{20}	FER	Férias	-	16/02/2000	16/03/2000
15	f_{21}	ANI	Folga de aniversário	-	17/02/2000	
16	f_{23}	FER	Férias	-	01/02/2000	27/02/2000
17	f_{24}	MAN	Manobra	07:00	23/02/2000	
18	f_{24}	EXM	Exame médico	08:00	24/02/2000	
19	f_{35}	MAN	Manobra	07:00	16/02/2000	
20	f_{35}	EXM	Exame médico	-	17/02/2000	
21	f_{43}	MAN	Manobra	07:00	16/02/2000	
22	f_{43}	EXM	Exame médico	-	17/02/2000	
23	f_{44}	FER	Férias	-	01/02/2000	20/02/2000
24	f_{48}	FER	Férias	-	01/03/2000	01/04/2000

Tabela 3.2 – Pré-alocações de tarefas e extra-tarefas

Além disto, é evidente que a escala gerada deve respeitar as restrições impostas pela escala do mês anterior, janeiro de 2000. Estes dados estão na Tabela 3.3. A escala também deve procurar equalizar a carga de trabalho dos funcionários levando em consideração o trabalho realizado no período de dezembro de 1999 e janeiro de 2000. Estes são fornecidos pela Tabela 3.4.

Funcionário	Data da última folga	Término da última tarefa
f_1	31/01/2000	00:00 01/02/2000
f_2	25/12/1999	16:00 26/01/2000
f_3	28/01/2000	17:00 31/01/2000
f_4	29/01/2000	12:00 31/01/2000
f_5	31/12/1999	00:00 31/01/2000
f_6	27/01/2000	02:00 01/01/2000
f_7	25/12/1999	00:00 31/01/2000
f_8	26/01/2000	09:00 01/02/2000
f_9	29/01/2000	12:00 31/01/2000
f_{10}	28/01/2000	22:00 31/01/2000
f_{11}	26/01/2000	23:00 31/01/2000
f_{12}	26/01/2000	23:00 31/01/2000
f_{13}	26/01/2000	06:00 01/02/2000
f_{14}	31/01/2000	22:00 01/02/2000
f_{15}	31/01/2000	22:00 01/02/2000
f_{16}	25/01/2000	06:00 31/01/2000
f_{17}	27/01/2000	23:00 01/02/2000
f_{18}	31/01/2000	08:00 01/02/2000
f_{19}	30/01/2000	12:00 01/02/2000
f_{20}	26/01/2000	06:00 01/02/2000
f_{21}	30/01/2000	22:00 31/01/2000
f_{22}	30/01/2000	15:00 31/01/2000
f_{23}	30/01/2000	12:00 01/02/2000
f_{24}	31/01/2000	22:00 01/02/2000

Funcionário	Data da última folga	Término da última tarefa
f_{25}	28/01/2000	15:00 31/01/2000
f_{26}	30/01/2000	26:00 31/01/2000
f_{27}	28/01/2000	22:00 31/01/2000
f_{28}	28/01/2000	17:00 31/01/2000
f_{29}	27/01/2000	23:00 01/02/2000
f_{30}	29/01/2000	05:00 31/01/2000
f_{31}	30/01/2000	15:00 31/01/2000
f_{32}	29/01/2000	05:00 31/01/2000
f_{33}	28/01/2000	15:00 31/01/2000
f_{34}	26/01/2000	09:00 01/02/2000
f_{35}	13/01/2000	06:00 31/01/2000
f_{36}	30/01/2000	03:00 01/02/2000
f_{37}	31/01/2000	04:00 01/02/2000
f_{38}	28/01/2000	19:00 31/01/2000
f_{39}	29/01/2000	15:00 31/01/2000
f_{40}	29/01/2000	15:00 31/01/2000
f_{41}	27/01/2000	00:00 01/02/2000
f_{42}	29/01/2000	03:00 31/01/2000
f_{43}	31/01/2000	04:00 01/02/2000
f_{44}	29/01/2000	03:00 31/01/2000
f_{45}	28/01/2000	19:00 31/01/2000
f_{46}	31/01/2000	08:00 01/02/2000
f_{47}	27/01/2000	00:00 01/02/2000
f_{48}	27/01/2000	02:00 01/02/2000

Tabela 3.3 - Dados do passado dos funcionários

Funcionário	Horas noturnas trabalhadas	Horas diurnas trabalhadas
f_1	0	0
f_2	0	0
f_3	32,4	78,18
f_4	38,64	155,58
f_5	44,58	153,13
f_6	45,7	73,45
f_7	51,3	136,15
f_8	51,45	121,31
f_9	57,53	189,83
f_{10}	59,67	204,47
f_{11}	62,0	202,80
f_{12}	62,5	198,7
f_{13}	63,12	238,32
f_{14}	66,3	171,32
f_{15}	66,4	122,24
f_{16}	67,19	179,76
f_{17}	68,65	208,85
f_{18}	68,75	9,45
f_{19}	68,8	184,86
f_{20}	68,88	183,9
f_{21}	69,45	142,31
f_{22}	70,45	182,68
f_{23}	70,5	185,31
f_{24}	71,35	172,72

Funcionário	Horas noturnas trabalhadas	Horas diurnas trabalhadas
f_{25}	71,4	178,47
f_{26}	72,56	219,03
f_{27}	72,9	215,56
f_{28}	75,4	194,5
f_{29}	76,2	214,5
f_{30}	76,51	203,31
f_{31}	77,54	144,09
f_{32}	77,56	207,84
f_{33}	78,6	202,22
f_{34}	79,7	175,49
f_{35}	81,31	249,88
f_{36}	84,4	211,6
f_{37}	84,53	219,31
f_{38}	84,76	211,92
f_{39}	85,53	183,54
f_{40}	85,53	212,84
f_{41}	85,65	201,78
f_{42}	86,94	222,36
f_{43}	88,63	194,06
f_{44}	88,79	226,36
f_{45}	89,75	203,47
f_{46}	90,85	176,72
f_{47}	91,6	216,188
f_{48}	96,64	223,84

Tabela 3.4 – Trabalho realizado no período de dezembro de 1999 a janeiro de 2000

3.5 Algoritmo A1

O primeiro algoritmo (A1) baseia-se no conceito de busca em profundidade informada [Lug98][Rus95]. O algoritmo A1 é uma instanciação do algoritmo genérico caracterizado por incluir o laço (b) – vide Figura 3.1. A seguir discute-se um exemplo de aplicação do algoritmo através de uma seqüência de passos que utiliza os dados apresentados na seção anterior. Posteriormente analisa-se o algoritmo, terminando com a sua descrição formal via pseudocódigo.

3.5.1 Um exemplo de aplicação do algoritmo A1

Passo 1 e 2. No primeiro e no segundo passo do algoritmo são realizadas as inicializações. Nestas etapas todos os dados necessários para a alocação são definidos, conforme seção 3.4.

Passo 3. No terceiro passo, é criada uma lista T_i das tarefas a serem realizadas no dia i (no primeiro dia i=1) a partir da programação diária de tarefas T. No exemplo em questão, esta lista é uma cópia de T pois não há tarefas sazonais. Esta lista é então ordenada pelo instante de tempo inicial de cada tarefa.

Ainda no passo 3 as pré-alocações do conjunto P (vide Tabela 3.2) são analisadas. Todas as pré-alocações $pr_j \in P$, programadas para o dia atual (i) são instanciadas, isto é, efetivamente alocadas. Por exemplo, para i = 1, é alocada para o funcionário f_1 a tarefa AAT. Observe que algumas pré-alocações não têm horários definidos. Quando isto ocorre, o próprio algoritmo deve decidir quais são os horários em que elas devem iniciar. Isto é feito analisando-se a última tarefa alocada e alguns parâmetros que definem os intervalos desejados entre as tarefas. Este detalhe será abordado com mais atenção na seção 3.10. Quando uma pré-alocação envolve uma tarefa da programação diária, ela é retirada da lista T_i.

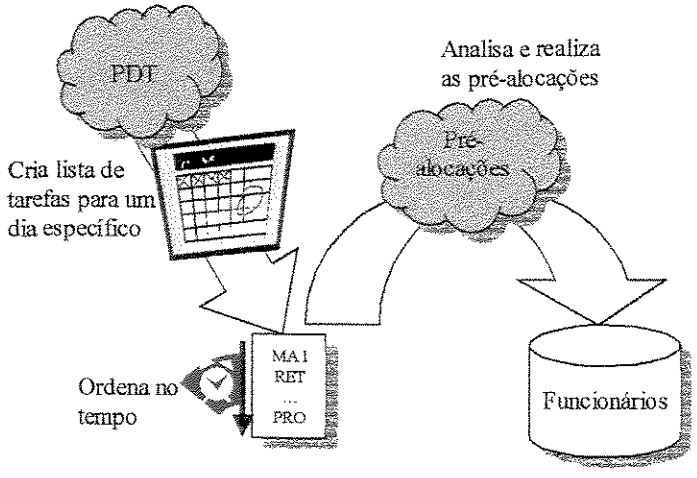

Figura 3.2 - Passo 3 do algoritmo A1

Passo 4. No quarto passo inicia-se um laço em que o índice j vai de 1 até o número de tarefas em T_i. Os passos 5 e 6 são executados dentro deste laço.

Neste passo a j-ésima tarefa de T_i, isto é, t_j é retirada da lista. É então criada uma lista A_j a partir do conjunto F contendo os funcionários (equipes) que estão aptos a realizarem a tarefa t_j no dia i. Um elemento a_{jk} de A_j é uma tupla ($t_j \in T_i$, $f \in F$, av) que indica uma possível atribuição da tarefa t_j ao funcionário f. O atributo av define uma avaliação associada ao elemento a_{jk}. Para simplificar a notação iremos referenciar ao funcionário f contido em uma tupla a_{jk} como sendo f_k.

Devido a sua complexidade e sua relevância, o algoritmo utilizado na criação da lista A_j (denominado filtro) será detalhado na seção 3.9. No momento o importante é o fato de que uma lista A_j é criada a partir de F e t_j e que esta lista contém todas as atribuições possíveis da tarefa $t_j \in T_i$ no dia i.

Ainda no quarto passo, a avaliação (av) é calculada para cada elemento a_{jk} da lista A_j. Esta avaliação é uma medida da qualidade da atribuição da tarefa t_j para f_k. Esta medida leva em consideração o passado, tanto de f_k, quanto do destacamento D como um todo. Novamente o detalhamento desta avaliação será postergada para uma seção posterior, devido a sua importância e complexidade (seção 3.11). Por último a lista A_j é ordenada em ordem crescente pela avaliação (quanto menor o valor, melhor).

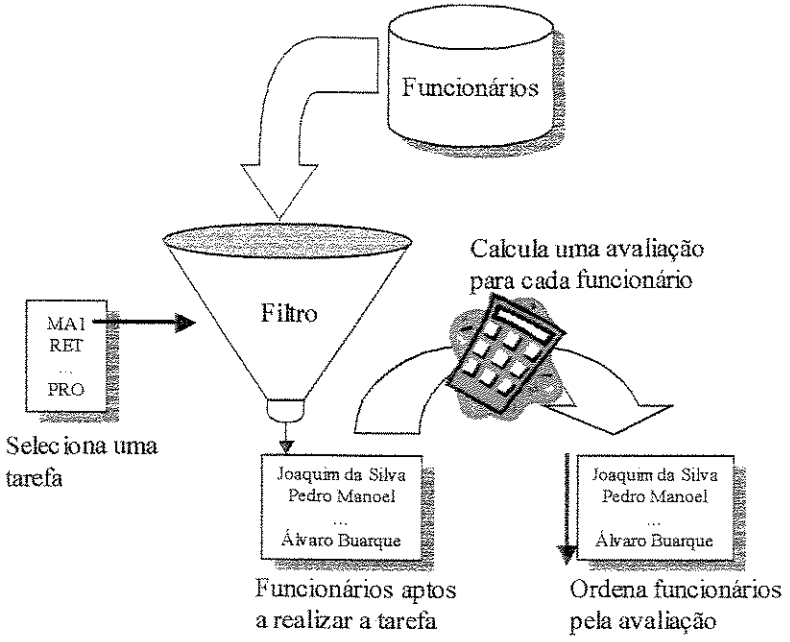

Figura 3.3 - Passo 4 do algoritmo A1

Passo 5. No quinto passo, a primeira atribuição da lista A_{jk} é realizada, ou seja, a tarefa retirada da lista T_i no passo 4 é atribuída a um funcionário $f_k \in F$.

Passo 6. No sexto e último passo, as estatísticas do funcionário para qual a tarefa foi atribuída no passo 5 são atualizadas. Também são atualizadas as estatísticas referentes ao destacamento como um todo. Estas estatísticas serão abordadas na seção 3.11.

A seguir, o algoritmo retorna para o passo 4, considerando a próxima tarefa da lista T_i até que esta lista fique vazia. Quando isto ocorrer, incrementa-se o valor de i (o dia corrente) e retorna-se para o passo 3 até que i seja o último dia do período de interesse para a escala.

Se o algoritmo não falhar, o resultado produzido é uma escala que atende todos os requisitos dados pela PDT e pelas pré-alocações. Este algoritmo falha quando, para uma tarefa escolhida no passo 4, a seleção de funcionários compatíveis (filtro) retorna uma lista vazia. Neste caso é possível realizar um "*backtrack*", desfazendo-se as atribuições realizadas no dia corrente e no dia anterior, decrementando o contador i (dia corrente da escala) e voltando para o passo 3, mas desta vez modificando o critério de seleção de funcionários do passo 5. Por exemplo, pode-se ao invés de selecionar o primeiro funcionário da lista, selecionar o segundo.

3.5.2 Análise e definição formal

Se toda escala factível for uma folha de uma árvore de busca e se as escalas incompletas, geradas durante a execução do algoritmo a cada vez que uma tarefa é atribuída a um funcionário, forem nós não terminais desta mesma árvore, então o algoritmo em questão realiza uma busca em profundidade informada que pára quando uma solução é encontrada. Um nó terminal é encontrado toda vez que a lista de funcionários criada no passo 4 está vazia. Neste caso um "*backtrack*" é realizado e um outro caminho é escolhido mudando-se o critério de atribuição do passo 5 no dia anterior àquele em que o nó terminal foi encontrado. Na realidade, neste caso, diferentemente do "*backtrack*" tradicional, a busca não retorna para o nó imediatamente anterior. Isto ocorre porque a atribuição realizada no nó imediatamente superior normalmente tem pouco efeito sobre a atribuição do nó corrente. Este efeito somente tem alguma relevância nos nós do dia anterior. Portanto o "*backtrack*" retorna $2n-j$ nós onde n é o número de tarefas da PDT e j é o número de tarefas restantes no conjunto T_i quando ocorre o "*backtrack*".

Por questão de simplicidade não foi incluído explicitamente no algoritmo o controle do "*backtracking*". Evidentemente é necessário controlar quantas vezes ocorre o "*backtrack*" em determinado ponto. O algoritmo implementado utiliza a seguinte política: o "*backtrack*" retorna n_b dias para trás do ponto onde ocorreu a falha, onde n_b é o número de vezes que ocorreu o "*backtrack*" naquele mesmo ponto. Após N_b tentativas (N_b normalmente igual 4 ou 5), desistir e terminar o procedimento indicando ausência de uma solução.

```
1)    Procedure A1 (T,F)
2)    BEGIN
3)       /* Passo 1 e 2*/
4)       Carregar todos os dados relevantes com relação aos funcionários contidos em F
5)       Calcular as horas noturnas e diurnas de cada funcionário de F
6)       FOR (i=1; i<=numero de dias da escala; i++) /*Para todos os dias*/
7)          /* Passo 3 */
8)          backtrack = false;
9)          Ti = lista_de_tarefas(T,i)
10)         Ti = aloca_tarefas_pré-alocadas(Ti,F,i)
11)         x = tamanho de Ti
12)         /* Passo 4 */
13)         FOR (j=1; j<= numero de tarefas em Ti; j++)
14)            tj = Ti[j];
15)            Aj = filtro(tj,F,i)
16)            IF (Aj=∅)
17)               backtrack = true;
18)               break; //sai do for mais interno...
19)            ELSE
20)               FOR (k=1; k<= tamanho de Aj; k++)
21)                  ajk = Aj[k];
22)                  ajk.av = avaliação(Aj, t)
23)               ENDFOR
24)               sort(Aj)
25)               f = Aj[1]
26)               /* Passo 5 */
27)               Atribui(f,t)
28)               /* Passo 6 */
29)               recalcula_estatisticas()
30)            ENDIF
31)         ENDFOR
32)         IF (backtrack = true)
33)            desfazalocacoes(F,x,j)
34)            i--
35)         ENDIF
36)      ENDFOR;
37)   END;
38)
39)   Procedure lista_de_tarefas(T,i)
40)      retorna a lista das tarefas que deverão ser realizadas no dia i conforme a lista de tarefas T
41)   Procedure aloca_tarefas_pré-alocadas (Ti,F,i)
42)      aloca as tarefas que estão pré-alocadas para o funcionário F. Retira as tarefas já pré-alocadas de Ti.
43)      retorna Ti modificado
44)   Procedure desfazalocacoes(F,x,j)
45)      desfaz as últimas 2x-j alocações
```

Algoritmo 3.1 - Algoritmo A1

3.6 Algoritmo A2

O segundo algoritmo (A2) baseia-se no conceito de busca em largura [Lug98][Rus95]. O algoritmo A2 é uma instanciação do algoritmo genérico onde o laço (b) da Figura 3.1 não é utilizado. Como no caso anterior, este algoritmo será apresentado via um exemplo de utilização. Posteriormente será realizada uma análise deste algoritmo e por último ele será apresentado na forma de pseudocódigo.

3.6.1 Um exemplo de aplicação do algoritmo A2

Passo 1, 2 e 3. Idênticos aos apresentados para o algoritmo A1 (seja seção 3.5.1).

Passo 4. No quarto passo, todas as tarefas $t_j \in T_i$ são utilizadas para criar listas A_j contendo as possíveis atribuições para as tarefas $t_j \in Ti$, da mesma forma como foi realizado no algoritmo A1.

Ainda no quarto passo, as avaliações dos elementos $a_{jk} \in A_j$ são calculadas. Esta avaliação é a mesma utilizada no algoritmo A1 e será detalhada na seção 3.11. Uma vez criadas, cada uma das listas A_j é ordenada utilizando-se as avaliações calculadas.

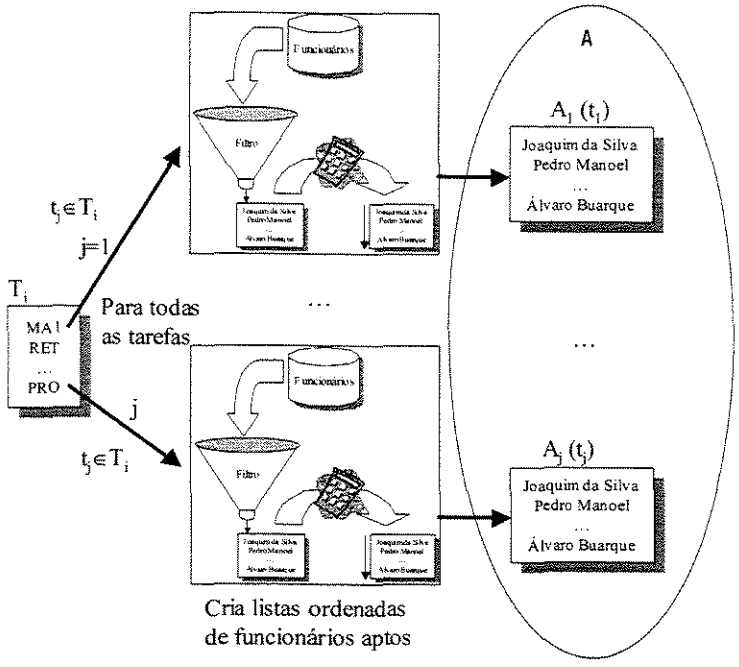

Figura 3.4 - Passo 4 do algoritmo A2

Posteriormente é criada uma lista A que contém as listas A_j. A lista A é então ordenada. O critério utilizado para comparar duas listas A_j é o tamanho das mesmas. Por exemplo, uma lista com 4 elementos é maior do que uma lista com 2 elementos.

Passo 5. No quinto passo, executa-se um laço, que se repete até que a lista A fique vazia, conforme o seguinte:

- t é a tarefa da primeira atribuição da primeira lista de A
- f é o funcionário da primeira atribuição da primeira lista de A
- a tarefa t é atribuída a f
- todos os elementos $a_{jk} \in A_j$ de todas as listas $A_j \in A$, nas quais f_j=f, ou seja, que se referem ao mesmo funcionário f para o qual foi atribuída a tarefa t, são retirados das listas correspondentes.
- a primeira lista A_j é retirada de A
- a lista A é reordenada

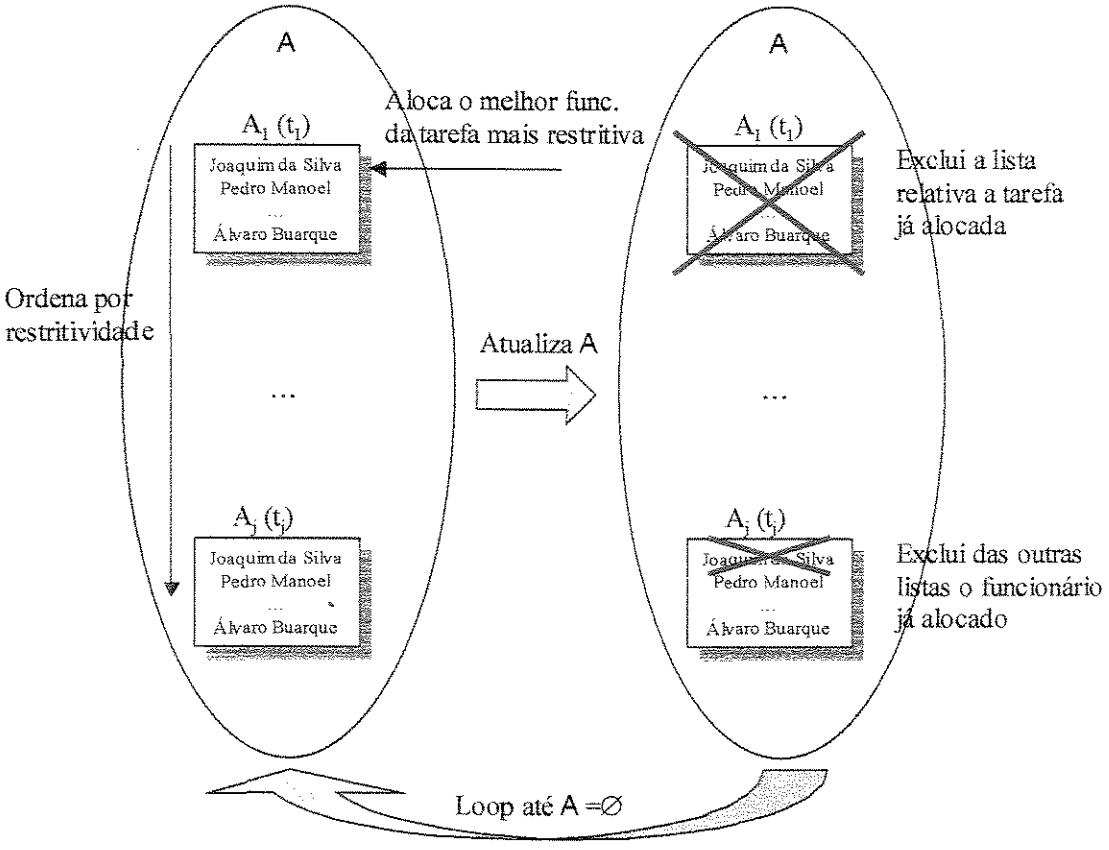

Figura 3.5 – Passo 5 do algoritmo A2

Passo 6. No sexto e último passo, os funcionários para os quais foram realizadas atribuições no passo 5, têm suas estatísticas atualizadas. Também são atualizadas as estatísticas referentes ao destacamento como um todo. Estas estatísticas serão abordadas na seção 3.11.

O algoritmo então retorna ao passo 3 até que i seja o último dia da escala.

Se o algoritmo não falhar, o resultado produzido é uma escala que atende todos os requisitos dados pela PDT e pelas pré-alocações. Este algoritmo falha quando, para uma tarefa escolhida no passo 4, o algoritmo de seleção de funcionários compatíveis (filtro) retorna uma lista vazia. Neste caso é possível realizar um "*backtrack*", desfazendo-se as atribuições realizadas no dia corrente e no dia anterior, decrementando o contador i (dia corrente da escala) e voltando para o passo 3 mas desta vez mudando o critério de seleção de funcionários do passo 5. Por exemplo, pode-se ao invés de selecionar o primeiro funcionário da lista, selecionar o segundo.

3.6.2 Análise e definição formal

Observa-se que, para cada nó não terminal correspondente a uma escala incompleta no início de um dia, são expandidos vários nós. Cada elemento da lista A_j corresponde a um nó. Somente após processar estes nós é que se dá a atribuição. Sendo assim, nota-se que este algoritmo é um algoritmo com características de uma busca em largura. No entanto, como o número total de combinações para cada nó é extremamente grande, somente alguns nós são analisados. Além disto, as expansões só ocorrem para os nós correspondentes ao início de um novo dia da alocação. Portanto o algoritmo combina busca em largura com busca em profundidade.

Quando a lista A é ordenada, o que se faz é tentar aumentar a possibilidade de sucesso do algoritmo, alocando primeiro aquelas tarefas que tem menos pessoas aptas, ou seja, aloca-se primeiro as mais *restritivas*. A reordenação após cada alocação é necessária porque quando uma equipe é alocada a uma tarefa, ela deixa de estar disponível para outras. Assim a ordem das listas A_j pode mudar a cada atribuição (a *restritividade*[11] de cada tarefa pode mudar). No algoritmo A1 isto é realizado parcialmente e de forma implícita ao alocar primeiro as tarefas que iniciam mais cedo. O que se percebeu na prática, utilizando-se o algoritmo A2, é que a ordem cronológica, apesar de semelhante, é, na maioria das vezes, diferente da ordem de restritividade das tarefas.

Da mesma forma que o algoritmo A1, pode-se implementar um "*backtrack*" no algoritmo A2 simplesmente desfazendo-se as atribuições para o dia corrente e o dia anterior e mudando-se, para aqueles dias, o critério de avaliação ou de atribuição.

[11] Restritividade – termo não existente no português oficial. Cunhado aqui para designar a medida do quão restritiva é uma tarefa, ou seja, do quão exigente ela é com relação aos recursos humanos para executa-la em um determinado instante. Normalmente esta medida é dada pelo número de pessoas que estão aptas a executa-la em um determinado instante de tempo

```
1)    Procedure A2 (T,F)
2)    BEGIN
3)       /* Passo 1 e 2*/
4)       Carregar todos os dados relevantes com relação aos funcionários contidos em F
5)       Calcular as horas noturnas e diurnas de cada funcionário de F
6)       FOR (i=1; i<=numero de dias da escala; i++) /*Para todos os dias*/
7)          /* Passo 3 */
8)          backtrack = false;
9)          Ti = lista_de_tarefas(T,i)
10)         Ti = aloca_tarefas_pré-alocadas(Ti,F,i)
11)         x = tamanho de Ti
12)         FOR (j=1; j<= numero de tarefas em Ti; j++)
13)            tj = Ti[j];
14)            Aj = filtro(tj,F,i)
15)            IF(Aj=∅)
16)               backtrack = true;
17)               break; //sai do for mais interno...
18)            ELSE
19)               FOR(k=1; k<= tamanho de Aj; k++)
20)                  ajk = Aj[k];
21)                  ajk.av = avaliação(Aj, t)
22)               ENDFOR
23)               sort(Aj)
24)               insere Aj em A
25)            ENDIF
26)         ENDFOR
27)         IF (not backtrack)
28)            while (A≠∅)
29)               sort(A)
30)               Aux = A[1]
31)               t = Aux[1].t
32)               f = Aux[1].f
33)               Atribui(f,t)
34)               Retira todas os elementos que se referem ao funcionário f em todas as listas Aj∈A
35)               if (∃j | Aj∈A = ∅)
36)                  backtrack = true;
37)                  break;
38)            ENDIF
39)         ENDFOR
40)         IF(backtrack = true)
41)            desfazalocacoes()
42)            i--
43)         ENDIF
44)      ENDFOR;
45)   END;
```

Algoritmo 3.2 - Algoritmo A2

3.7 Algoritmo A3

O algoritmo A3 baseia-se no fato de que a escala de N dias está sendo considerada como N escalas de um dia e que, apesar da solução ótima da escala de N dias não ser necessariamente a concatenação das N escalas ótimas de um dia, esta pode ser uma boa metodologia de alocação. Assim, da mesma forma que o algoritmo A2, a cada início de dia são analisadas várias possibilidades. Estas possibilidades são utilizadas na elaboração de um problema de minimização. Sua solução fornece todas as atribuições que devem ser realizadas naquele dia. Este algoritmo também será aqui apresentado conforme o padrão das seções anteriores.

3.7.1 Um exemplo de aplicação do algoritmo A3

Passos 1, 2 e 3. Idênticos aos apresentados para o algoritmo A1 (seção 3.5.1).

Passo 4. No quarto passo, todas as tarefas (t_j) de T_i são utilizadas, uma a uma, para criar as listas A_j a partir do conjunto F, da mesma forma como foi realizado no algoritmo A2.

Ainda no quarto passo, da mesma forma como realizado no algoritmo A2, a avaliação para cada um dos elementos de A_j é calculada.

Uma vez criadas, cada uma das listas A_j é ordenada utilizando-se das avaliações contidas em cada um dos seus elementos a_{jk}.

Posteriormente é formulado um problema de programação inteira visando atribuir todas as tarefas de forma a otimizar a distribuição. Como veremos mais adiante (seção 3.11), as avaliações calculadas para cada uma das possíveis atribuições são, na verdade, uma medida de erro. Portanto, ao minimizar a somatória destes erros, estaremos otimizando a distribuição. O problema de otimização será descrito com mais detalhes na próxima seção.

Passo 5. No quinto passo, o problema de otimização é resolvido e a solução encontrada é aplicada, ou seja, são realizadas as atribuições descritas pela solução do problema de otimização.

Passo 6. No sexto e último passo, os funcionários para os quais foram atribuídas tarefas no passo 5, têm suas estatísticas atualizadas. Também são atualizadas as estatísticas referentes ao destacamento como um todo. Estas estatísticas serão abordadas na seção 3.11.

O algoritmo retorna ao passo 3 até que i seja o último dia da escala.

Se o algoritmo não falhar, o resultado produzido é uma escala que atende todos os requisitos dados pela PDT e pelas pré-alocações. Este algoritmo falha quando, para uma tarefa escolhida no passo 4, o algoritmo de seleção de funcionários compatíveis (filtro) retorna uma lista vazia ou quando o problema de otimização não tem solução factível.

3.7.2 Análise e definição formal

O algoritmo A3 é muito semelhante ao algoritmo A2. A única diferença é a metodologia utilizada para realizar as atribuições no passo 5. O modelo de programação inteira utilizado no passo 4 e 5 é na verdade o problema de atribuição tradicional por:

$$\min \sum_{A_j} c_{jk} x_{jk} \tag{3.1}$$

sujeito a

$$\sum_{j} x_{jk} = 1 \tag{3.2}$$

$$\sum_{k} x_{jk} \leq 1 \tag{3.3}$$

$$x_{jk} \in \{0,1\} \tag{3.4}$$

onde:

- x_{jk} é uma variável de decisão associada a atribuição $a_{jk} \in A_j$, ou seja, se $x_{jk}=1$, então a tarefa $t_j \in T_i$ é atribuída ao funcionário $f_k \in F$.
- c_{jk} é o valor associado a uma atribuição $a_{jk} \in A_j$

A restrição (3.2) garante que todas as tarefas serão atribuídas a um (e apenas um) funcionário. Por sua vez, a restrição (3.3) garante que cada funcionário irá, receber no máximo, uma atribuição. Esta relação é de desigualdade porque pode haver funcionários que não recebam tarefas.

Observe que devido a forma como é realizada a escolha das atribuições no passo 4 e no passo 5, não é possível implementar um "*backtrack*" alterando-se a forma de atribuição. Neste caso a única maneira é alterar a forma de cálculo dos custos (avaliações).

O problema de otimização inteira resultante tem dimensões reduzidas e pode ser facilmente resolvido através de um algoritmo do tipo simplex. Isto ocorre porque esta formulação resulta em matrizes unimodulares o que garante que o resultado é sempre uma solução inteira [Tie82].

3.8 Algoritmo A4

O algoritmo A4 é uma tentativa de melhorar o desempenho do algoritmo A3. Ele segue exatamente os mesmos passos do algoritmo A3, modificando somente o modelo de programação inteira. Devido a esta semelhança não será apresentado para este algoritmo um exemplo de utilização.

O modelo de programação inteira do algoritmo A4 encontra uma solução que minimiza a função objetivo e ao mesmo tempo procura aumentar a chance de sucesso nas alocações dos dias consecutivos, incluindo no modelo dados sobre o dia seguinte. Este modelo é descrito pelas equações (3.5) a (3.8).

$$\min \sum c_{ij} x_{ij} \tag{3.5}$$

sujeito a

$$\forall_i \left(\sum_{j=1}^{t1} x_{ij} + \sum_{j=1}^{t1} \sum_{k=1}^{t2} y_{ijk} + \sum_{k=1}^{t2} z_{ik} \right) \leq 1 \tag{3.6}$$

$$\forall_j \left(\sum_{i=1}^{w} x_{ij} + \sum_{i=1}^{w} \sum_{k=1}^{t2} y_{ijk} \right) = 1 \tag{3.7}$$

$$\forall_k \left(\sum_{i=1}^{w} z_{ik} + \sum_{i=1}^{w} \sum_{j=1}^{t1} y_{ijk} \right) = 1 \tag{3.8}$$

Seja $G=(N \cup M \cup P, X \cup Y \cup Z)$ um grafo orientado onde:

- $N=\{n_1, n_2, \ldots, n_w\}$ é um conjunto de nós representando todos os funcionários;
- $M=\{m_1, m_2, \ldots, m_{t1}\}$ é um conjunto de nós representando todas as tarefas pertencentes a T_i
- $P=\{p_1, p_2, \ldots, p_{t2}\}$ é um conjunto de nós representando todas as tarefas pertencentes a T_{i+1}.

X é um conjunto de arcos $x_{ij} \in X$ que conectam funcionários n_i a tarefas m_j. Esta variável de decisão representa o caso onde um funcionário é atribuído a uma tarefa no dia d mas não atribuído a nenhuma tarefa no dia d+1. Este é o caso mostrado na Figura 3.6.

Y é um conjunto de arcos $y_{ijk} \in Y$ que conectam funcionários n_i a tarefas m_j e posteriormente a tarefas p_k. Esta variável de decisão representa o caso onde um funcionário é atribuído a uma tarefa no dia d e posteriormente a uma tarefa do dia d+1. Este é o caso representado na Figura 3.7.

Z é um conjunto de arcos $z_{ik} \in Z$ que conectam funcionários n_i a tarefas p_k sem conectar a nenhuma tarefa m_i. Este é o caso em que um funcionário executa uma tarefa do dia d+1 mas não executa nenhuma tarefa do dia d.

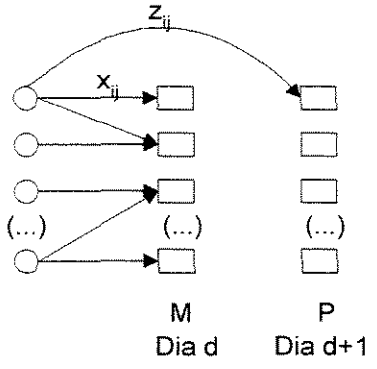

Figura 3.6 - Arcos X e Z

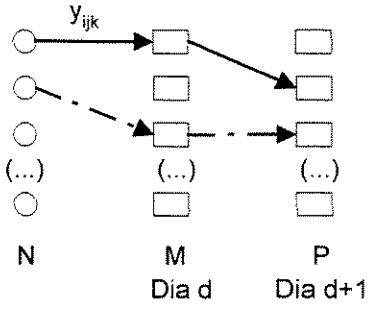

Figura 3.7 - Arcos Y

Os valores c_{ij} são calculados da mesma forma que aqueles dos algoritmos anteriores.

A restrição (3.6) certifica que cada trabalhador irá ser atribuído a no máximo uma tarefa. A restrição (3.7) garante que todas as tarefas do primeiro dia serão atribuídas. Por sua vez, a restrição (3.8) garante que todas as tarefas do segundo dia serão atribuídas.

Alguns cuidados devem ser tomados na construção do modelo com relação às pré-alocações. Estes cuidados aumentam a complexidade da implementação deste algoritmo. São alguns deles:

- uma tarefa que foi pré-alocada somente pode aparecer em arcos relacionados ao funcionário para o qual ela foi pré-alocada e na posição correta (dia d ou dia d+1)
- um arco do conjunto X só é valido nos seguintes casos:
  - quando o funcionário não tem nenhuma pré-alocação nos dias d e d+1, ou
  - quando o funcionário não tem nenhuma pré-alocação de tarefas da PDT no dia d e no dia d+1, ou
  - quando o funcionário tem uma pré-alocação de uma tarefa não pertencente a PDT (exemplo folga ou exame médico) e a tarefa do dia d é compatível com a pré-alocação.

Cuidados adicionais incluem aqueles que verificam a compatibilidade das pré-alocações com os arcos dos conjuntos Y e Z (análogo à análise realizada para os arcos do conjunto X).

3.9 Filtros

Todos os algoritmos apresentados neste capítulo utilizam o que chamamos de "filtros", conforme a Figura 3.3. O filtro é um algoritmo que realiza um processo de seleção de elementos de um conjunto de funcionários conforme ilustrado na Figura 3.8.

A complexidade do algoritmo de filtragem varia dependendo das heurísticas que se deseja implementar. A seguir serão apresentados vários filtros que são combinados em um único para ser utilizado nos algoritmos apresentados nas seções anteriores. Apesar da combinação dos filtros resultar em um único filtro, a ordem em que eles são concatenados pode alterar o resultado. Isto pode ser utilizado para estabelecer prioridades entre eles, o que é interessante uma vez que os objetivos neles implícitos são quase sempre conflitantes.

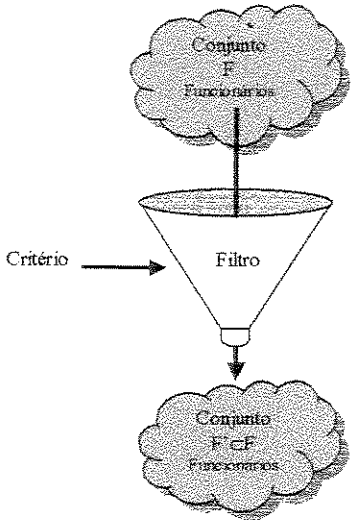

Figura 3.8 – Filtros

3.9.1 Filtro trivial

O filtro trivial é aquele que seleciona os funcionários compatíveis com uma determinada tarefa, ou seja, aqueles que podem executá-la. Este filtro sempre deve ser o primeiro a ser aplicado em qualquer tipo de combinação de filtros utilizada. Sendo assim, seja:

- t a tarefa que se deseja atribuir
- F o conjunto de funcionários
- t_i é a última tarefa que foi executada pelo funcionário $f_i \in F$
- p_i é a próxima pré-alocação para o funcionário $f_i \in F$.
- τ_1 é o tempo mínimo entre uma tarefa e uma extra-tarefa
- τ_2 é o tempo mínimo entre uma tarefa fixa e outra tarefa fixa ou não-fixa
- τ_3 é o tempo mínimo entre uma tarefa não-fixa e uma tarefa fixa
- τ_4 é o tempo mínimo entre duas tarefas não-fixas
- inicio( ) uma função que retorna o início de uma tarefa
- término( ) uma função que retorna o término de uma tarefa

O Algoritmo 3.3 a seguir detalha o filtro trivial.

A diferenciação dos tempos entre tarefas fixas e não-fixas existe devido aos atrasos que normalmente ocorrem nas tarefas não-fixas. Antes de uma extra-tarefa o tempo considerado é sempre o mínimo exigido por lei (τ_1=10 horas). Depois de uma extra-tarefa não é necessário nenhum intervalo para o início de outra tarefa.

O resultado do filtro trivial é um subconjunto de F onde todos os elementos são funcionários aptos a exercitar a tarefa passada como parâmetro, respeitando seu passado e suas pré-alocações futuras.

```
1)    Procedure filtro_trivial(F,t) //Retorna F'
2)    BEGIN
3)       F'=∅
4)       FOR (i=1; i<=|F|; i++)
5)          compativel = false;
6)          IF (t_i é extra-tarefa)                    //Testa compatibilidade entre t_i e t
7)             IF (término(t_i)>=inicio(t)
8)                compativel = true;
9)             ENDIF
10)         ELSEIF (t_i é tarefa fixa)
11)            IF (t é extra-tarefa)
12)               IF (termino(t_i)+ τ1 >=inicio(t))
13)                  compativel = true;
14)               ENDIF
15)            ELSE
16)               IF (termino(t_i)+ τ2 >=inicio(t))
17)                  compativel = true;
18)               ENDIF
19)         ELSEIF (t_i é tarefa não-fixa)
20)            IF (t é extra-tarefa)
21)               IF (termino(t_i)+ τ1 >=inicio(t))
22)                  compativel = true;
23)               ENDIF
24)            ELSEIF (t é tarefa fixa)
25)               IF (termino(t_i)+ τ3 >=inicio(t))
26)                  compativel = true;
27)               ENDIF
28)            ELSEIF (t é tarefa não-fixa)
29)               IF (termino(t_i)+ τ4 >=inicio(t))
30)                  compativel = true;
31)               ENDIF
32)            ENDIF
33)         ENDIF
34)         IF (compativel)                    //Testa compatibilidade entre t e p_i
35)            compativel = false;
36)            IF (t é extra-tarefa)
37)               IF (término(t)>=inicio(p_i)
38)                  compativel = true;
39)               ENDIF
40)            ELSEIF (t é tarefa fixa)
41)               IF (p_i é extra-tarefa)
42)                  IF (termino(t)+ τ1 >=inicio(p_i))
43)                     compativel = true;
44)                  ENDIF
45)               ELSE
46)                  IF (termino(t)+ τ2 >=inicio(p_i))
47)                     compativel = true;
48)                  ENDIF
49)            ELSEIF (t é tarefa não-fixa)
50)               IF (p_i é extra-tarefa)
51)                  IF (termino(t)+ τ1 >=inicio(p_i))
52)                     compativel = true;
53)                  ENDIF
54)               ELSEIF (p_i é tarefa fixa)
55)                  IF (termino(t)+ τ3 >=inicio(p_i))
56)                     compativel = true;
57)                  ENDIF
58)               ELSEIF (p_i é tarefa não-fixa)
59)                  IF (termino(t)+ τ4 >=inicio(p_i))
60)                     compativel = true;
61)                  ENDIF
62)               ENDIF
63)            ENDIF
64)         ENDIF
65)         IF (compativel)
66)            inclui fi no conjunto F'
67)         ENDIF
68)      ENDFOR
69)      retorna F'
```

```
70)   END
```

Algoritmo 3.3 - Filtro trivial

3.9.2 Filtro pernoites

Este filtro cria uma lista de funcionários que, se forem selecionados para executar uma determinada tarefa, não farão dois pernoites seguidos (vide seção 3.2). A utilização deste filtro em combinação com outros, faz com que a repetição de pernoites seja proibida na escala, o que é diferente de penalizar este caso, de alguma forma, na função de avaliação. Portanto seja:

- t a tarefa que se deseja atribuir
- F o conjunto de funcionários
- t_i é a última tarefa que foi executada pelo funcionário $f_i \in F$

O filtro pernoites é simples e é descrito pelo Algoritmo 3.4.

```
1)    Procedure filtro_pernoites(F,t) //Retorna F'
2)    BEGIN
3)       F'=∅
4)       FOR (i=1; i<=|F|; i++)
5)          ok = true;
6)          //Testa compatibilidade entre ti e t
7)          IF (ti seguido de t provoca dois pernoites seguidos)
8)             ok = false;
9)          ENDIF
10)         IF (ok)
11)            inclui fi em F'
12)         ENDIF
13)      ENDFOR
14)      retorna F'
15)   END
```

Algoritmo 3.4 - Filtro pernoites

3.9.3 Filtro de repetição de tarefas

Este filtro cria uma lista de funcionários que, se forem selecionados para executar uma determinada tarefa, não executarão duas tarefas repetidas. Aqui entende-se como tarefas repetidas a repetição de tarefas com o mesmo código, independente do seu horário de início. Portanto seja:

- t a tarefa que se deseja atribuir
- F o conjunto de funcionários
- t_i é a última tarefa que foi executada pelo funcionário $f_i \in F$

O filtro de repetição de tarefas é bastante simples e é descrito pelo Algoritmo 3.5.

```
1)    Procedure filtro_repetição_tarefas (F,t) //Retorna F'
2)    BEGIN
3)       F'=∅
4)       FOR (i=1; i<=|F|; i++)
5)          ok = true;
6)          //Testa compatibilidade entre ti e t
7)          IF (ti é do mesmo tipo de t)
8)             ok = false;
9)          ENDIF
10)         IF (ok)
11)            inclui fi em F'
12)         ENDIF
13)      ENDFOR
```

```
14)       retorna F'
15)   END
```

Algoritmo 3.5 - Filtro de repetição de tarefas

3.9.4 Filtro de preservação de folga

Quando uma tarefa não-fixa precede uma folga, existe o risco de haver atrasos e, com isto, perda de parte da folga do funcionário. Por outro lado, a utilização de um tempo entre tarefas não-fixas e extra-tarefas diferente do mínimo pode levar, em alguns casos, a impossibilidade de criação da escala. Assim, um compromisso aceitável é introduzir um tempo adicional quando for possível. O filtro de preservação de folgas tenta preservar a folga, ou seja, ele tenta inserir um intervalo extra entre as tarefas não-fixas e as extra-tarefas. Se a introdução do tempo entre a tarefa não-fixa e a extra-tarefa levar a um conjunto vazio de funcionários, o intervalo é então flexibilizado. Portanto sejam:

- t a tarefa que se deseja atribuir
- F o conjunto de funcionários
- p_i a próxima pré-alocação para o funcionário $f_i \in F$.
- τ_p o tempo de preservação de folga desejado
- inicio( ) uma função que retorna o início de uma tarefa
- término( ) uma função que retorna o término de uma tarefa

O filtro é descrito pelo Algoritmo 3.6.

```
1)    Procedure filtro_preservação_folgas (F,t) //Retorna F'
2)    BEGIN
3)       F'=∅
4)       tp = τp;
5)       acabou = false;
6)       WHILE (tp>=0 e acabou=false)
7)          FOR (i=1; i<=|F|; i++)
8)             ok = true;
9)             IF (pi é uma extra tarefa
10)                e entre t e pi não cabe nenhuma outra tarefa
11)                e termino(t)+tp>inicio(pi) )
12)                ok = false;
13)            ENDIF
14)            IF (ok)
15)               inclui fi em F'
16)            ENDIF
17)         ENDFOR
18)         IF (F'=∅)
19)            acabou = false;
20)            tp = tp-1; // Menos uma hora - flexibiliza o filtro
21)         ELSE
22)            acabou = true;
23)         ENDIF
24)      ENDWHILE
25)      retorna F'
26)   END
```

Algoritmo 3.6 - Filtro de preservação de folgas

3.9.5 Outros filtros

O mecanismo de filtragem permite a customização do algoritmo de alocação para incluir qualquer tipo de regras. Isto é útil na implementação de situações como, por exemplo: "uma folga dupla somente pode ocorrer quando o banco de horas do funcionário estiver com saldo positivo".

Alguns filtros desenvolvidos não foram mencionados com detalhes nesta seção devido a sua extrema simplicidade e semelhança com os já apresentados. Este é o caso dos seguintes filtros:

- filtro "evita repetição de pegadas";
- filtro "bloqueia repetição de tarefas com mesmo código e hora de início";
- filtro "evita tarefas com horários parecidos";
- etc.

3.10 Pré-alocação de extra-tarefas sem horário pré-determinado

Uma extra-tarefa, diferentemente dos outros tipos de tarefas, pode não ter seu horário pré-definido em uma pré-alocação. Neste caso, o algoritmo deve decidir seu horário de início convenientemente. Para tanto são definidos dois parâmetros úteis:

- hf_{min} – menor hora possível para início de uma folga
- hf_{Max} – maior hora possível para início de uma folga

Normalmente os funcionários das ferrovias denominam as folgas que se iniciam muito cedo, ou muito tarde, de "falsa folga"[12]. Os valores típicos dos parâmetros para evitar "falsas folgas" são os seguintes:

- hf_{min} = 05:00
- hf_{Max} = 17:00

Este intervalo tem que ser levado em consideração quando qualquer um dos algoritmos (principalmente os filtros), testam o início ou o fim de uma extra-tarefa. Nestes casos, apesar de não estar explícito nos algoritmos apresentados, são testados os dois limites de início e fim da extra-tarefa em questão (o inferior e o superior).

3.11 Avaliação

Todos os algoritmos de alocação apresentados neste capítulo (A1,...,A4) utilizam uma função que avalia uma atribuição de uma determinada tarefa a um determinado funcionário. O objetivo principal desta avaliação é garantir que todos os funcionários tenham a mesma carga de trabalho, tanto em horas noturnas quanto em horas diurnas. A divisão entre horas noturnas e horas diurnas é motivada pela diferença de remuneração e desgaste que há entre elas. Ela é a mesma utilizada no capítulo 3, ou seja, é considerado como trabalho diurno todo aquele que ocorre entre 5:00 e 22:00. O trabalho noturno ocorre no período complementar do trabalho diurno.

[12] Jargão utilizado em algumas ferrovias brasileiras

Para realizar esta avaliação, diferentes abordagens foram desenvolvidas. Todas elas retornam um valor entre 0 e 1 para indicar a qualidade de uma determinada atribuição. Quanto mais próximo de 0, melhor é a atribuição (para o funcionário e para o destacamento).

Durante os primeiros passos dos algoritmos de alocação apresentados, é levantado para cada um dos funcionários os valores Wd_i e Wn_i, que são, respectivamente, o número de horas diurnas e o número de horas noturnas trabalhadas por cada um dos funcionários no passado. Normalmente é utilizado um período de 2 a 3 meses passados. Exemplos destes valores podem ser obtidos na seção 3.4. No decorrer do algoritmo de alocação, estes valores são atualizados para refletir o total trabalhado até o dia corrente da alocação.

Para uma tarefa t_j e um funcionário f_i, as seguintes fórmulas podem ser aplicadas no cálculo da avaliação da atribuição da tarefa para o funcionário em questão:

$$Eval1_{ij} = Worst_i = \max(ED_i, EN_i) \quad (3.9)$$

$$Eval2_{ij} = E_{ij} = Av_{ij} \quad (3.10)$$

$$Eval3_{ij} = \nabla E_{ij} = Av_{ij} - Av_i \quad (3.11)$$

$$Eval4_{ij} = fuzzy(ED_{ij}, EN_{ij}) \quad (3.12)$$

Onde:

Fórmula	Significado	
$Td = \sum_i Wd_i$	total de trabalho diurno do destacamento até o presente momento	(3.13)
$Tn = \sum_i Wn_i$	total de trabalho noturno do destacamento até o presente momento	(3.14)
$Tdf = \sum_i Wd_i + Tdna$	total de trabalho diurno do destacamento em algum ponto no futuro (usualmente no final do dia corrente)	(3.15)
$Tnf = \sum_i Wn_i + Tnna$	total de trabalho noturno do destacamento em algum ponto no futuro (usualmente no final do dia corrente)	(3.16)
Tdna	total de trabalho diurno a ser alocado do presente momento até um ponto desejado no futuro	(3.17)
Tnna	total de trabalho noturno a ser alocado do presente momento até um ponto desejado no futuro.	(3.18)
N_j	total de trabalho noturno da tarefa t_j	(3.19)
D_j	total de trabalho diurno da tarefa t_j	(3.20)
$\overline{D} = Td/w$	média de trabalho diurno de um destacamento	(3.21)
$\overline{N} = Tn/w$	média de trabalho noturno de um destacamento	(3.22)
$\overline{\overline{D}}_j = \frac{(Td + D_j)}{w}$	média de trabalho diurno de um destacamento após a alocação da tarefa j	(3.23)
$\overline{\overline{N}}_j = \frac{(Tn + N_j)}{w}$	média de trabalho noturno de um destacamento após a alocação da tarefa j	(3.24)
$ED_i = \frac{Wd_i - \overline{D}}{\overline{D}}$	erro (desvio com relação a média) do trabalho diurno do funcionário i	(3.25)

$$EN_i = \frac{Wn_i - \overline{N}}{\overline{N}}$$ erro (desvio com relação a média) do trabalho noturno do funcionário i (3.26)

$$ED_{ij} = \frac{(Wd_i + D_j) - \overline{\overline{D_j}}}{\overline{\overline{D_j}}}$$ erro (desvio com relação a média) do trabalho diurno do funcionário i se a tarefa t_j for atribuída a ele (3.27)

$$EN_{ij} = \frac{(Wn_i + N_j) - \overline{\overline{N_j}}}{\overline{\overline{N_j}}}$$ erro (desvio com relação a média) do trabalho noturno do funcionário i se a tarefa t_j for atribuída a ele (3.28)

$$Av_i = 1 - \exp\left(-\left(Ed_i^2 + En_i^2\right)\right)$$ medida do erro da carga de trabalho do trabalhador i (3.29)

$$Av_{ij} = 1 - \exp\left(-\left(Ed_{ij}^2 + En_{ij}^2\right)\right)$$ medida do erro da carga de trabalho do trabalhador i se a tarefa t_j for atribuída a ele (3.30)

fuzzy é uma função que calcula a medida do erro da carga de trabalho através de uma base de regras fuzzy [Lee94], [Gom98].

A Tabela 3.5 a seguir resume as funções de avaliações que serão detalhadas a seguir:

Função	Significado
Eval1	minimizar o máximo do erro da distribuição da carga de trabalho
Eval2	minimizar o erro da distribuição da carga de trabalho
Eval3	maximizar a melhoria na distribuição da carga de trabalho (medida do gradiente)
Eval4	base de regras fuzzy que utiliza *Eval3* e *Eval2*

Tabela 3.5 – Funções de avaliação

A equação de avaliação $Eval1_{ij}$ (3.9) tem como filosofia, a busca pela minimização do máximo erro da distribuição da carga de trabalho.

Tanto a equação $Eval2_{ij}$ (3.10) quanto a $Eval3_{ij}$ (3.11), agregam os erros das horas noturnas e diurnas em um único valor utilizando uma função exponencial. A idéia por trás desta função é implementar uma superfície de decisão semelhante à da Figura 2.13.

A equação $Eval2_{ij}$ (3.10) fornece uma medida do quanto o funcionário i estará distante da média do destacamento se a tarefa t_j for atribuída a ele. Já a função *Eval3* (3.11) dá uma medida do gradiente desta distância, ou seja, do quanto ele melhora (ou piora), comparado com a média do destacamento, se a tarefa t_j for atribuída a ele. Valores negativos trazem o funcionário para próximo da média do destacamento, enquanto valores positivos o levam para longe desta mesma média.

A equação *Eval4* (3.12) usa uma base de regras fuzzy para inferir o valor da qualidade da atribuição. A base de regras fuzzy utiliza três entradas e uma saída. Em outras palavras, três variáveis lingüísticas de entrada e uma de saída.

- entrada 1 = *erroHorasDiurnas* (ED_{ij})
- entrada 2 = *erroHorasNoturnas* (EN_{ij})
- entrada 3 = *gradiente* ($Eval3_{ij}$)
- saída = *avaliação*

As variáveis lingüísticas *erroHorasDiurnas* e *erroHorasNoturnas* têm o mesmo universo de discurso e são representadas com os mesmos valores lingüísticos:

- ZE = zero;
- PS = positivo pequeno;
- PB = positivo grande;
- NS = negativo pequeno;
- NB = negativo grande.

A Figura 3.11 e a Figura 3.12 mostram detalhes do universo e da granularização destas variáveis. A variável lingüística *gradiente* utiliza os seguintes valores lingüísticos:

- N = negativo;
- Z = zero;
- P = positivo.

A Figura 3.10 mostra detalhes do universo e da granularização desta variável.

A variável lingüística *avaliação* utiliza os seguintes valores lingüísticos:

- G = bom (*Good*);
- B = ruim (*Bad*);
- VG = muito bom (*Very Good*);
- VB = muito ruim (*Very Bad*);
- NE = neutro (*NEutral*);

A Figura 3.13 mostra detalhes do universo e da granularização desta variável.

A Figura 3.9 mostra a base de regras utilizadas para realizar as inferências. As regras ali expressas foram obtidas através de especialistas de centro de escalas de diferentes ferrovias.

ED_{jk} \ EN_{jk}	NB	NS	ZE	PS	PB
NB	VB	VB	VB	VB	VB
NS	VB	G	G	B	VB
ZE	VB	G	VG	NE	VB
PS	VB	B	NE	B	VB
PB	VB	VB	VB	VB	VB

E

Se ($Eval3_{ij}$ é N) então (avaliação é VG)
Se ($Eval3_{ij}$ é Z) então (avaliação é N)
Se ($Eval3_{ij}$ é P) então (avaliação é VB)

Figura 3.9 - Base de regras fuzzy

Para reduzir o tempo de desenvolvimento, a base de regras fuzzy foi implementada utilizando-se do *fuzzy toolbox* do software Matlab e depois introduzida no sistema através de uma tabela. A Figura 3.14 mostra uma interface do Matlab contendo outras informações relevantes na definição do sistema de inferência fuzzy, como por exemplo, o tipo de defuzzificação utilizada e a definição dos operadores.

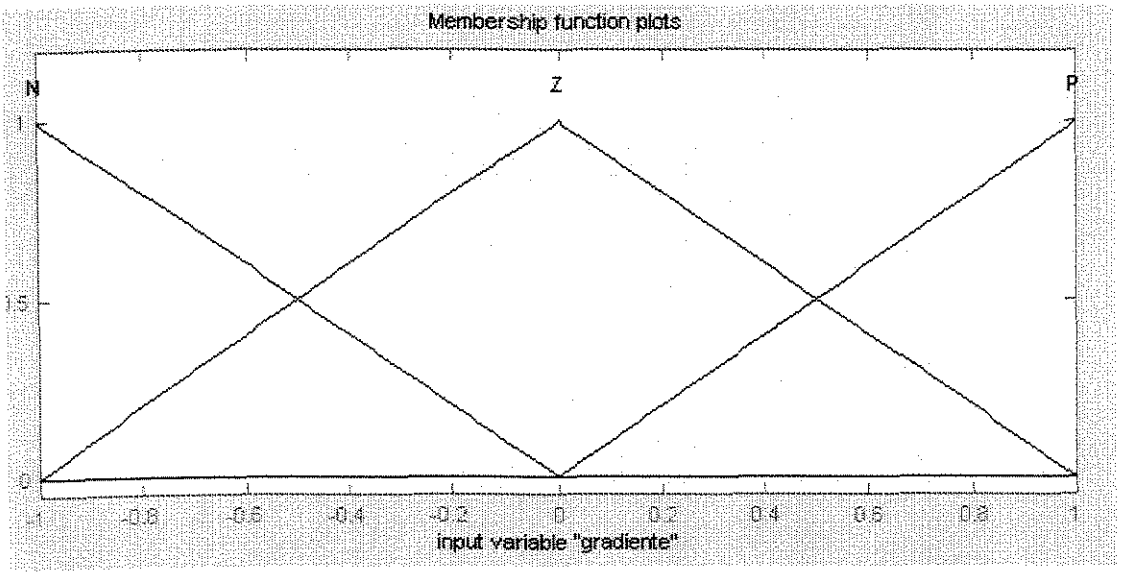

Figura 3.10 - Funções de pertinência dos valores lingüísticos da variável lingüística "gradiente"

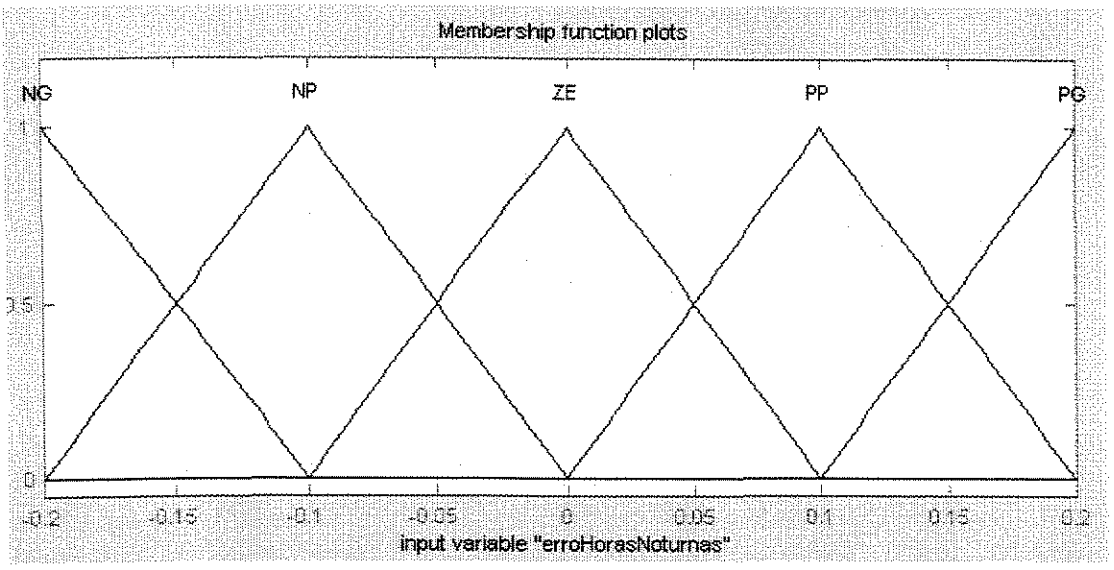

Figura 3.11 - Funções de pertinência dos valores lingüísticos da variável lingüística "erroHorasNoturnas"

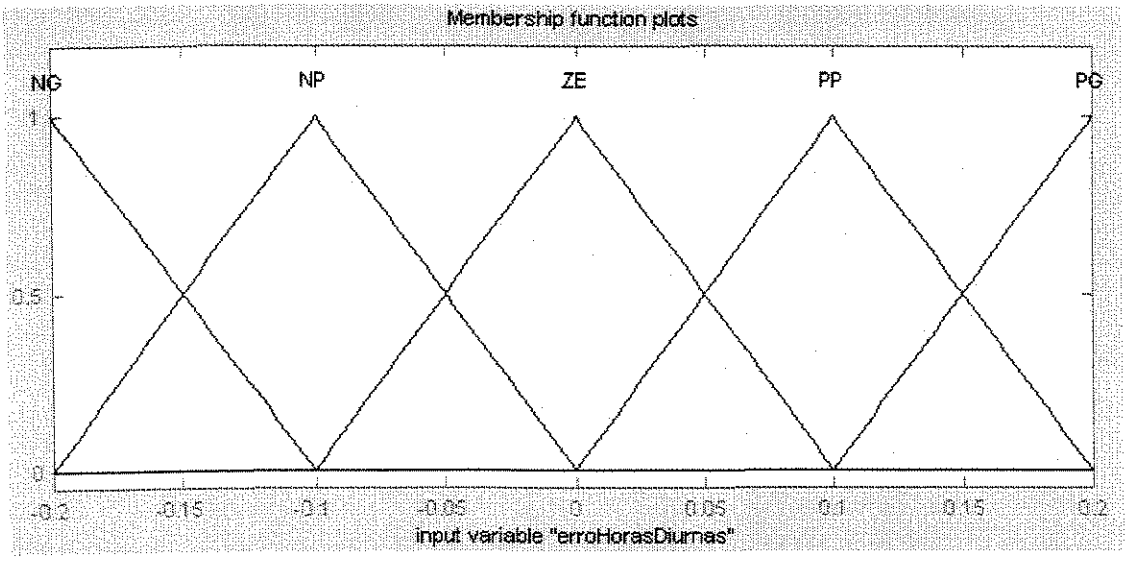

Figura 3.12 - Funções de pertinência dos valores lingüísticos da variável lingüística 'erroHorasNoturnas'

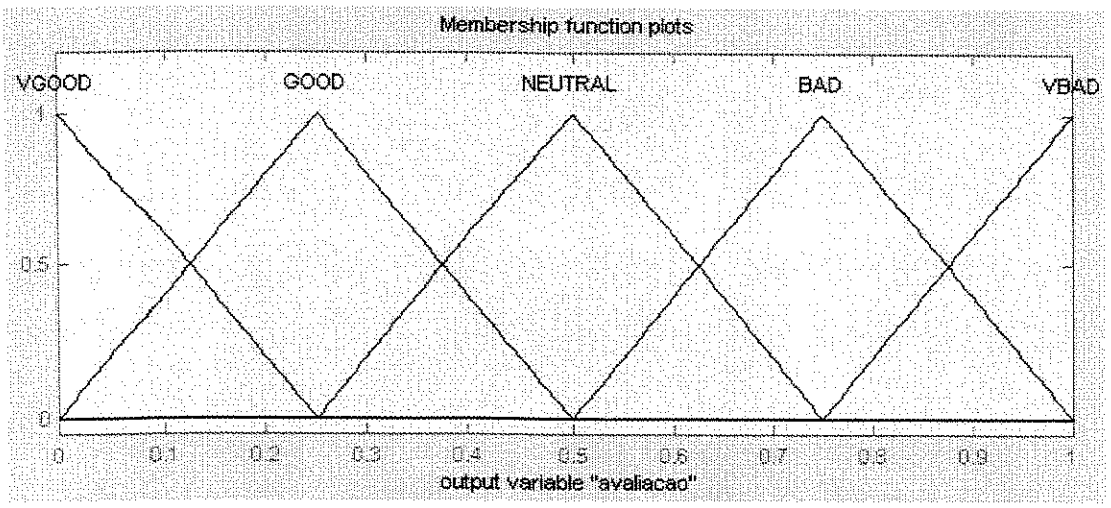

Figura 3.13 - Funções de pertinência dos valores lingüísticos da variável lingüística 'avaliação'

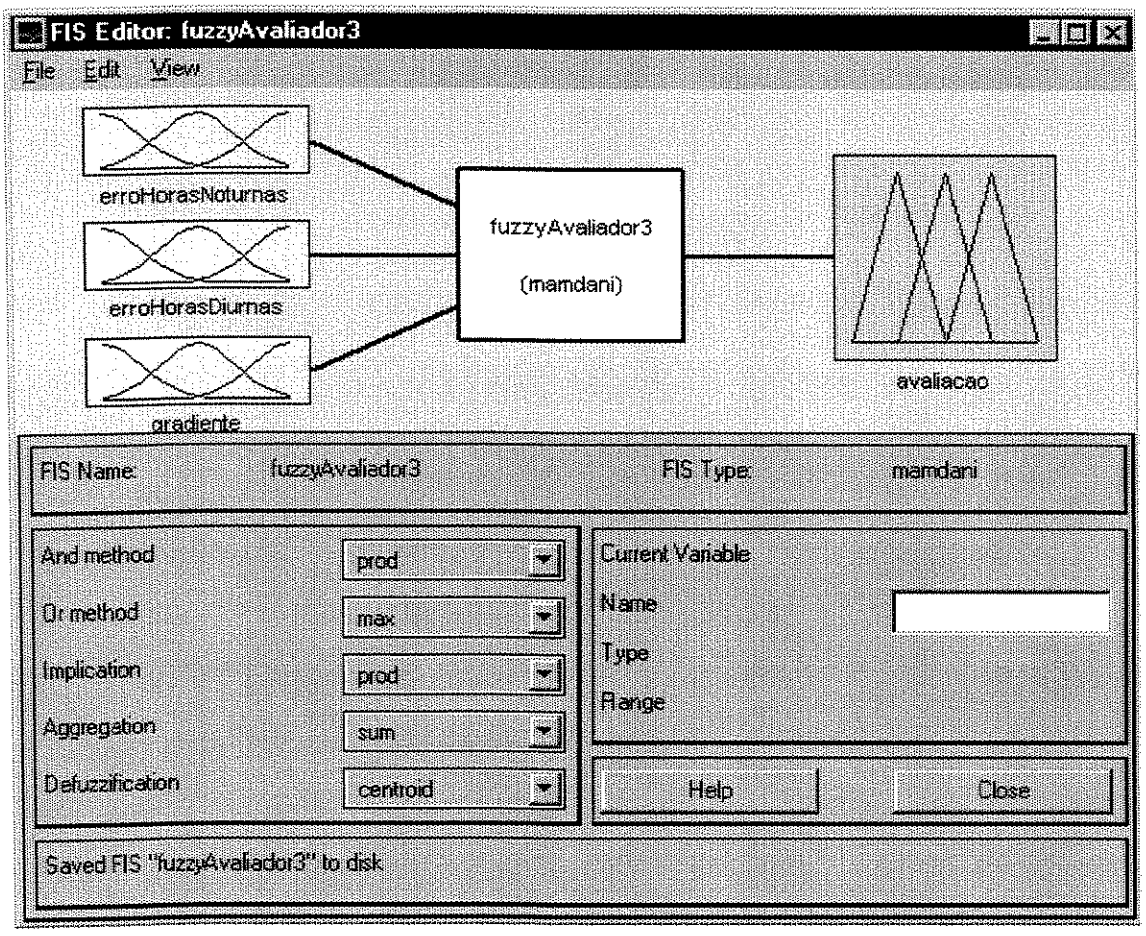

Figura 3.14 – Interface do Matlab contendo parâmetros importantes

3.12 Discussão e resultados

Este capítulo apresentou quatro diferentes algoritmos para a criação de escalas através do paradigma denominado método direto. Estes quatro algoritmos são especializações de um algoritmo genérico apresentado na seção 3.3. Cada um destes algoritmos pode ser combinado com 4 diferentes estratégias de avaliação (*Eval1* a *Eval4*), resultando assim em 16 diferentes algoritmos.

A primeira vista os algoritmos baseados em programação matemática (A3 e A4) parecem ser melhores do que os outros. Isto nem sempre é verdade porque, dificilmente uma escala ótima de N dias é a concatenação de N escalas ótimas de um dia. Além do mais, a prática mostra que todos os algoritmos (inclusive o A3 e o A4) podem levar a situações onde não há solução, forçando o "*backtrack*". Neste caso os algoritmos A3 e A4 encontram maiores dificuldades. Isto ocorre porque nos algoritmos A1 e A2, após ocorrer um "*backtrack*", pode-se mudar o critério de atribuição e o critério de avaliação. Já nos algoritmos A3 e A4, somente o critério de avaliação pode ser modificado já que o critério de atribuição é sempre o mesmo (a solução da programação inteira). Assim, como as várias avaliações fornecem medidas diferentes para a mesma grandeza, podem ocorrer situações em que o "*backtrack*" efetivamente não muda de caminho, levando a uma falha do algoritmo.

Outra diferença que podemos destacar é que o algoritmo A1 utiliza um horizonte mais curto que os outros, conforme a Tabela 3.6.

Algoritmo	Horizonte de análise
A1	Uma tarefa
A2	Um dia
A3	Um dia
A4	Dois dias

Tabela 3.6 - Horizontes de decisão utilizados pelos algoritmos

Todos os algoritmos foram testados em 10 casos reais (semelhantes ao apresentado na seção 3.4) com características diferentes e parâmetros diferentes (principalmente intervalo entre tarefas e distância entre folgas). A Tabela 3.7 sumariza os resultados obtidos.

Todos os métodos de avaliação também foram testados. Os resultados obtidos indicam que as avaliações *Eval1* e *Eval2* normalmente levam a algoritmos mais flexíveis enquanto as *Eval3* e *Eval4* levam a soluções melhores.

Algoritmo	Ranking		
	Flexibilidade	Qualidade	Custo computacional
A1	90% de sucesso	4°	1°
A2	40% de sucesso	3°	2°
A3	60% de sucesso	1°	3°
A4	60% de sucesso	2°	4°

Tabela 3.7 - Resultados obtidos

3.13 Unificando o método direto – algoritmo AM

Conforme exposto na seção anterior, o método direto é uma coleção de algoritmos e avaliações que podem ser combinados entre si. Cada uma destas combinações entre algoritmos e avaliações leva a resultados qualitativamente diferentes. Além disto, cada uma delas tem uma chance de sucesso diferente. Na tentativa de se criar um algoritmo único, que aproveite o melhor de cada um dos outros, criou-se o algoritmo AM (Algoritmo Misto).

No algoritmo AM um dos outros algoritmos é escolhido para ser o "*default*". O mesmo ocorre com a avaliação. O algoritmo e a avaliação escolhida são utilizados para gerar a escala normalmente. Se ocorre uma falha, faz-se o "*backtracking*", conforme o algoritmo escolhido, trocando-se a avaliação por outra. Se todas as avaliações forem utilizadas no mesmo ponto sem sucesso, então o algoritmo (para aquele ponto) é trocado. Assim todas as combinações entre algoritmos e avaliações são testadas. Se nenhuma das combinações for bem sucedida, é realizado um "*backtracking*" maior (volta-se dois dias) e novamente todas as combinações são testadas. Se a falha persistir, pode-se realizar um novo "*backtracking*" até um limite de dias pré-determinado.

Usando-se o algoritmo AM com A3 e *Eval4* como "*defaults*", chegou-se a resultados promissores: a porcentagem de sucesso foi de 90% com resultados de qualidade praticamente igual ao algoritmo A3. Os casos em que o algoritmo AM teve menor qualidade do que o algoritmo A3 foram aqueles que o algoritmo A3 falhou. Em compensação o custo computacional do algoritmo AM é, no pior caso, muito maior do que os outros.

3.14 Resumo

Neste capítulo foram apresentados os algoritmos desenvolvidos para a alocação de escalas de equipagens no paradigma das escalas individualizadas, também chamado de método direto. Até onde sabemos, não há disponível na literatura trabalhos que abordem a aplicação deste paradigma no contexto de alocação de escalas de equipagens ferroviárias.

Diferentemente da abordagem cíclica, o método direto não utiliza o conceito de seqüencial de tarefas. Foram apresentados 4 diferentes abordagens para um mesmo algoritmo genérico, variando, dentre outras coisas, o horizonte de tempo considerado. Todas as abordagens utilizam avaliações das possíveis atribuições que podem ser realizadas entre funcionários e tarefas. Para estas avalições foram prospostas 4 diferentes abordagens.

Os resultados obtidos com os algoritmos e as avaliações propostas neste capítulo serão discutidas no capítulo 5.

4. Sistema de planejamento de escalas de equipagem

4.1 Introdução

Para testar e validar os métodos e algoritmos propostos neste trabalho, foi desenvolvido um sistema computacional para planejamento de escalas de equipagem. Este sistema tem como característica a filosofia de sistema de suporte a tomada de decisão. Dentro desta filosofia, todas as saídas do sistema são vistas como sugestões. A decisão final cabe sempre ao usuário. Além disto, houve uma preocupação especial no projeto das interfaces. Além de simples, a interface do sistema provê as informações necessárias nos locais e momentos mais apropriados. Ela também é ativa no sentido de que sempre critica as ações do usuário, fornecendo avisos e sugestões.

4.2 Estrutura do sistema

Uma propriedade do sistema é sua modularidade. Ele foi concebido para ser facilmente customizável e integrável. Assim ele pôde ser testado em várias circunstâncias práticas diferentes, garantindo a validação dos algoritmos propostos. A Figura 4.1 apresenta a estrutura geral do sistema:

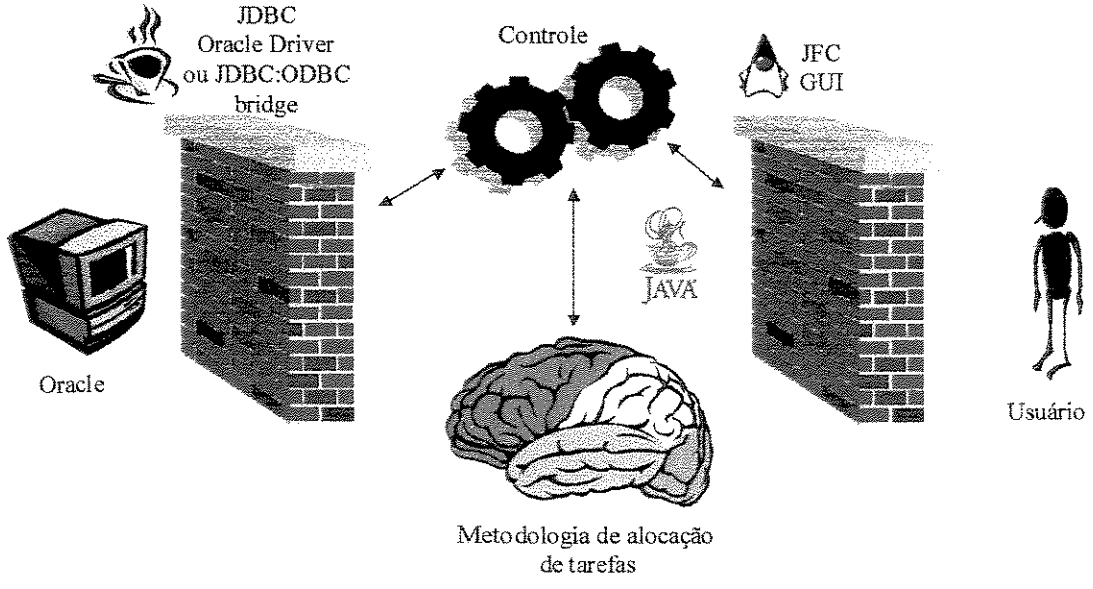

Figura 4.1 - Módulos do sistema computacional

4.2.1 Interface com o usuário (GUI)

A interface com o usuário, assim como todo o sistema, foi implementada em Java com um *look'n'feel* independente da plataforma. Assim sua aparência independe da máquina e do sistema operacional que executa a aplicação. Conforme mencionado anteriormente, um grande esforço foi desprendido para que a interface fosse a mais intuitiva e útil possível preservando sempre a filosofia de sistema de apoio à tomada de decisão.

4.2.2 Controle e algoritmos

O módulo de controle e algoritmos de alocação é o núcleo do sistema. Como foi utilizado o paradigma de orientação a objetos na confecção do sistema, algumas diferenças semânticas entre ambientes diferentes são facilmente resolvidas através de mecanismos de herança e sobrecarga de métodos, sem alterar os algoritmos de alocação.

4.2.3 Interface com a base de dados

A interface com a base de dados isola a fonte de dados do usuário do restante do sistema. Com isto, basta a substituição deste módulo que ele se adapte a uma base de dados com configuração diferente. Isto garante uma fácil adaptação à diferentes ambientes.

4.2.4 Base de dados

A base de dados não faz parte do sistema em si. Ela é a fonte dos dados que o alimenta podendo ser desde uma base de dados de pequeno porte como o Access, até uma base de dados de grande porte como o Oracle.

4.3 Roteiro de geração de escalas

A interface do sistema (Figura 4.2) é composta por painéis numerados que sugerem um roteiro de geração de escalas conforme Figura 4.3.

O primeiro passo é a inicialização. Na inicialização é estabelecida uma conexão com a base de dados e todos as informações relevantes são obtidas.

O segundo passo do roteiro é a "preparação". Na preparação são realizadas várias entradas de dados. A Tabela 4.1 resume estas entradas de dados e sua importância no processo de criação de escalas.

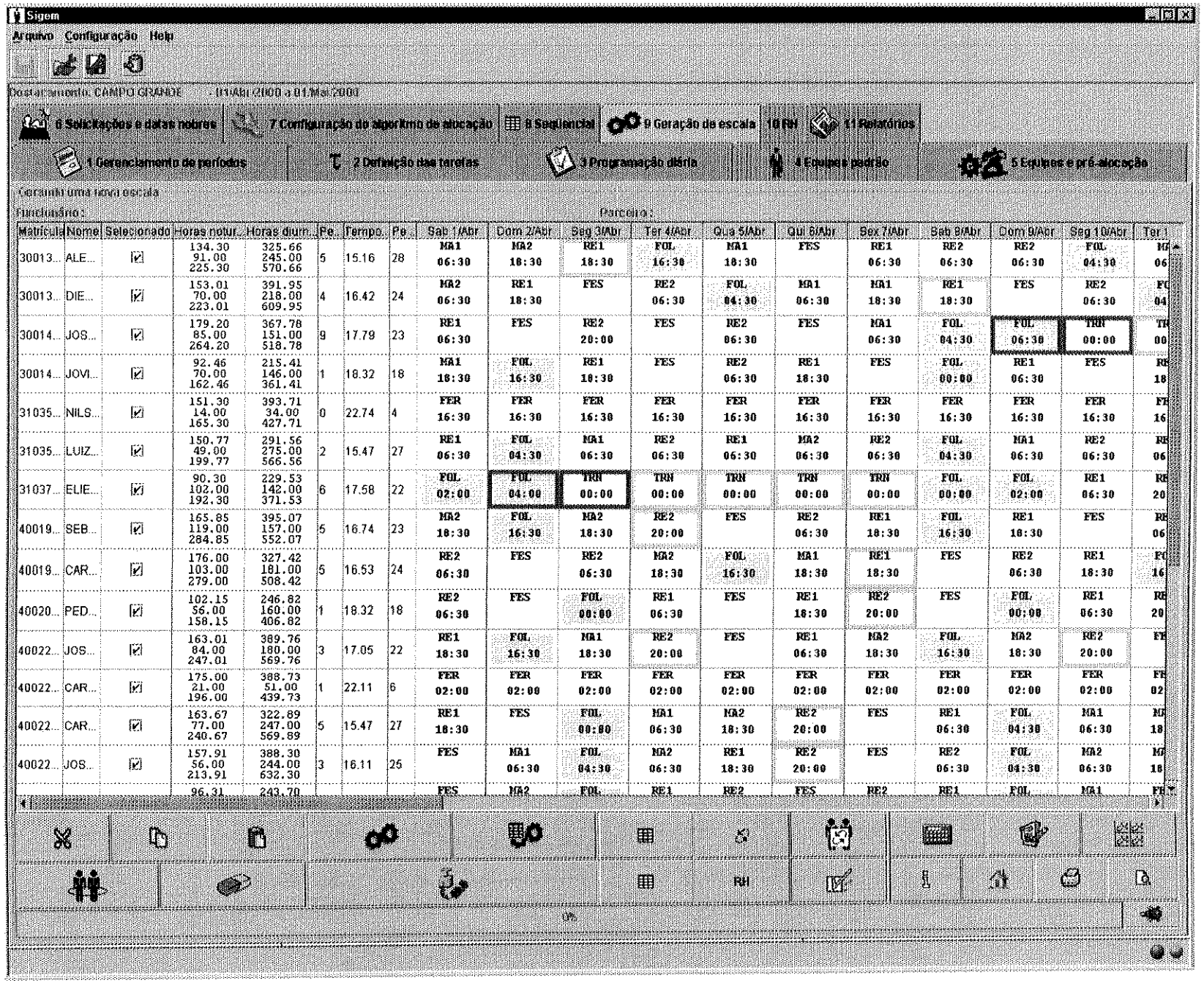

Figura 4.2 - Sistema de planejamento de escalas

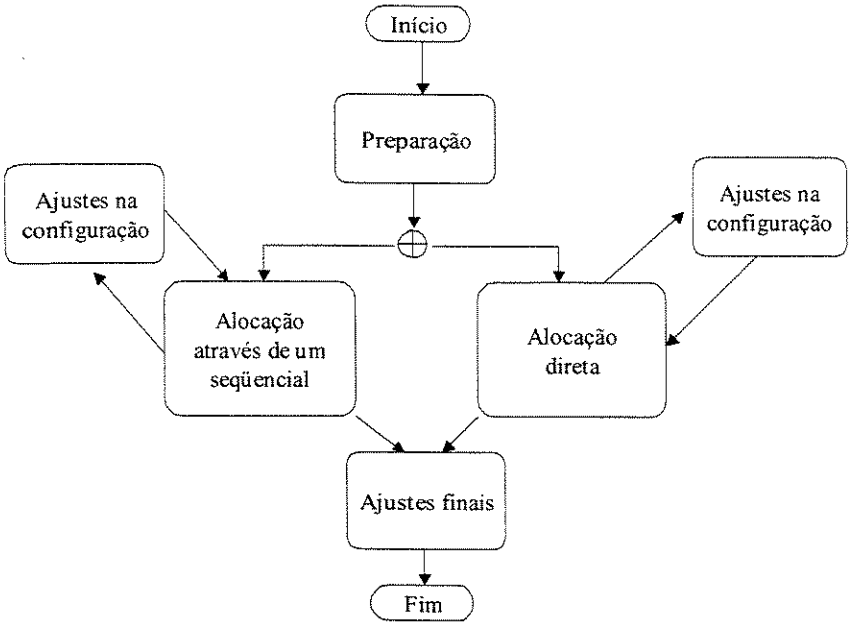

Figura 4.3 - Roteiro de geração de escalas

Etapa	Perguntas respondidas na etapa
Definição dos períodos	Que período passado é relevante para a escala?
Definição das tarefas	Como são as tarefas que serão executadas pelos funcionários?
Definição da programação diária de tarefas	O que os funcionários farão?
Definição das equipes padrão	Normalmente, quem trabalha com quem?
Definição das equipes	Este mês, quem trabalha com quem?
Manutenção das estatísticas	Quem trabalhou menos/mais porque teve sorte/azar e quem trabalhou menos/mais porque estava de férias ou em treinamento?
Pré-alocação de folgas e outros eventos do RH	Quais são as pré-alocações de extra-tarefas para esta escala?
Pré-alocação de tarefas	Quais são as pré-alocações de tarefas para esta escala?

Tabela 4.1- Etapas da preparação

O sistema provê meios para a realização de cada uma destas entradas de dados. Muitas delas têm interfaces próprias, conforme ilustrado na Figura 4.4.

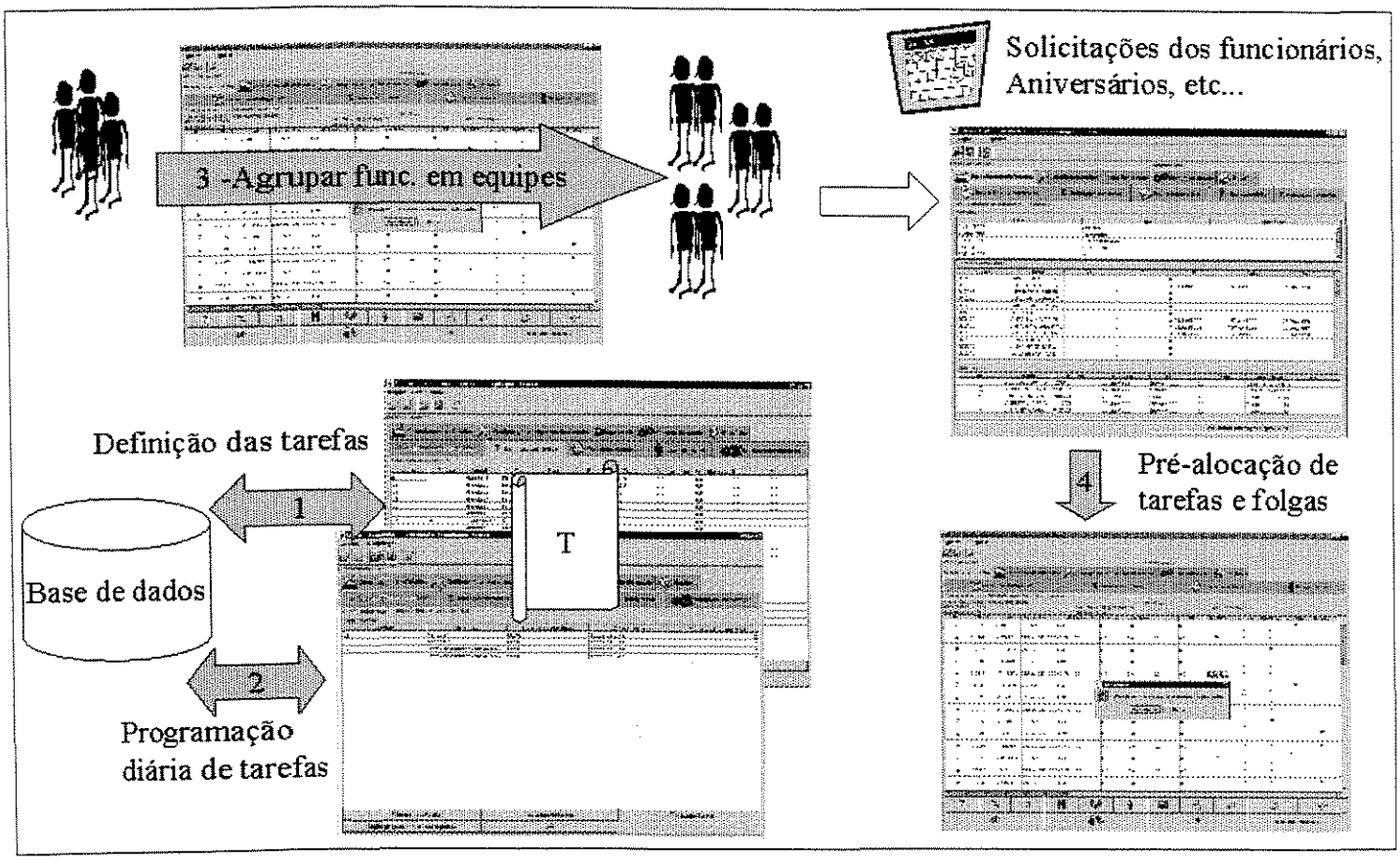

Figura 4.4 -Preparação

Depois de realizada a preparação, o usuário deve optar por gerar a escala através do método de seqüencial de tarefas ou através do método direto. A geração de escalas através do método do seqüencial de tarefa se dá em duas etapas: geração do seqüencial e atribuição do seqüencial aos funcionários, respectivamente. Muitas vezes são realizadas várias iterações, modificando-se as configurações do sistema até que a escala definitiva seja gerada. Estas iterações são sempre realizadas com o intuito de se melhorar a qualidade da escala conforme as particularidades de cada destacamento.

Uma vez obtida a escala sempre é necessário um ajuste final. No caso das escalas geradas através do método seqüencial, são necessários os ajustes de férias e pré-alocações. No caso da escala direta, pode-se ajustar pequenos desvios ou situações indesejadas trocando-se tarefas entre os funcionários.

5. Resultados

5.1 Introdução

Apesar de haver muitos trabalhos publicados sobre alocação de recursos humanos, somente um subconjunto muito restrito se refere à alocação de trabalho para equipagens ferroviárias. Além disto, percebe-se que o problema resolvido pelos algoritmos expostos na literatura, apesar de sua grande complexidade, é somente uma parte do problema de gerenciamento de equipagem que os centros de escala das ferrovias lidam diariamente. Encarando o problema de gerenciamento de equipagem de uma forma mais ampla, percebe-se que um grande esforço é dispendido em etapas freqüentemente ignoradas pela literatura, como a consideração de pedidos de extra-tarefas e outras exceções. Além disto, nota-se que, devido ao caráter multi-critério do problema de geração de escalas e de um grande número de critérios de avaliação que são variáveis e não quantificáveis, dificilmente uma solução "ótima" segundo uma função objetivo, por melhor que ela seja, é ótima do ponto de vista do usuário.

Este trabalho introduziu novos algoritmos de planejamento de escalas de equipagem considerando dois paradigmas possíveis: cíclico e individualizado. É difícil comparar os resultados obtidos nas seções 2 e 3 com os existentes na literatura devido às particularidades do domínio da aplicação. Além disto, a literatura não provê dados suficientes para subsidiar uma comparação. O exemplo descrito na seção 3.4 tem o intuito de cobrir esta lacuna e viabilizar comparações com trabalhos futuros. Apesar desta dificuldade, a aplicação prática dos algoritmos indicou que os mesmos são promissores para resolver a classe de problemas de planejamento de equipagem aqui considerada.

A seguir apresenta-se os resultados e conclusões dos métodos e algoritmos propostos.

5.2 Método do seqüencial de tarefas

Conforme exposto nas seções 1.3 e 2.1, as ferrovias do país, em sua totalidade, utilizam métodos manuais para a criação e alocação de escalas. O sistema computacional aqui desenvolvido foi testado por especialistas de algumas destas ferrovias com sucesso. A filosofia de sistema de suporte de tomada de decisão assegurou a viabilidade e a realização de testes pois ela reduz a curva de aprendizagem do sistema. Os testes efetuados com especialistas ocorreram de forma gradativa. No início o sistema computacional foi utilizado somente como uma interface sofisticada para se gerar escalas manualmente. Logo depois descobriu-se que era possível poupar tempo fazendo a conexão do seqüencial (gerado manualmente) com o passado de forma automática (vide seção 2.6). Somente em uma etapa posterior o sistema foi utilizado para criar seqüenciais de tarefas, complementado assim o ciclo de geração através do método do seqüencial de tarefas.

O teste completo do sistema somente ocorreu vários meses após o início de sua utilização. Apesar disto os resultados foram promissores. O tempo médio de geração de uma escala caiu em (no pior caso) 50%. Os especialistas passaram a utilizar o tempo extra para analisar melhor outros problemas do gerenciamento de equipagens, indicando ganhos adicionais na realização das escalas.

A Tabela 5.1 resume os resultados obtidos com os algoritmos de criação de seqüencial de tarefas propostos neste trabalho. Observe que os dois algoritmos praticamente não utilizam uma condição de parada fixa. Isto ocorre porque o algoritmo de busca raramente experimenta todas as combinações possíveis e o algoritmo evolutivo raramente atinge o limite máximo de gerações. Portanto, em todos os testes, os algoritmos foram terminados após 3 e 10 minutos. Estes são intervalos de tempo toleráveis pelo usuário para obter uma solução.

Método	Número de passos do seqüencial	Avaliação após 3 minutos	Avaliação após 10 minutos
Busca	30	0.37	0.35
Evolutivo	30	0.53	0.35
Busca	50	0.47	0.40
Evolutivo	50	0.49	0.41
Busca	80	0.54	0.53
Evolutivo	80	0.47	0.45

Tabela 5.1 – Comparação entre os algoritmos de criação de seqüencial de tarefas

O método de criação de seqüencial de tarefas baseado em algoritmos de busca mostrou-se mais eficiente para seqüenciais menores uma vez que chega rapidamente a uma boa solução e, se necessário, usa tempo adicional para refina-la. O método baseado em computação evolutiva mostrou-se mais eficiente para seqüenciais maiores, pois o tempo de busca é mais dispendioso quando necessita analisar um espaço de busca mais amplo, mesmo realizando podas e limitando o número máximo de nós expandidos.

Estes resultados e a experiência adquirida após inúmeros experimentos levam a tabela de decisão ilustrada a seguir:

		Tamanho do seqüencial	
		< 35 passos	≥ 35 passos
Tempo de espera	3 a 5 minutos	Busca	Evolutivo
	> 5 minutos	Evolutivo	Evolutivo

Tabela 5.2 – Tabela de decisão entre algoritmos para criação de seqüencial de tarefas

5.3 Método direto

O método direto foi desenvolvido neste trabalho tendo em mente que um dos grandes problemas do gerenciamento de equipagens é o controle e distribuição de pedidos de extra-tarefas e outras exceções. Além disso, gerar escalas justas do ponto de vista da distribuição da carga de trabalho é uma das grandes preocupações dos centros de escalas das ferrovias.

Para analisar a eficiência deste método, foi considerado o exemplo da seção 3.4. A escala gerada para este caso, usando o algoritmo AM, foi considerada boa pelos especialistas. Os indicadores utilizados para sua avaliação estão ilustrados nas Figuras 5.1 a 5.5.

As Figuras 5.1, 5.2 mostram a distribuição da carga de trabalho no passado que foi levada em consideração durante a geração da escala (vide dados na seção 3.4) e a distribuição da carga de trabalho final em horas noturnas e diurnas. O erro final da distribuição das horas noturnas e diurnas leva em consideração o passado e a escala gerada. Pode-se notar claramente que houve uma melhoria na distribuição da carga de trabalho, pois um número maior de funcionários está concentrada nas faixas com erros em torno do zero. A Figura 5.3 mostra a distribuição da somatória dos valores absolutos destes erros (erro das horas diurnas e noturnas).

As Figuras 5.4, 5.5 mostram a distribuição da variação do erro. Na Figura 5.4 são mostradas as distribuições para os erros das horas noturnas e diurnas e na Figura 5.5 são mostradas as distribuições para a somatória dos absolutos destes erros. Este valor é zero quando não há mudança no erro, positivo quando o erro aumenta e negativo quando o erro diminui. Pode-se observar que a imensa maioria dos funcionários tem variações menor ou igual a zero.

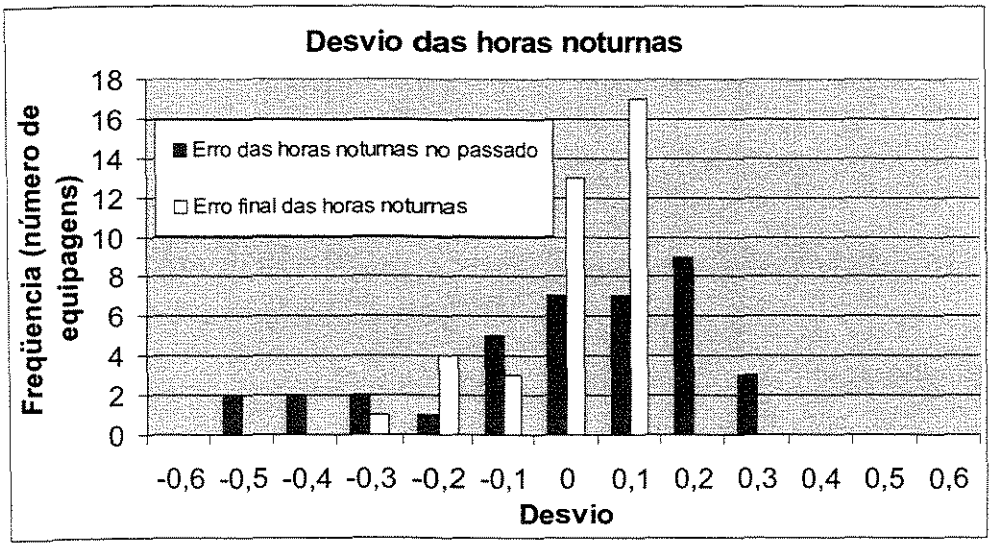

Figura 5.1 - Erro das horas noturnas

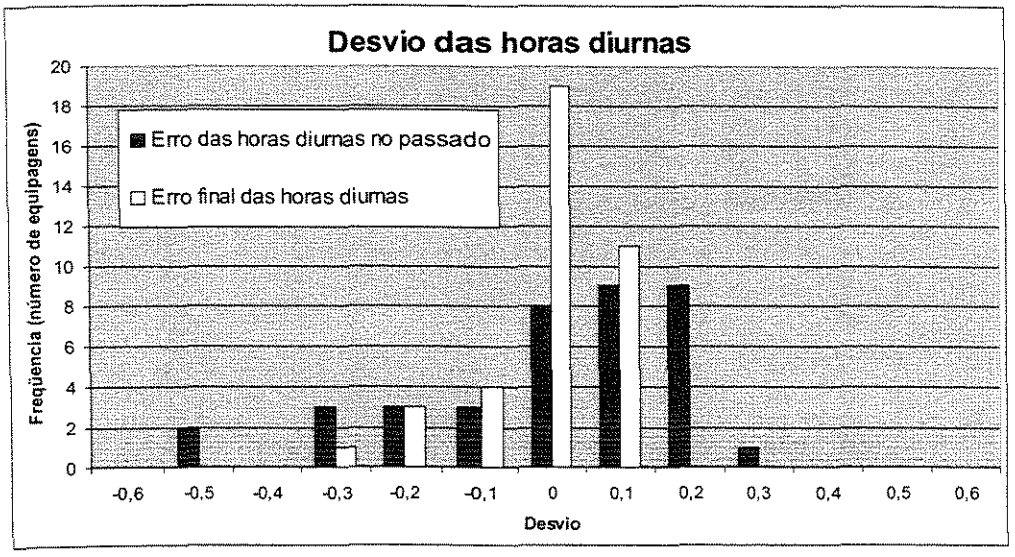

Figura 5.2 - Erro das horas diurnas

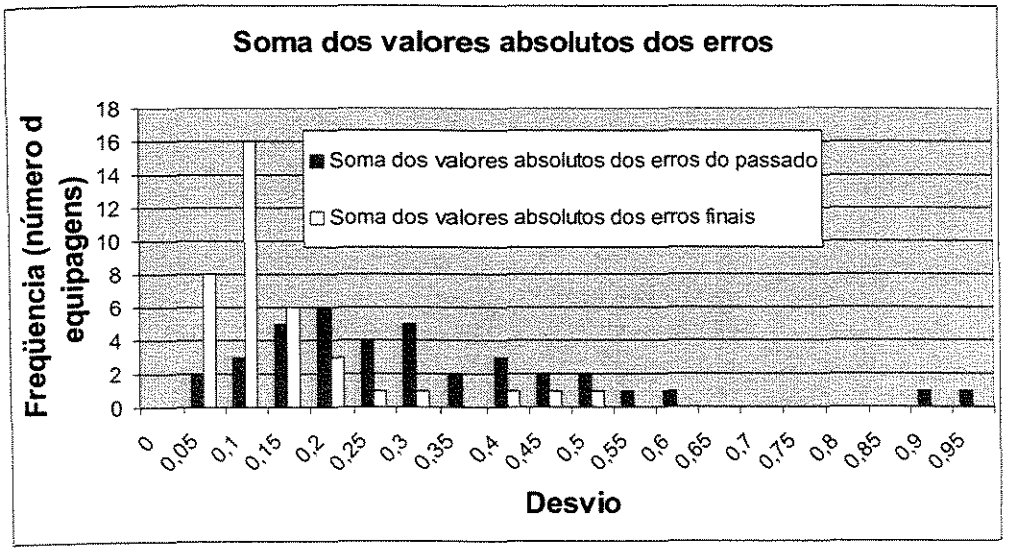

Figura 5.3 – Soma dos valores absolutos dos erros do passado

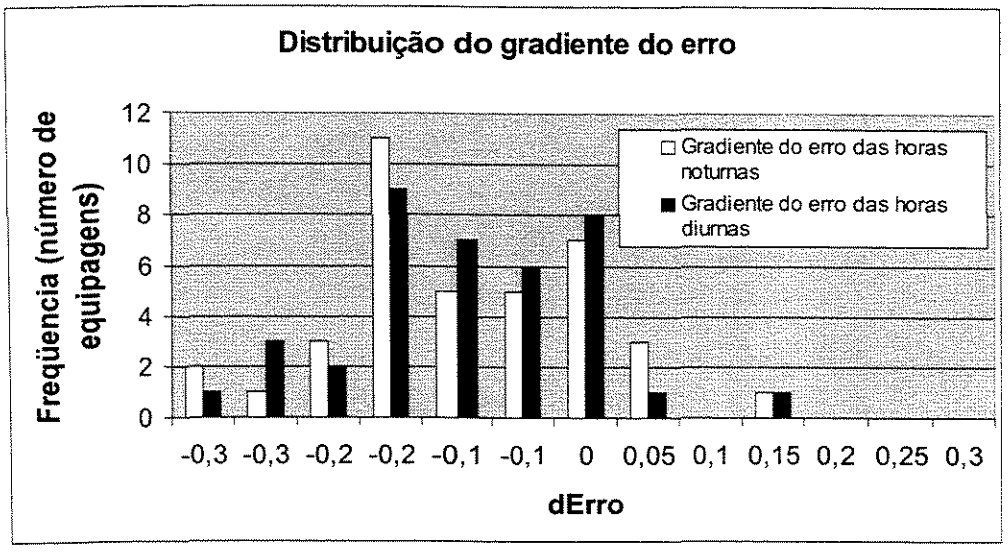

Figura 5.4 - Variação do erro das horas noturnas

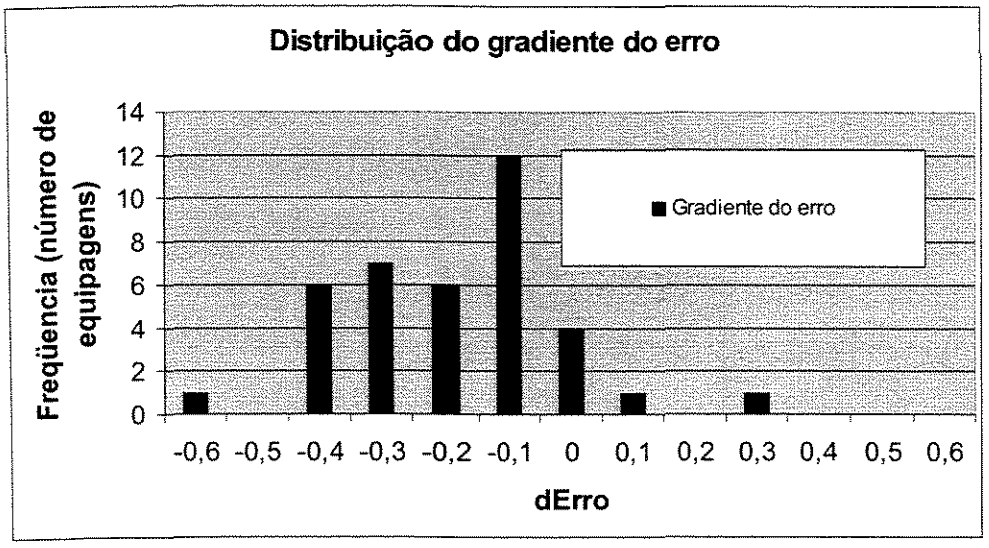

Figura 5.5 - Variação da soma do valor absoluto dos erros

Se o método do seqüencial de tarefas fosse utilizado neste caso, não haveria forma de compensar o passado uma vez que o seqüencial de tarefas resultante não teria mais do que 24 passos e portanto seria completamente executado por todos os funcionários em um mês. Assim todos eles teriam aproximadamente a mesma carga de trabalho, ou seja, não haveria modificações no perfil da distribuição do erro no passado (Figuras 5.1, 5.2).

Outra vantagem deste método com relação ao baseado em seqüenciais de tarefas é a minimização do impacto de más programações. Freqüentemente, devido à falta de pessoal ou ao excesso de tarefas, o programador é obrigado a programar seqüências de tarefas que tem segmentos

"apertados", ou seja, que exigem muito esforço dos equipagens mesmo que a princípio sejam realizáveis. Quando isto ocorre em um seqüencial de tarefas, aquele trecho crítico se repete todos os dias do período de programação e freqüentemente torna-se ponto de reclamação de todos as equipagens do destacamento. Em casos extremos pode até se tornar um ponto de ruptura onde as equipagens provocam atrasos, levando assim a um colapso da escala planejada. Como não há um padrão na escala gerada pelo método direto, os pontos críticos ocorrem de forma distribuída ao longo da escala e em horários diferentes. Assim eventuais problemas com relação à escala programada ficam mais restritas e mais fáceis de serem gerenciadas.

Para comparar os dois paradigmas, usando como critério a capacidade de compensar o passado, foram geradas escalas para um destacamento de 172 equipagens com dados reais, levando-se em consideração todas as extra-tarefas e pré-alocações. Para se resolver o problema de pré-alocação no paradigma cíclico, foram deixadas de fora do seqüencial ("na sobra") 4 equipes, que foram utilizadas posteriormente em permutações de tarefas. Os resultados dos desvios da carga de trabalho (erro) podem ser vistos na Figura 5.6 e na Figura 5.7. O passado utilizado neste exemplo foram as escalas *realizadas* de dois meses anteriores.

Nas Figuras 5.6 e 5.7 podemos observar que o paradigma cíclico compensa o passado de forma mais efetiva uma vez que a distribuição dos desvios ficou mais concentrada em torno do zero. Como já era esperado, o paradigma cíclico não consegue o mesmo resultado por duas razões: como o seqüencial de tarefas é balanceado (vide medida de entropia na função de avaliação do seqüencial na seção 2.4.1), as diferenças que podem ser utilizadas para compensar o passado são pequenas e nem sempre podem ser utilizadas com eficiência devido a compatibilidade necessária com a escala anterior das equipagens. Por último, mesmo quando isto pode ser realizado com eficiência, o fato de deixar equipagens fora do seqüencial para realizar permutações atrapalha a distribuição obtida, gerando "pontos fora da curva".

Conforme mencionado anteriormente, este tipo de comparação só faz sentido se o seqüencial utilizado para gerar a escala tiver mais de 30 passos. Quando isto não ocorre, todas as equipes executam praticamente o mesmo números de horas noturnas e diurnas durante a escala, impossibilitando qualquer tipo de compensação da carga de trabalho ocorrida em escalas anteriores (passado).

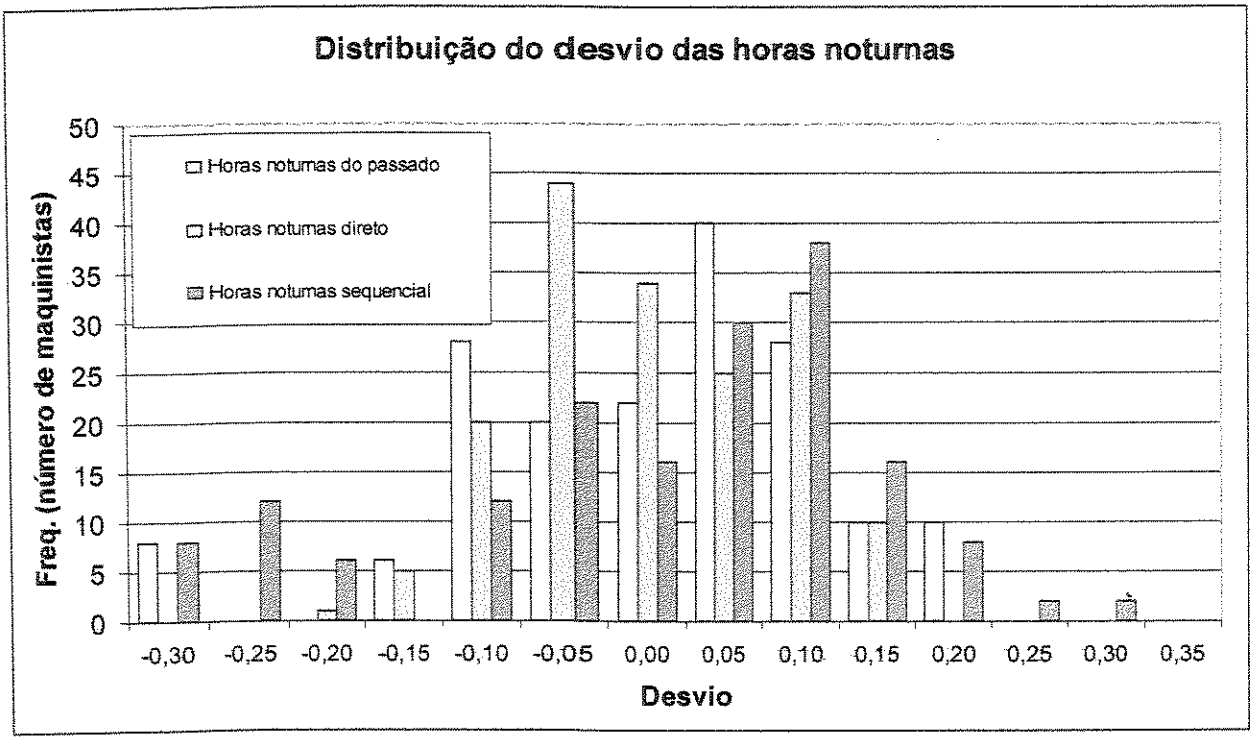

Figura 5.6 – Distribuição do desvio das horas noturnas

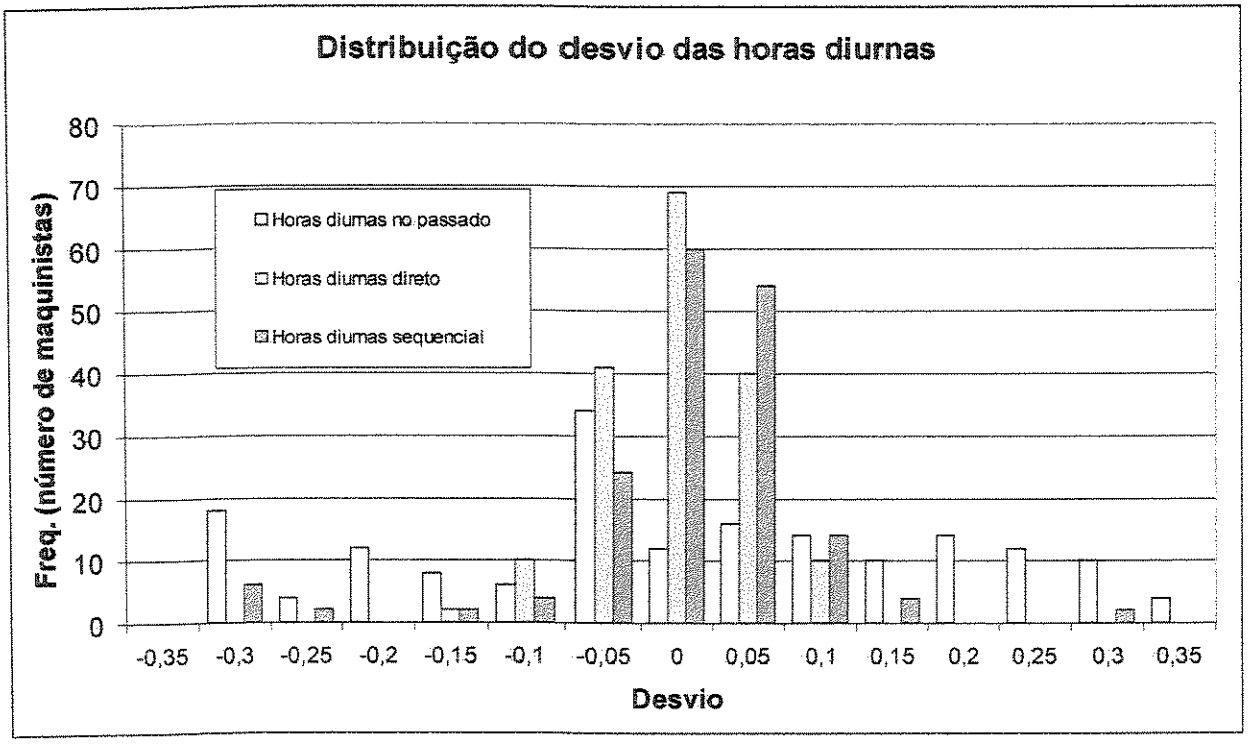

Figura 5.7 – Distribuição do desvio das horas diurnas

Até o momento em que este texto foi escrito este paradigma foi testado em 8 diferentes destacamentos com bons resultados. A geração da escala já prevendo as exceções e sem sobras[13] leva a um menor retrabalho durante a realização da mesma. Isto por sua vez leva a um aumento do índice de realização da escala, ou seja, a porcentagem de sucesso na realização. A utilização do passado na confecção das escalas e seu conseqüente efeito de compensação agradou às equipagens e aos especialistas.

Uma desvantagem do método direto é o fato de que ele não garante factibilidade e otimalidade. Outra desvantagem reside no fato de que ele garante os intervalos mínimos entre as tarefas mas não os intervalos máximos. Isto pode levar em alguns casos a folgas excessivamente longas, exigindo que o usuário efetue ajustes manuais para adequar a escala.

Uma desvantagem do método direto, quando comparado ao método seqüencial é a complexidade de análise da escala. Uma escala gerada através do seqüencial de tarefas pode ser completamente analisada através do seqüencial de tarefas. No método direto é necessário analisar a escala como um todo, tornando-se um processo penoso e complicado. Por outro lado, isto sugere o desenvolvimento de ferramentas automáticas de análise.

Apesar dos resultados promissores obtidos com o método direto neste trabalho, observa-se que este ainda é o primeiro passo em direção a métodos ainda mais sofisticados. Por exemplo, não foi utilizado um horizonte maior do que dois dias no método direto devido ao aumento substancial da complexidade do problema quando um terceiro dia é introduzido. Porém a introdução de um terceiro dia no horizonte de planejamento aumentaria a robustez dos algoritmos dado que 3 dias é um intervalo grande o suficiente para levar em consideração as conseqüências de uma atribuição. Isto ocorre porque uma tarefa dura no máximo 38 horas aproximadamente, com um intervalo entre tarefas médio de 16 horas, o que resulta numa duração máxima de aproximadamente 54 horas.

5.3.1 Considerações de ordem prática

Apesar de "estatisticamente eficiente" e de atender todas as especificações colocadas, ou seja, criar uma escala que obedece todas as restrições impostas, o método direto representa uma quebra de paradigma difícil de ser assimilada pelos especialistas das ferrovias. Além disto, por maior que tenha sido o esforço para tornar o sistema computacional simples e intuitivo, este método é menos transparente do que o método do seqüencial de tarefas. Assim ele se torna mais difícil de ser utilizado. Por todos estes motivos, alguns meses foram necessários para que os especialistas dessem início aos testes deste novo paradigma.

A princípio acreditava-se que a não previsibilidade da escala seria um obstáculo à utilização deste novo paradigma, uma vez que, quando o método cíclico é utilizado, mesmo sem ter acesso ao seqüencial de tarefas que deu origem a escala, os funcionários facilmente o obtém. Nestes casos eles podem prever horizontes além daqueles entregues na escala mensal. Isto porém só se mostrou problemático nos destacamentos em que a carga de trabalho, por outras razões que não o algoritmo de alocação, encontrava-se elevada. Nos demais destacamentos as equipagens perceberam os outros benefícios do novo paradigma e não criaram nenhum tipo de impedimento.

[13] Sem funcionários na sobra, (vide nota 10 na página 48)

6. Conclusões

Seguindo a tendência de outras áreas de pesquisa, os métodos baseados em inteligência computacional mostraram-se eficazes na resolução de problemas de alocação de recursos humanos, mais especificamente na geração de escalas de equipagens ferroviárias.

Neste trabalho foi dado um enfoque global ao problema de gerenciamento de equipagens. Assim as necessidades das ferrovias nacionais foram mais precisamente identificadas e as soluções encontradas na literatura reavaliadas. No intuito de melhor atender estas necessidades, foram desenvolvidos métodos de geração de escalas segundo dois paradigmas diferentes, sendo o método direto inovador pois, até onde temos conhecimento, não há trabalho similar no mesmo domínio de aplicação disponível na literatura.

Apesar de apresentar um desempenho promissor em termos computacionais e da qualidade das soluções, muito ainda pode ser realizado. Podemos destacar os seguintes trabalhos futuros para o paradigma do método direto:

- melhoria do método direto para utilização de um horizonte de 3 ou mais dias;
- pesquisa e desenvolvimento de métodos automáticos de análise;
- melhoria da interface de utilização do método direto;
- pesquisa e desenvolvimento de métodos de pré-otimização de agrupamento de equipes;
- refinamento das avaliações da escala, utilizando-se técnicas de preferência declarada;
- flexibilização de restrições.

Apesar de promissor, o método direto não visa excluir o método do seqüencial de tarefas. Neste caso os seguintes trabalhos futuros podem ser apontados:

- pesquisa e desenvolvimento para a elaboração de um sistema especialista visando a realização de pré-alocações em uma escala obtida por um seqüencial de tarefas através de operações de permutação com funcionários de sobra[14];
- transformar o algoritmo branch-and-bound de criação de seqüencial em um algoritmo distribuído com o uso de agentes autônomos colaborativos e redes de objetos;
- consideração de restrições flexíveis nos algoritmos de computação evolutiva.

[14] Equipagens que participam da escala mas não do seqüencial.

7. Referências bibliográficas

[Alb91] - J. S. Albus, *Outline for a Theory of Intelligence*, IEEE Transactions on System Man and Cybernetics, Vol. 21, No. 3, May/June 1991.

[Alb97] - J. S. Albus, *Why Now Semiotics? From Real-Time Control to Signs And Symbols*, Proceedings of the ISAS'97 - Inteligent Systems and Semiotics: A Learning Perspective - Gaithersburg, MD, USA - 22-25 September, 1997.

[Alf95] - J. Bailey, H. Alfares, Win Yuan Lin, *Optimization and heuristic models to integrate project test and manpower schedule*, Computers ind. Engng, 29(1-4):473-476, 1995

[Alf97] - H. K. Alfares, J. E. Bailey, *Integrated project task and manpower scheduling*, IIE Transactions, 29:711-717, 1997

[AlS96] - K. S. Al-Sultan, M. F. Hussain and J. S. Nizami, *"A Genetic Algorithm for the Set Covering Problem"*, JORS 47:702-709, 1996

[Ash92] - R. W. Ashford, R. C. Daniel, *Some lessons in solving practical integer programs*, Journal of the Operational Research Society, 43:425-433, 1992

[Aur94] - C. A. Lacerda, P. Geiger (editores), *Dicionário Aurélio Eletrônico v1.4*, Editora Nova Fronteira, Dezembro, 1994

[Ayk96] - T. Aykin, *Optimal shift scheduling with multiple break windows*, Management Science, 42(4):591-602, April 1996

[Bak79] - K. R. Baker, R. N. Burns, M. Carter, *Staff scheduling with day-off and workstretch constraints*, AIIE Transactions, 11(4):286-292

[Bal80] - E. Balas, A. Ho, *Set covering algorithms using cutting planes, heuristics and subgradient optimization: A computational study, Mathematical Programming Study*, 12:37-60, 1980

[Bal90] - N. Balakrishnan, R. T. Wong, *A network model for the rotating workforce scheduling problem*, Networks, 20:25-42, 1990

[Bal96] - E. Balas, M.C.Carrera, *A dynamic subgradient-based branch-and-bound procedure for set covering. Operations Research*, 44:875-890, 1996

[Bea91] - J. E. Beasley, *Lagrangean relaxation*. Em C. R. Reeves (editor), *Modern Heuristic Techniques for Combinatorial Problems*, pp.243-303. McGraw-Hill, London, 1995.

[Bea96] - J. E. Beasley, B. Cao, *A tree search algorithm for the crew scheduling problem*, European Journal of Operational Research, 94:517-526, 1996.

[Bea96b] - J. E. Beasley and P. C. Chu, *A genetic algorithm for the set covering problem*, European Journal of Operational Research 94:392-404, 1996

[Ber9X] - D. S. Bernstein, *A Student's Guide to Research*, IEEE Control Systems, pp.102-108

[Bod83] - L. Bodin, B. Golden, A. Assad, M. Ball. *Routing and Schedulling of Vehicles and Crews: The state of the art*, Computers and Operations Research, 10(2):163-211, 1983

[Bok80] - U. Bokinge, D. Hasselström, *Improved vehicle scheduling in public transport through systematic changes in the time-table*, European Journal of Operational Research, 5:388-395, 1980

[Bro76] - W. S. Brownell, J. M. Lowerre, *Scheduling of work forces required in continous operations under alternative labor polices*, Management Science, 22(5):597-605, 1976

[Bru85] - N. T. Bruvold, J. R. Evans, *Flexible Mixed-Integer Programming Formulations for production scheduling problems*, IIE Transactions, 17(1):2-7, 1985

[Bur85] - R. N. Burns, M. W. Carter, *Work force size and single shift schedules with variable demands*, Management Science, 31(5):599-607

[Bur87] - R. N. Burns, G. J. Koop, *A modular approach to optimal multiple-shift manpower scheduling*, Operations Research, 35(1):100-110

[Cap97] - A. Caprara, M. Fischetti, P. Toth, D. Vigo, *Algorithms for railway crew management*, Mathematical Programming, 79:125-141, 1997.

[Cap98] - A. Caprara, P. Toth, D. Vigo, M. Fischetti, *Modeling and solving the crew rostering problem*, Operations Research, 46(6):820-830, November-December, 1998

[Cap99]- A. Caprara, M. Fischetti, P. Toth, *A heuristic method for the set covering problem*. Operations Research, 47(5):730-743, September-October 1999

[Cer98] - S. Ceria, P. Nobili, A. Sassano, *A lagrangian based heuristic for large-scale set covering problems*, Mathematical Programming, 81:215-228, 1998

[Cle93] - R. Clement, A. Wren, *Greedy Genetic Algorithms, Optimising mutations and Bus Driver Scheduling*, 6th International Workshop on Computer Aided Scheduling of Public Transpotation, Lisboa, 1993

[Con97] - A. A. Constantino, *Otimização de escala de trabalho para condutores de trem: seqüenciamento de tarefas e alocação baseada em preferência declarada*, Tese de doutorado, Universidade Federal de Santa Catarina, 1997

[Day96] - P. R. Day, D. M. Ryan, *Flight Attendant Rostering for Short-Haul Airline Operations*, Operations Research, Septemper-October, pp649-661, 1997

[Emm85] - H. Emmons, *Work-force schedule with cyclic requirements and constraints on days off, weekends off and work stetch*, IIE Transactions, 17(1):8-16, 1985

[FerHP] - http://www.fs-on-line.com/index.html , Homepage da *Ferrovie dello Stato*

[Fre97] - R. Freling, *Models and Techniques for Integrating Vihicle and Crew Scheduling*, Ph.D. Thesis, Erasmus University, Rotterdam, Nederland, 1997

[Geo74] - A. M. Geoffrion, *Lagrangean relaxation for integer programming*, Mathematic programming studies, 2:82-114, 1974

[Glo97] - F. Glover and M. Laguna, *Tabu search*, Kluwer Academic Publishers, Norwell, Massachusetts, 1997

[Gol89] - D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989

[Gom98] - W. Pedrycz, F. Gomide, *An Introduction to Fuzzy Sets : Analysis and Design*, MIT Press Complex Adaptive Systems, 1998

[Gon97] - R. Gonçalves, *Computação Flexível em Simulação e Controle de Sistemas Não Lineares*, tese de mestrado, Departamento de Engenharia de Computação, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, 1997

[Gon98] - R. Gonçalves, R. Gudwin, F. Gomide, *Semiotic Oriented Autonomous Intelligent Systems Engineering*. Proceedings of the ISAS'98 Gaithersburg, MD, USA - Semptember, 1998

[Gon99] - R. Gonçalves, R. Gudwin, F. Gomide, *Emotions: a computational semiotics perspective*, Proceedings of the ISAS'99, Washington, USA, 1999.

[Gud96] - R. Gudwin, *Contributions to the Mathematical Study of Intelligent Systems* - Ph.D. Thesis - DCA-FEEC-UNICAMP - May, 1996. (in portuguese)

[Gud97a] - R. Gudwin e F. Gomide, *Computational Semiotics : An Approach for the Study of Intelligent Systems - Part I : Foundations*, Technical Report RT-DCA09 - DCA-FEEC-UNICAMP, 1997.

[Gud97b] - R. Gudwin and F. Gomide, *Computational Semiotics : An Approach for the Study of Intelligent Systems - Part II : Theory and Applications*, Technical Report RT-DCA09 - DCA-FEEC-UNICAMP, 1997.

[Gud99] - J. A. S. Guerrero, L. L. Suárez, R. R. Gudwin - *Análise da Importância de Parâmetros em um Algoritmo Genético por meio de sua Aplicação no Aprendizado de uma Rede Neural* - Anais do XIX Congresso Nacional da Sociedade Brasileira de Computação, Rio de Janeiro, RJ, Brasil, 1999. v.4. p.423-435

[Hol76] - H. E. Miller, W. P. Pierskalla, G. J. Rath, *Nurse scheduling using mathematical programming*, Operations Research, 24(5):857-870, 1976

[Ken95] - J. R. Kennedy Jr., *Solving Unweighted and Weighted Bipartite Matching Problems in Theory and Practice*, Ph.D. dissertation, Department of Computer Science of Stanford University.

[Ken95b] - J. V. Goldberg, J. Robert Kennedy Jr., *An efficient cost scaling algorithm for the assignment problem*, Technical report, Computer Science Department, Stanford University, E-mail robert@cs.stanford.edu to get a copy

[Kho94] - C. M. Khoong, A. C. Lau, L. W. Chew, *Automated Manpower Rostering: Techiniques and Experience*. International Transactions in Operational Research, 1(3):353-361, 1994

[Koo88] - G. J. Koop, *Multiple shift workforce lower bounds*, Management Science 34(10):1221-1229

[Lad96] - S. R. Ladd, *Genetic Algorithms*. M&T Books, 1996

[Lan92] - C. G. Langton, ed. *Artificial Life II*. Addison-Wesley, 1992

[Lan94] - C. G. Langton, ed. *Artificial Life III*. Addison-Wesley, 1994

[Lau96] - H. C. Lau, *Combinatorial approaches for hard problems in manpower scheduling*, Journal of the Operations Research, 39(1):88-98, 1996

[Lau96] - H. C. Lau, *On the complexity of manpower shift scheduling*, Computers Operational Research, 23:93-102, 1996

[Lee94] - C. C. Lee, *Fuzzy Logic in Control Systems, Fuzzy Logic Controller - Part I*, IEEE Transactions on System Man and Cybernetics, 20(2):406-418, 1994

[Lug98] - G. F. Luger, William A. Stubblefiel, *Artificial Intelligence : Structures and Strategies for Complex Problem Solving*, Addison-Wesley Pub Co, 1998

[Lui00] - L. A. R. Domínguez, *Programacao de Escalas usando Algoritmos Evolutivos. Aplicacao em Empresas de Transporte Ferroviario*, Tese de mestrado, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, 2000

[Lui00b] - L. A. R. Dominguez, *Railway Crew Scheduling Using a Fuzzy-Evolutionary Strategy*, 8th Information Processing and Management of Uncertainty in Knowledge-Based Systems Conference (IPMU 2000) Madrid SPAIN, July 3-7, 2000

[Mib93] - S. Uckun, S. Bagchi, K. Kawamura and Y. Mibaye, *Managing Genetic Search in Job Shop Scheduling, IEEE Expert*, pp. 15-23, October, 1993

[Mur98] - H. Ishibuchi and T. Murata, *A Multi-Objective Genetic Local Search Algorithm and Its Applications to Flowshop Scheduling. IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews*, Vol. 28, No. 3, August, 1998

[Nil80] - N. J. Nilsson , *Artificial Intelligence*, Morgan Kaufman Publishers,1980

[Niz96] - A. S. Al-Sutan, M. F. Hussain, J. S. Nizami, *A genetic algorithm for the set covering problem*, Journal of the Operational Research Society, 47:702-709

[Not95] - W. Noth, *Handbook of Semiotics*, Indiana Univ. Press, 1995

[Orl93] - R. K. Ahuja, T. L. Magnanti, J. B. Orlin, *Network Flows : Theory, Algorithms, and Applications*, Prentice Hall, fevereiro, 1993

[Pan98] - A. J. Mason, D. M. Ryan, D. M. Panton, *Integrated simulation, heuristic and optimisation approaches to staff scheduling*, Operations Research 46(2):161-175, 1998

[Pet95] - B. Paechter, H. Luchian H. and M. Petriuc, *Two Solutions to the General Timetable Problem, Tech Report No. RR-95-2.* Napier University, Computer Studies, EH14 1DJ, 1995

[Pic97] - R. Picard, *Affective Computing*, The MIT Press, 1997

[Por98] - H. Ramalhinho Lourenço, José Pinto Paixão, Rita Portugal, *Metaheuristics for the bus-driver scheduling problem*, Economic Working Papers Series, no. 304, Universitat Pompeu Fabra, 1998

[Por98b] - R. Portugal, *Metaheuristics for the bus-driver scheduling problem*, Tese de mestrado, Faculdade de Ciências da Universidade de Lisboa, Portugal, 1998

[Rav93] - R. K. Ahuja, T. L. Magnanti, J. B. Orlin, *Network flows : theory, algorithms, and applications*, Upper Saddle River : Prentice-Hall, 1993.

[Rus95] - S. J. Russell, P. Norvig, *Artificial Intelligence : A Modern Approach*, Prentice Hall Series in Artificial Intelligence, 1995

[Smi90] - A. J. Smith, *The Task of the Referee*, IEEE Computer, April, 65-71, 1990

[Tan95] - J. Tanomaru, "Planejamento de Mão-de-Obra via um Algoritmo Genético", *Anais do II Congreso de Redes Neurais*, Curitiba, Outubro, 1995

[Ter92] - T. Terano, K. Asai, M. Sugeno - *Fuzzy Systems Theory and Its Applications* - Academic Press - 1992

[Tho93] - T. E. Morton, D. W. Pentico, *Heuristic Scheduling Systems*, Wiley Series in Engineering & Technology Management, John Wiley & Sons, Inc., New York, 1993.

[Tie82] - J. M. Tien, A. Kamiyama, *On manpower scheduling algorithms*, Society for Industrial and Applied Mathematics, 24(3):275-287

[Tot93] - M. Dell'Amico, M. Fischetti, P. Toth, *Heuristic Algorithm for the multiple depot vehicle scheduling problem*, Management Science, 39:115-125, 1993

[Van97] - P. H. Vance, C. Barnhart, E. L. Johnson, G. L. Nemhauser, *Airline crew scheduling: a new formulation and decomposition algorithm*, Operations Research, 45(2), March-April 1997

[War76] - D. M. Warner, *Scheduling nursing personnel according to nursing preference: A mathematical programming approach*, Operations Research, 24(5):842-856, 1976

[Win92] - P. H. Winston, *Artificial Intelligence - 3th edition*, Addison-Wesley Pub Co, 1992

[Woo95] - M. Wooldridge, N. Jennings, *Intelligent Agents: Theory and Practice*, Cambridge University Press, 1995.

[Wre95] - A. Wren and D. O. Wren, *A genetic algorithm for public transport driver scheduling*, Computers and Operations Research 22:101-110, 1995

[Zad94] - A. L. Zadeh, *Soft Computing and Fuzzy Logic*, IEEE Software, 11(6), 1994, pp 48-56

[Zup99] - B. Filipec and D. Zupanic, *Near-Optimal of Line Production with an Evolutionary Algorithm, Proceedings of the Congress on Evolutionary Computation-CEC'99*, Washington D.C., U.S.A., Vol. 2, pp. 1237-1244, July 1999

8. Anexos

8.1 Propriedade da comutatividade rotacional da entropia

Para um determinado seqüencial de tarefas $S = [t_1,t_2,\ldots t_n]$, onde $t_1,t_2,\ldots t_n$ são tarefas de tempo $\tau_1, \tau_2, \ldots, \tau_n$, respectivamente, define-se como uma escala ψ rotacionada de S a escala $S^{\varphi} = [T_{(1+\varphi-1)\%n+1}, T_{(2+\varphi-1)\%n+1}, \ldots T_{(n+\varphi-1)\%n+1}]$. Por exemplo:

- $S^0 = S$
- $S^1=[t_2, t_3, \ldots, t_n, t_1]$
- $S^2=[t_3, t_4, \ldots, t_1, t_2]$

Deseja-se provar que *Entropia*(S) = *Entropia*(S^{Ψ}) para qualquer valor de Ψ.

Prova:

A partir da equação (2.16) definimos *Entropia*(S^{Ψ}) como:

$$Entropia\left(S^{\Psi}\right) = \frac{\sum_{ij} \delta_{ij}^{\Psi}}{n^2} = \frac{\sum_{i}\sum_{j} \delta_{ij}^{\Psi}}{n^2} \tag{8.1}$$

$$\delta_{ij}^{\Psi} = abs\left(\frac{\xi_{ij}^{\Psi} - med_i^{\Psi}}{med_i^{\Psi}}\right) \tag{8.2}$$

$$med_j^{\Psi} = \frac{\sum_{i=1}^{n} \xi_{ij}^{\Psi}}{n} \tag{8.3}$$

$$\xi_{ij}^{\Psi} = \sum_{k=j-1}^{i-1+j-1} x^{\Psi}{}_{(k\%n)+1} \tag{8.4}$$

$$x_i^{\Psi} = \begin{cases} \tau_{(i+\psi-1)\%n+1}, & \text{se i} \le \text{m} \\ \hat{x}, & \text{se i} > \text{m} \end{cases} \tag{8.5}$$

Para φ=0,

Como, por definição $S^0 = S$, então *Entropia*(S^0) = *Entropia*(S).

Supondo que *Entropia*(S^{n-1}) = *Entropia*(S) e usando $\Psi = n$, temos que:

$$med_1{}^n = \frac{x_1^n + x_2^n + \cdots + x_n^n}{n} = \frac{x_n + x_1 + \cdots + x_{n-1}}{n} =$$
$$\frac{x_{n-1} + x_n + \cdots + x_{n-2}}{n} = \frac{x_1^{n-1} + x_2^{n-1} + \cdots + x_n^{n-1}}{n} = med_1^{n-1} \tag{8.6}$$

$$med_2{}^n = \frac{\left(x_1^n + x_2^n\right) + \left(x_2^n + x_3^n\right) + \cdots + \left(x_n^n + x_1^n\right)}{n} = \frac{2\left(x_1^n + x_2^n + \cdots + x_n^n\right)}{n} =$$
$$\frac{2\left(x_n + x_1 + \cdots + x_{n-1}\right)}{n} = \frac{2\left(x_{n-1} + x_n + \cdots + x_{n-2}\right)}{n} = \frac{2\left(x_1^{n-1} + x_2^{n-1} + \cdots + x_n^{n-1}\right)}{n} = \tag{8.7}$$
$$\frac{\left(x_1^{n-1} + x_2^{n-1}\right) + \left(x_2^{n-1} + x_3^{n-1}\right) + \cdots + \left(x_n^{n-1} + x_1^{n-1}\right)}{n} = med_2^{n-1}$$

$$(\ldots)$$

$$med_n{}^n = \frac{n\left(x_1^n + x_2^n + \cdots + x_n^n\right)}{n} = \frac{n\left(x_n + x_1 + \cdots + x_{n-1}\right)}{n} =$$
$$\frac{n\left(x_{n-1} + x_n + \cdots + x_{n-2}\right)}{n} = \frac{n\left(x_1^{n-1} + x_2^{n-1} + \cdots + x_n^{n-1}\right)}{n} = med_n^{n-1} \tag{8.8}$$

De (8.1) e (8.3) vem que, para todo i,:

$$\sum_j \delta_{ij}{}^{\psi} = \sum_j \delta_{ij}{}^{n} = \frac{abs\left(\xi_{i1}{}^n - med_i^n\right) + \ldots + abs\left(\xi_{in}{}^n - med_i^n\right)}{med_i^n} \tag{8.9}$$

Aplicando o mesmo raciocínio utilizado em (8.6), (8.7) e (8.8), temos que:

$$\sum_j \delta_{ij}{}^{n} = \frac{abs\left(\xi_{in} - med_i^n\right) + \ldots + abs\left(\xi_{i,n-1} - med_i^n\right)}{med_i^n} \tag{8.10}$$

Reordenando as parcelas e aplicando as equações (8.6), (8.7) e (8.8) temos que:

$$\sum_j \delta_{ij}{}^{n} = \frac{abs\left(\xi_{i,n-1} - med_i^{n-1}\right) + \ldots + abs\left(\xi_{i,n-2} - med_i^{n-1}\right)}{med_i^{n-1}} \tag{8.11}$$

e que:

$$\sum_j \delta_{ij}{}^{n} = \frac{abs\left(\xi_{i1}{}^{n-1} - med_i^{n-1}\right) + \ldots + abs\left(\xi_{in}{}^{n-1} - med_i^{n-1}\right)}{med_i^{n-1}} = \sum_j \delta_{ij}{}^{n-1} \tag{8.12}$$

Portanto:

$$Entropia\left(S^n\right) = \frac{\sum_{ij} \delta_{ij}{}^{n}}{n^2} = \frac{\sum_i \sum_j \delta_{ij}{}^{n}}{n^2} = \frac{\sum_i \sum_j \delta_{ij}{}^{n-1}}{n^2} = Entropia\left(S^{n-1}\right) \tag{8.13}$$

que pelo princípio da indução finita, conclui a demonstração.

8.2 Propriedade de limitante inferior da entropia

Seja um conjunto $T_n = \{t_1, t_2, \ldots, t_n\}$ de n tarefas, definimos como sendo um seqüencial m-parcial de n passos $S_{m/n} = [t_{x_1}, t_{x_2}, \ldots, t_{x_m}, \_]$, onde '_' designa *m-p* posições em aberto no seqüencial.

Deseja-se provar que $Entropia(S_{m/n}) \leq Entropia(S)$, onde S é um seqüencial completo construído a partir do seqüencial $S_{m/n}$.

Prova:

Quando a matriz Ξ de um seqüencial incompleto é construída (2.19)/(8.20), para cada linha, os elementos correspondentes às tarefas faltantes são preenchidos com a média (da linha correspondente da matriz Ξ). Assim, para estes valores, o resultado da equação (2.17)/(8.16) é sempre zero. Qualquer quer seja a tarefa incluída no seqüencial parcial para torna-lo completo, irá resultar valores maiores ou iguais a zero na equação (2.17)/(8.16). Assim a soma de todos os valores fornecidos pela equação (2.17)/(8.16) (equação (2.16)/(8.14)) para uma escala completa será sempre maior ou igual à mesma soma da escala parcial.

$$Entropia(S) = \frac{\sum_{ij} \delta_{ij}}{n^2} \qquad (8.14)$$

$$\xi_{ij} = \sum_{k=j-1}^{i-1+j-1} x_{(k\%n)+1} \qquad (8.15)$$

$$\delta_{ij} = abs\left(\frac{\xi_{ij} - med_i}{med_i}\right) \qquad (8.16)$$

$$x_i = \begin{cases} \tau_i, & \text{se } i \leq m \\ \hat{x}, & \text{se } i > m \end{cases} \qquad (8.17)$$

$$med_j = \frac{\sum_{i=1}^{n} \xi_{ij}}{n} \qquad (8.18)$$

$$\hat{x} = \frac{\sum_{i=1}^{m} \tau_i}{m} \qquad (8.19)$$

$$\Xi = \begin{bmatrix} \xi_{1,1} & \cdots & \xi_{1,n} \\ \vdots & \ddots & \vdots \\ \xi_{n,1} & \cdots & \xi_{n,n} \end{bmatrix} \qquad (8.20)$$

8.3 Notação utilizada nos algoritmos

Os algoritmos são apresentados neste trabalho em um pseudo-código bastante simples, semelhante ao Pascal e C++ (e Java). Neste pseudo-código as seguintes palavras reservadas são utilizadas:

Nome	Significado	Sintaxe
BEGIN END	Inicio e fim de uma sentença ("*statement*")	BEGIN sentença END
Procedure	Indica o início de um procedimento, método ou função	Procedure NONE(PARAMETROS) BEGIN sentença END

IF THEN
ELSEIF
ELSE
ENDIF
	Execução condicional de sentenças	IF condição THEN sentença ELSEIF condicao THEN sentença ELSE sentença ENDIF

Os seguintes operadores são utilizados:

Nome	Significado	Sintaxe
: =	Atribuição	a : = b
.	Definição de escopo	Se a é uma variável que contém a ênupla (b,c,d,e), a.b é o valor do primeiro elemento da ênupla contida pela variável a.
|	Concatenação de listas	A={1,2,3}, B={4,5,6} A | B={1,2,3,4,5,6}
∅	Lista vazia	a:= ∅
=,!=,<=,>=,<,>	Testes	IF (a=b) THEN ... ELSE ... ENDIF
-,+,/,*	Operadores matemáticos	2*4=8
-	Operador de exclusão de um elemento de uma lista	A={1,2,3,4,5} A-3 = {1,2,4,5}
//	Comentário de linha	// Esta linha é um comentário
/* */	Início e término de um comentário longo	/* Este é um comentário longo */
%	Operador matemático que resulta no resto de uma divisão de inteiros	5%2 = 1
| |	Operador que resulta no número de elementos de um conjunto	|{a,b,c,d}| = 4