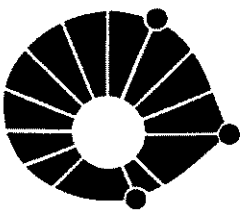
Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Engenharia de Computação e Automação Industrial



Programação de Escalas usando Algoritmos Evolutivos : Aplicação em Empresas de Transporte Ferroviário.

Luis Alberto Ramírez Domínguez

Orientador: Prof. Dr. Fernando Antonio Campos Gomide

Dissertação de Mestrado
Campinas – SP – Brasil
2000

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Engenharia de Computação e Automação Industrial

Programação de Escalas usando Algoritmos Evolutivos : Aplicação em Empresas de Transporte Ferroviário.

Luis Alberto Ramírez Domínguez

Orientador: Prof. Dr. Fernando Antonio Campos Gomide

Banca examinadora:

Prof. Dr Fernando Antonio Campos Gomide
DCA/FEEC/UNICAMP

Prof. Dr. Lúcia Valéria Ramos de Arruda
CEFET/Curitiba

Prof. Dr. Fernando José Von Zuben
DCA/FEEC/UNICAMP

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação (FEEC) da Universidade Estadual de Campinas (UNICAMP), como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Elétrica.

Área de Concentração: Automação

Dissertação de Mestrado
Campinas – SP – Brasil
2000

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

R145p

Ramírez Domínguez, Luis Alberto
Programação de escalas usando algoritmos evolutivos: aplicação em empresas de transporte ferroviário / Luis Alberto Ramírez Domínguez.-- Campinas, SP: [s.n.], 2000.

Orientador: Fernando Antonio Campos Gomide.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Administração da produção. 2. Algoritmos genéticos. 3. Otimização combinatória. I. Gomide, Fernando Antonio Campos. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Aos meus pais, meus avós, minha irmã, minha sobrinha, a todos meus familiares e a todas aquelas pessoas que souberam me dar um carinho familiar.

Agradecimentos

Agradeço ao Professor Fernando Gomide pela constante ajuda, pelas valiosas sugestões, pelo contínuo apoio e pelo exemplo de ensino, aspectos básicos e essenciais que integraram-se para a culminação e sucesso deste trabalho.

Aos meus pais Luis e Noris, que com todo amor, dedicação e exemplo souberam me orientar pelos caminhos do esforço e o sacrifício, lutando sempre pelo futuro, nas tentativas de reverter com sucesso os momentos difíceis da nossa existência.

À minha querida colega, companheira, amiga e irmã, Lizet pela ajuda, paciência e carinho, mostrados e demonstrados durante tantos anos, ajudando sempre a manter esse lindo tesouro chamado amizade.

Aos meus amigos Luisa, Leo, Gisel, Elizabeth, Beatriz, Miguel, Julio, Adrian e muitos outros, que apesar da distância que hoje nos separa, estiveram e estarão sempre juntos em pensamento me desejando o melhor.

Ao José, Franklin, Maria Eugenia, Lisyenia, Electo, Zaida, Vicente, Celeste, Wilson, Marta Inés, Luis Mariano, Sahudy, Eduardo, Daynet, Juan, Anabell, Raúl, Ania que juntos temos integrado uma grande comunidade cubana a qual considero uma alegre e linda família.

Aos meus amigos Cristina, Viviane, Daniela, Thaís, Ester, Cida, Rafael, Verena, Daniela Pereira, Alessandra, Christian, Dino, Daniela Soárez, Leandro, Míriam, Paulo, Ellen, Raquel e muitos outros pela ajuda e carinho que ao longo de todo este tempo têm me dado, incondicionalmente, me envolvendo e me tornando mais um grande admirador do Brasil.

Às pessoas especiais como Rosa, Isolina e Luisa por terem me dado apoio, ajuda, força e sobretudo um grande carinho de mãe que jamais esquecerei.

Ao professor Claudio Cabezas que ao longo da minha formação, soube me dar o estímulo inicial no difícil caminho da engenharia.

Aos professores Fernando Von Zuben e Ricardo Gudwin pelos conhecimentos transmitidos ao longo dos cursos e aos colegas Rodrigo Gonçalves e Pedro Chávez, pelas oportunas sugestões e contribuições.

À Magali, Miguel, Rocio, Pedrito, Charlis e a todos os amigos da Unicamp pela convivência do dia-a-dia e pela maneira tão simples de oferecer ajuda e amizade.

Ao CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico pelo suporte financeiro (bolsa de estudos), permitindo a minha integridade no desenvolvimento deste trabalho.

Resumo

Este trabalho descreve uma nova ferramenta computacional para resolver um problema de programação de escalas encontrado no Sistema de Transporte Ferroviário Brasileiro. Um procedimento de busca evolutiva é usado na procura das melhores seqüências de trabalho, usando para isso critérios práticos que permitam um balanço aceitável no sentido econômico e social. A função de avaliação das soluções é baseada em duas metodologias de análise. A primeira usa um método de soma ponderada e a outra baseia-se em conceitos da teoria de conjuntos *fuzzy*. O algoritmo proposto foi testado usando dados reais fornecidos por uma empresa ferroviária. As escalas geradas pelo algoritmo evolutivo apresentaram resultados qualitativamente superiores quando comparados com os resultados produzidos por técnicas atualmente em uso.

Palavras chaves: Programação de Escalas, Algoritmos Evolutivos, Otimização Multi-objetivos.

Abstract

This work describes a new computational tool to solve a crew-scheduling problem common in the Brazilian Railway Transport Systems. An evolutionary search procedure is used to generate the preferred work sequences with respect to evaluating criteria. The aim is to obtain an acceptable balance between human and economic aspects. The fitness function is based on two methodologies. The first one uses a weighting sum scheme to aggregate the objectives whereas the second uses fuzzy set theory for the same purpose. The proposed algorithm was tested with actual data provided by a railway company. The schedules generated by the evolutionary algorithm are qualitatively better than those used in practice.

Keywords: Crew-scheduling, Evolutionary Algorithms, Multi-objective Optimization.

CONTEÚDO

Lista de Figuras

Lista de Tabelas

Capítulo1

Introdução

O presente trabalho consiste no estudo, formalização e implementação de uma nova metodologia de programação de escalas de trabalho, aplicando recentes técnicas de computação evolutiva, dentre elas as Estratégias Evolutivas, em combinação com algumas heurísticas indispensáveis para complementar o processo de busca de soluções factíveis.

Neste capítulo introdutório, abordam-se brevemente aspectos gerais que envolvem os problemas de programação de escalas encontrados em vários setores econômicos.

A seguir, apresentam-se os principais objetivos e motivações que levaram ao desenvolvimento da ferramenta implementada e, por último, apresenta-se uma breve descrição da forma estrutural do texto como base para um bom entendimento e acompanhamento do trabalho.

1.1.- Esboço geral sobre programação de escalas

Elaborar uma programação de escalas é um procedimento extremamente difícil e está presente em muitos setores da economia. Uma estrutura genérica para esta classe de problemas é mostrada na Figura 1.1.

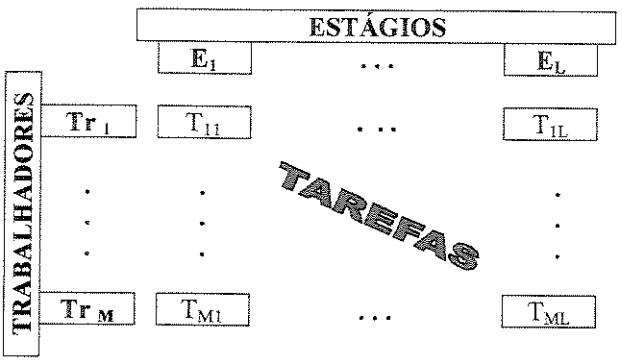


Figura 1.1. Estrutura básica de um problema de programação de escalas.

Segundo esta representação gráfica, cada trabalhador $Tr_{i\ \{i=1,...M\}}$, deve realizar uma tarefa $T_{j\ \{j=1,...N\}}$, em um estágio $E_{k\ \{k=1,...L\}}$. Cada estágio pode por exemplo, ser considerado como um dia de trabalho. Os resultados são seqüências $\{T_{ik}\}$ de tarefas, i=1,...M, k=1,...L, com $T_{ik} \in \{T_1, ..., T_N\}$. O problema associado é chamado de programação de escalas.

Além disso, dependendo do problema, uma série de limitações ou restrições podem ser incorporadas, algumas delas com caráter obrigatório, outras com certa flexibilidade com respeito à regulamentação exigida pelos estatutos empresariais.

O processo de geração de cada uma destas seqüências (escala), onde cada alocação de uma escala depende de um grande número de condições e restrições, induz um problema de otimização combinatória e multimodal *NP-hard*. A indústria tem hoje em dia a necessidade de desenvolver métodos eficientes para sua solução, como alternativa para metodologias antigas e ineficientes.

Presentemente, muitas empresas não dispõem de ferramentas computacionais para a obtenção destes programas de trabalho. Nestas instituições, cabe a um pessoal qualificado e experiente realizar este árduo trabalho que, consequentemente, se converte em uma metodologia de processamento manual e acarreta certas deficiências nos resultados produzidos, além da demora na geração da programação desejada.

Na medida que estes problemas vão se tornando cada vez mais complexos, é preciso adaptar ou desenvolver novos métodos de busca de soluções factíveis, que possam determinar um aumento da produtividade.

Uma das principais causas da elevada complexidade destes problemas encontra-se nos muitos requisitos que precisam ser satisfeitos para atingir as metas propostas, às vezes com limitação de recursos.

Planejar neste sentido torna-se difícil, principalmente naqueles setores em que as operações são 24 horas por dia, 7 dias por semana, com várias rotações no dia e onde qualquer mudança de horário, seja para outro turno de trabalho ou para outro dia, pode não ser facilmente tolerada.

Em organizações onde existem turnos irregulares de horários de trabalho, precisa-se enfrentar o problema de designar a escala para cada trabalhador de modo a satisfazer a demanda prevista de trabalho a um custo mínimo e levando em consideração todas as limitações inerentes, além de critérios adicionais que surgem durante a dinâmica de execução das seqüências programadas.

Setores como saúde, educação, transporte entre outros, não escapam a este problema. Neste aspecto, a literatura reporta aplicações em hospitais, universidades, empresas de transporte urbano e muitas outras instituições, que precisam de uma organização eficaz dos turnos rotativos de trabalho [16, 24, 23, 26, 1, 10, 14].

Geralmente, um escalonamento envolve restrições do tipo empresarial (exigências das metas de produção, entre outras) e restrições do tipo social (direitos dos trabalhadores). A relação causal, entre os requisitos da empresa e as restrições de caráter social, define um comportamento em conflito. Contudo, precisa-se trabalhar em um ponto de operação possível, com um custo do trabalho aceitável e com a devida satisfação humana dos trabalhadores envolvidos, Figura 1.2.

Obter o ponto de operação preferencial de trabalho não tem sido tarefa simples e muitos já são os trabalhos encaminhados com este objetivo. Apesar dos esforços realizados, ainda existem dificuldades na procura de boas soluções quando os problemas envolvem muitas variáveis e restrições.

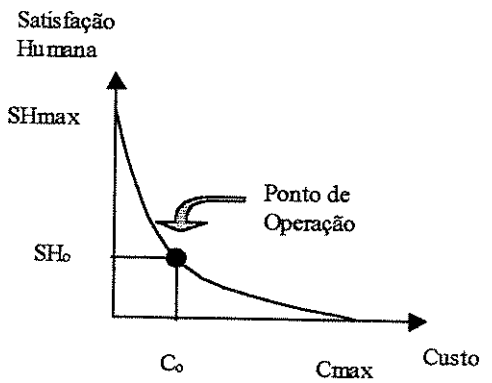


Figura 1.2. Relação aproximada entre o custo de trabalho e a satisfação humana.

Dadas estas circunstâncias, atribuímo-nos a tarefa de desenvolver uma metodologia que permitirá contribuir para solucionar um problema de geração de escalas encontrado em empresas de transporte ferroviário brasileiro, aplicando procedimentos conhecidos na literatura como metaheurísticas, que ajudam no processo de busca de soluções satisfatórias, segundo os critérios de avaliação utilizados.

1.2.- Formulação do problema

Geralmente, um problema de programação de escalas (escalonamento), de forma similar ao "*Timetable Problem*" [23, 26] e aos já bem conhecidos "*Job Shop Scheduling*" [33, 34] e "*Flow Shop Scheduling*" [17], consiste em alocar um conjunto de tarefas a um determinado número de trabalhadores, considerando um conjunto de restrições a serem satisfeitas.

O problema de interesse assume uma estrutura similar à apresentada na Figura 1.1, mas possui algumas caraterísticas que limitam alguns conceitos e metodologias básicas de solução, comumente usados nos problemas de escalonamento de força de trabalho reportados na literatura.

Neste caso, estaremos nos referindo ao clássico problema de planejamento tático, também conhecido como programação de escalas de equipagens nas empresas ferroviárias do Brasil.

A realização do planejamento tático consiste em alocar as tarefas para cada trabalhador (ou equipe de trabalho) durante 24 horas por dia, em períodos mensais. Além da análise mensal, neste trabalho será incorporada uma análise adicional, referente a um determinado período antecedente (programa ou programas anteriores), como base histórica para a realização dos programas a serem analisados.

Para obter soluções factíveis, todas as tarefas devem ser alocadas de tal forma que possam ser satisfeitas uma série de restrições (legislação, regulamentos e estatutos empresariais) e usando alguns critérios de avaliação que permitam estabelecer uma divisão justa de trabalho com respeito aos horários planejados.

Em problemas com estas características ou com características similares [9, 20, 32], muitos têm sido os esforços na procura de soluções utilizando técnicas heurísticas ou métodos convencionais, mas apesar desta preferência, as limitações antes mencionadas desaconselham o uso destas estratégias, que geralmente tendem a fornecer ótimos locais.

Estes elementos, junto com os resultados proporcionados pelas técnicas emergentes de computação evolutiva, motivou este trabalho, em particular quanto à aplicação de estratégias evolutivas, cujos resultados já são promissores tanto na teoria como na prática. Atualmente já apresentam soluções eficientes no sentido operacional [12, 22].

1.3.- Motivação e relevância do trabalho

A motivação deste trabalho consiste principalmente na impossibilidade de se obter soluções satisfatórias para problemas de programação de escalas, usando métodos polinomiais convencionais ou processamentos manuais. Isto repercute em esferas de grande importância econômica e social, e motiva o desenvolvimento de métodos alternativos.

Baseado nas limitações mencionadas e analisando algumas inconveniências encontradas no histórico de programações realizadas por empresas, encontramos o incentivo para levar a cabo o desenvolvimento deste projeto.

Como novidade, apresentamos a implementação de um algoritmo capaz de obter um escalonamento aceitável sob vários aspectos. Um primeiro aspecto é a utilização de informação anterior (análise do passado) para gerar uma nova programação, o que

ajudará a efetuar um balanceamento de carga de trabalho para cada equipe, em períodos maiores pré-definidos. A idéia de analisar o passado como referência para o balanço de alguns parâmetros relacionados com a escala de trabalho resulta no ponto fundamental da importância do trabalho, pois até hoje este processo vem sendo ignorado na elaboração das programações mensais efetuadas em empresas.

Por outro lado, todas as soluções a serem geradas vão garantir o cumprimento das restrições impostas pelos estatutos da empresa, concentrando o esforço na satisfação dos requisitos ou condições dos trabalhadores envolvidos, e garantindo alguns critérios adicionais surgidos nas diferentes análises para a elaboração das escalas.

Também é realizado um estudo comparativo dos horários administrados anualmente com o objetivo de balancear ao máximo as horas de trabalho diurno e noturno, os tempos de descanso e as horas pernoitadas associadas a equipe de trabalho.

Por último, realiza-se um estudo e desenvolvimento do algoritmo utilizado como ferramenta na solução de problemas de alta complexidade, em nosso caso particular os problemas de programação de escalas. O estudo do algoritmo evolutivo considera fundamentalmente alguns dos principais parâmetros, particularmente : representação dos indivíduos, tamanho inicial da população, número máximo de iterações adequadas e alguns efeitos colaterais originados pelas características das funções objetivos formalizadas e usadas como critério de avaliação da qualidade de cada solução.

1.4.- Objetivos do trabalho

Os principais objetivos deste trabalho são os seguintes:

1. Estudar e formalizar um problema de programação de escalas (seqüenciamento de tarefas), aplicado a sistemas de transporte ferroviário, levando em consideração todas as limitações, presentes no mundo real.
2. Definir e implementar uma ferramenta computacional usando técnicas heurísticas e técnicas de computação evolutiva, caracterizadas por operadores de *mutações* genéticas, simples e múltiplas, de caráter restrito.

3. Comparar os resultados atingidos pelas metodologias propostas com as metodologias de solução empregadas atualmente, usando critérios de avaliação do tipo :
   - Horas de trabalho diurno programadas.
   - Horas de trabalho noturno programadas.
   - Horas de descanso programadas.
   - Número de pernoites programados (tarefas noturnas consecutivas).

1.5.- Organização do texto

Neste trabalho, cada capítulo está elaborado de forma gradual e detalhada, com o propósito de dar maior fluidez à leitura e compreensão dos aspectos essenciais. Objetiva-se assim um bom entendimento dos problemas relacionados com programação de escalas e sua solução usando técnicas recentes da Inteligência Computacional, em geral, e dos algoritmos evolutivos, em particular .

A estrutura do texto, dividida em seis capítulos, foi organizada da seguinte maneira : após este capítulo introdutório, o Capítulo 2 apresenta os conceitos básicos envolvidos nos problemas de programação de escalas. A explicação detalhada dos aspectos mais comuns destes problemas vai constituir uma base teórica necessária para compreender o processo de escalonamento em sentido geral. Como parte final do capítulo, é apresentado um breve resumo de aplicações reportadas na literatura como exemplos ilustrativos dos estudos realizados até nossos dias na procura de metodologias de solução. Continuando, o Capítulo 3 detalha os modelos de programação de escalas utilizados na área do transporte, em especial no setor ferroviário. Nesta etapa, expõem-se as principais particularidades (com respeito ao escalonamento), inerentes ao transporte ferroviário, quais sejam : tipos e classificação das tarefas que são programadas, horários de trabalho, estatutos e regulamentos, direitos constitucionais dos trabalhadores e outros critérios práticos. No Capítulo 4, apresentam-se conceitos, definições e formalizações de diferentes estruturas de algoritmos evolutivos aplicados a problemas de escalonamentos de diferentes naturezas. No contexto, explicam-se as diferentes fases ou etapas que identificam os algoritmos evolutivos, dando ênfase às aplicações relacionadas a

problemas de geração de escalas de trabalho. A seguir, o Capítulo 5 apresenta a estrutura básica da ferramenta computacional desenvolvida para a geração de escalas típicas do setor ferroviário brasileiro. Em forma simples, explicam-se as principais classes implementadas, o modo operacional da ferramenta e, por último, é estudado um exemplo prático cujos resultados são analisados e comparados com os resultados obtidos atualmente na empresa ferroviária. Finalmente, o Capítulo 6 apresenta as conclusões do trabalho e também são sugeridas novas propostas de trabalhos para o futuro.

Capítulo 2

Programação de Escalas

2.1.- Introdução

Este capítulo introduz as definições e os conceitos básicos envolvidos em problemas de programação de escalas. O propósito é o de fornecer os subsídios necessários para a compreensão do processo geral de escalonamento de tarefas no contexto de programação de escalas.

Atualmente, muitos elementos estão presentes neste tipo de problema. Contudo, nem sempre pode-se detalhar cada um deles devido à complexidade das aplicações atuais. No entanto, pretende-se explicar, em forma simples, mas compreensível, alguns dos aspectos fundamentais tais como : períodos de planejamento (de trabalho e descanso), tipos de escalas, tipos de tarefas programadas em cada escala e outros. Também é apresentada uma definição global e estrutural de um problema de programação de escalas, a qual visa estabelecer a seqüência de passos necessários para conformar um bom escalonamento, ou seja, um programa de escalas.

Como resenha histórica, apresenta-se um esboço dos trabalhos realizados nesta área, comentando-se brevemente algumas das metodologias utilizadas e ressaltando suas vantagens e desvantagens.

2.2.- Programação de escalas

Basicamente, a solução de um problema de escalonamento apresenta-se como um processo seqüencial de tarefas, onde cada tarefa a seguir na seqüência, deve satisfazer os requisitos correspondentes ao processo de alocação.

Neste âmbito, vários fatores, além das restrições, estão presentes nos processos de geração das escalas. A seguir, abordam-se os mais importantes.

2.2.1.- Definições básicas

Todo processo de alocação deve manter uma determinada ordem ou seqüência lógica que, unida ao cumprimento das limitações impostas e critérios de avaliação, determina a qualidade do produto (escalonamento) resultante.

A seguir, mostramos um exemplo ilustrativo de um processo seqüencial de tarefas envolvendo 4 trabalhadores. O exemplo servirá de base para definir alguns dos principais aspectos explicita e implicitamente relacionados na tabela abaixo.

# Semana	Seg.	Ter.	Qua.	Qui	Sex.	Sab.	Dom.
1	D	D	D	D	X	X	D
2	D	D	V	V	V	X	X
3	V	V	V	N	N	N	N
4	X	X	X	D	D	D	X

{ D, V, N, X } ⇔ { tarefa diurna, tarefa vespertina, tarefa noturna, dia de descanso }

Tabela 2.1. Exemplo de uma programação de escalas.

Observa-se na Tabela 2.1 vários elementos relacionados, respectivamente : períodos de trabalho e descanso, período total de trabalho planejado (conhecido na literatura como horizonte de planejamento), tipos de escalas, tipos de tarefas programadas entre outros.

2.2.1.1.- Períodos de trabalho, descanso e horizonte de planejamento

Um dos elementos básicos que compõem um escalonamento são os períodos planejados. Neste contexto, podemos citar os períodos de trabalho, períodos de descanso e o período total ou horizonte de planejamento. Tais períodos, principalmente os períodos de trabalho, são determinados por padrões de demandas e outras restrições adicionais.

O período de trabalho (descanso) consiste em uma seqüência de dias em que uma pessoa pode trabalhar (ou descansar) e pode-se classificar segundo os tipos de tarefas programadas. No exemplo mostrado na Tabela 2.1, para a semana 1, (Seg-Qui) denota um período de trabalho diurno de tamanho 4, na semana 2, (Qua-Sex) denota um período de trabalho vespertino de tamanho 3, na semana 3, (Qui-Dom) denota um período de trabalho noturno de tamanho 4 e na semana 4, (Seg-Qua) representa um período de descanso.

O horizonte de planejamento ou período total planejado representa o tamanho total do escalonamento e corresponde a um ciclo completo das seqüências de trabalho e descanso. No exemplo apresentado na Tabela 2.1, o horizonte de planejamento corresponde a quatro semanas programadas e, no final de cada ciclo correspondente a estas 4 semanas, vai se repetir o processo de escalonamento anterior em forma completa.

Geralmente, estes períodos são determinados pela direção da empresa, sendo que os períodos (de trabalho ou descanso) são analisados semanalmente e os períodos de planejamento total são analisados mensalmente.

2.2.1.2.- Tipos de escalas

Em geral, as escalas citadas na literatura estão divididas em dois grupos: escalas do tipo cíclicas (ou comuns) e escalas do tipo individual (ou fixas) [1, 11].

As escalas cíclicas representam um conjunto de seqüências de trabalho (tarefas) comuns para todos os trabalhadores envolvidos, durante todo o período de planejamento. Neste caso, a duração do ciclo será de valor M, o número total de trabalhadores vezes D dias, o que significa que cada trabalhador passa pelos M sub-períodos de duração D, e repete a seqüência comum cada $M \times D$ dias.

Um dos principais motivos deste tipo de escalonamento consiste em obter a mesma seqüência de tempos de trabalho e descanso para cada trabalhador, considerando todos os períodos planejados. Desta forma, é garantida uma distribuição uniforme concernente aos horários, tanto de trabalho como de descansos. No entanto, nem sempre é possível atingir este objetivo pois, em muitas aplicações, restrições e perturbações impedem o desenvolvimento de seqüências iguais para cada um dos trabalhadores.

Devido a isto, são estabelecidas escalas por seqüências individuais em que cada trabalhador responde a um grupo de tarefas diferentes das programadas para o resto dos trabalhadores. Este tipo de programação normalmente está presente em aplicações onde o número de trabalhadores é superior ao horizonte de planejamento.

Por exemplo, se durante 4 dias alocarmos as tarefas (jornadas de trabalho ou descanso) mostradas na Figura 2.1(a), para 5 trabalhadores, um possível resultado seria como o mostrado na Figura 2.1(b).

J1	J2	J3	J4	J5

(a)

Dias

Trabalhadores	D1	D2	D3	D4
T1	J1	J2	J3	J4
T2	J2	J3	J4	J5
T3	J3	J4	J5	J1
T4	J4	J5	J1	J2
T5	J5	J1	J2	J3

(b)

Figura 2.1. Escalonamento para 5 trabalhadores durante 4 dias de trabalho.
(a) Tarefas a programar. (b) Programação para 4 dias de trabalho.

Neste exemplo, uma das condições a satisfazer é que todas as tarefas sejam designadas em cada dia de trabalho evitando que dois ou mais trabalhadores façam uma mesma tarefa no mesmo dia. Na tabela mostrada na Figura 2.1(b), podemos observar que sempre vai existir uma tarefa diferente na seqüência designada a cada trabalhador.

mesma tarefa no mesmo dia. Na tabela mostrada na Figura 2.1(b), podemos observar que sempre vai existir uma tarefa diferente na seqüência designada a cada trabalhador.

No caso das seqüências atribuídas aos dois primeiros trabalhadores por exemplo, a tarefa representada pela jornada de serviço J5, programada para o segundo trabalhador, não entra na seqüência de tarefas correspondente ao primeiro enquanto a tarefa representada pela jornada de serviço J1, atribuída ao primeiro, não entra no programa do segundo trabalhador.

Desta forma, ao longo dos quatro dias planejados, cada trabalhador deverá realizar uma seqüência diferente de tarefas. Aqui surge um dos problemas ao elaborar escalas de tal forma que, nas estatísticas referentes à carga de trabalho, exista uma diferença mínima entre as horas de trabalho ou descanso designadas a cada trabalhador envolvido na programação.

2.2.1.3.- Classificação das tarefas (jornadas de trabalho)

Comumente, costuma-se classificar as tarefas segundo os horários de trabalho estabelecidos para cada uma delas e, dependendo das características de execução, podem ser incorporadas outras classificações, caso existam grandes diferenças entre as operações a realizar durante cada período de trabalho.

Geralmente, um dia de trabalho apresenta três períodos fundamentais, chamados de jornadas, usualmente definidos pelos horários de trabalho regulamentados nas empresas.

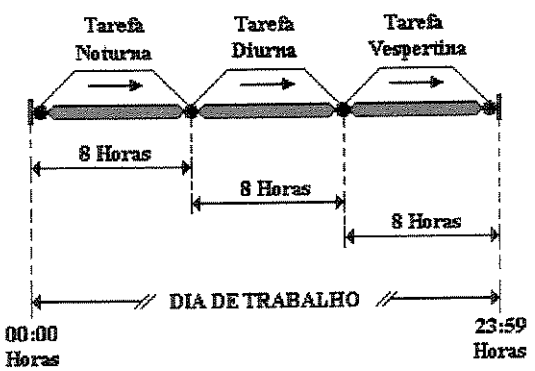


Figura 2.2. Jornadas de trabalho típicas.

Como observamos na Figura 2.2, um padrão típico de divisão de horários de trabalho é o seguinte:

1. ***Jornadas Noturnas*** (00:00 – 08:00 hrs).
2. ***Jornadas Diurnas*** (08:00 – 16:00 hrs).
3. ***Jornadas Vespertinas*** (16:00 – 24:00 hrs).

No exemplo da Tabela 2.1, onde observamos uma determinada seqüência de tarefas de trabalho e descanso através das semanas planejadas, apresentam-se estes três tipos genéricos de jornadas. Neste caso, a letra D representa um período de trabalho correspondente a uma tarefa diurna, a letra V correspondente a uma tarefa vespertina, a letra N representa uma tarefa noturna e a letra X representa um período de descanso ou dia de folga.

No entanto, uma conseqüência desta classificação é que nem sempre podemos generalizar tal definição, pois existem muitos setores onde as tarefas apresentam variações nos horários previstos, não se adequando à classificação mencionada.

De fato, os programas elaborados em muitas empresas não correspondem a estas definições, mas servem de base para compreender melhor o processo de escalonamento, pois as classificações podem ser estendidas e aplicadas a qualquer programa empresarial, mesmo que os horários previstos não correspondam às jornadas típicas.

Por exemplo, existem empresas em que as tarefas a programar apresentam jornadas superiores a oito horas de trabalho, ou em que os horários de início e fim podem ser estabelecidos para qualquer hora do dia. No entanto, isto não impede que tarefas possam ser classificadas em diurnas, vespertinas ou noturnas, dentro de limites conceituais permissíveis.

As tarefas que, por causa da natureza do problema, apresentam jornadas de trabalho superiores a 24 horas, e que não se adaptam a estas classificações, podem ser classificadas usando algum critério adicional, que possa diferenciá-las das tarefas restantes. Esta última classificação será mais específica da aplicação e depende do responsável pela elaboração da programação de escalas.

Além dos horários de trabalho, as tarefas podem ter outras diferenças que condicionam uma análise adicional no processo de alocação. Quando existem diferenças marcantes na execução das tarefas relacionadas, pode ser estabelecida uma determinada classificação segundo a manobra operativa. Desta forma, poderemos diferenciar aquelas tarefas que podem influir, através do comportamento da execução, na estabilidade do sistema operativo de trabalho. Por exemplo, no caso do setor ferroviário (área de nosso interesse), existem alguns tipos de tarefas nomeadas *Prontidão-Retorno*, onde a operação a realizar tem a ver com a movimentação dos trens de uma estação a outra e onde uma boa operação de trabalho depende das condições operacionais do trem, da linha, entre outras. Isto quer dizer que, se acontece alguma parada temporal causada por defeitos nos trens, interferências nas linhas, ou demoras causadas por fenômenos naturais, o intervalo de tempo para executar será alterado. A partir desta perturbação o sistema, aos poucos, vai perdendo estabilidade e pode levar à deterioração da programação, sendo necessária a procura de soluções alternativas (programação reativa). Nestes casos, para evitar tais inconvenientes e para facilitar as análises e a elaboração do escalonamento, deve-se classificar as tarefas segundo a execução das suas operações correspondentes (no próximo capítulo apresentaremos mais detalhes sobre estes aspectos).

Por outro lado, existem problemas em que não precisamos estabelecer tal classificação para as tarefas a programar. Por exemplo, existem aplicações em que o serviço não apresenta características diferentes de execução. As tarefas só se diferenciam em seus tempos de início e nas respectivas durações das jornadas de trabalho. Assim, para efetuar a programação, só seriam analisados outros critérios e as diferenças estariam fundamentalmente nos horários de trabalho.

Até o momento, mencionou-se alguns dos principais aspectos que envolvem problemas de escalonamento, mas o mesmo ainda não está totalmente definido. Existem outros aspectos que determinam a complexidade de problemas desta natureza. Outros parâmetros, que definem a qualidade de uma programação de trabalho, estão relacionados com o número de trabalhadores necessários para satisfazer a demanda diária e do período de planejamento, a ordem das tarefas de trabalho e descanso a programar, e as correspondentes restrições a serem satisfeitas. Estes elementos estão interrelacionados e

constituem, por sua vez, um conjunto de passos para se definir um problema geral de escalonamento.

2.2.2.- Passos que definem problemas de escalonamento

Baseado nas similaridades entre as diferentes aplicações descritas na literatura, foi possível estabelecer um quadro comum para representar ou definir problemas de escalonamentos. Sobre esta base, Tien and Kamiyama [32] apresentam uma definição estrutural para um problema geral de programação de escalas, considerando-o como a composição ou seqüência dos seguintes elementos ou passos :

Passo 1.- Alocação da força de trabalho temporal em termos de uma matriz $T(i,j)$, representando o número de trabalhadores necessários para executar as tarefas de um turno de trabalho j ($j = 1,2,...,J$), no dia i ($i = 1,2,...,I$).

Passo 2.- Alocação da força de trabalho total em termos de M, o número total de trabalhadores requerido para satisfazer outras restrições além do passo anterior.

Passo 3.- Identificação dos tempos de descanso (folgas) em termos de $X(t)$, o número de trabalhadores com período de descanso de tipo t durante um período planejado de I dias.

Passo 4.- Designação da seqüência dos períodos de descanso identificados no passo anterior, usando uma determinada função f que identifique uma seqüência de períodos de descanso durante o período de trabalho planejado para cada trabalhador.

Passo 5.- Designação das tarefas em termos de uma matriz R, a qual define as tarefas a serem realizadas durante o período planejado.

Em síntese, todo escalonamento é caracterizado por esta seqüência de passos, mesmo que não sejam totalmente explícitos.

Alguns aspectos importantes, derivados da estrutura apresentada, podem ser destacados a partir da análise desta definição. Em primeiro lugar, deve-se notar a relação entre a seqüência de passos e a garantia de uma boa formação de um escalonamento. Existem algoritmos que invertem passos ou resolvem dois ou mais passos simultaneamente. Por exemplo, existem casos em que a determinação da força de trabalho mínima requerida (passo 2), está condicionada às restrições impostas pelos passos 3 e 4. Ou seja, o número mínimo de trabalhadores necessário é calculado tendo em conta as restrições inerentes aos sequenciamentos dos períodos de trabalho e de descanso. Em segundo lugar, podem existir interações entre os diferentes passos apresentados. Um exemplo seria se uma determinada solução não factível do escalonamento obtida durante os passos 3, 4 e 5 induz uma modificação no número total de trabalhadores calculados no passo 2. Em geral, esta modificação exige um aumento no número de trabalhadores anteriormente calculado. No entanto, uma modificação no passo 2 (força de trabalho total), não implica em uma mudança da força temporal calculada no passo 1. Esta última poderia ser modificada se, por exemplo, a demanda sofresse mudanças decorrentes das necessidades de serviço.

Um forma mais compacta de encarar estes problemas, apesar de constituir o mesmo procedimento, foi baseada na composição de três etapas fundamentais :

1.- Alocação da força de trabalho (*allocation*).

2.- Designação dos tempos de descanso (*offday scheduling*).

3.- Designação das tarefas (*shift assignment*).

Neste caso, na primeira etapa determina-se o número de trabalhadores necessário para cada turno de trabalho, a cada dia (satisfazendo os requisitos da empresa). Determina-se também o número mínimo de trabalhadores necessário para satisfazer as demandas ao longo do período de planejamento.

A segunda etapa estabelece os períodos de descanso segundo as restrições, como por exemplo incluir *A* fins de semana de descanso a cada *B* fins de semana que cobre o período planejado, com $A<B$.

Por último, a terceira etapa estabelece os tempos de trabalho, sujeitos às demandas e às restrições que impõem as trocas de turnos de trabalho. Normalmente, estas restrições têm a ver com o descanso que cada trabalhador deve ter entre duas jornadas consecutivas. Além disso, tem a ver com a repetição de uma determinada jornada de trabalho, pois, em geral, as preferências são para trabalhar durante jornadas diurnas ou vespertinas, tentando sempre evitar os horários noturnos.

Uma vez definido o problema, o passo seguinte visa um estudo dos formalismos de análise mais comuns para modelar o problema, assim como as metodologias de solução.

2.3.- Teorias e metodologias de programação de escalas

Nesta seção, serão analisados os modelos e metodologias mais usados na elaboração de programas de escalas.

2.3.1.- Modelos de programação de escalas

Nos primeiros estudos realizados, a grande maioria das tentativas para desenvolver modelos para esta classe de problemas foram baseadas em métodos de programação inteira. Aykin [3] faz uma referência a um trabalho de Dantzig, onde se formula o seguinte problema :

$$\text{Minimizar} \quad \sum_{k \in K} c_k X_k ,$$

Sujeito a :

$$\sum_{k \in K} a_{k\,t} X_k \geq b_t \quad \forall\, t \in T,$$

$$X_k \geq 0 \text{ e inteiro.}$$

onde :

K : Conjunto de todas as tarefas a serem programadas.

T : Conjunto de todos os períodos de trabalho.

b_t : Número de trabalhadores necessários para um período de trabalho *t*.

c_k : Custo de designar um trabalhador a uma tarefa *k*.

$$a_{kt} = \begin{cases} 1, \text{ se o período } t \text{ é um período de trabalho associado a uma tarefa } k. \\ 0, \text{ caso contrário.} \end{cases}$$

X_k : Variável inteira que define o número de trabalhadores designados a uma tarefa k, $k \in K$.

Muitas aplicações foram baseadas neste modelo genérico de programação inteira (também conhecido como modelo *set covering*), mas na medida que se aumenta o número de variáveis e restrições, é necessário modificar ou desenvolver modelos que possam resolver as limitações impostas pela complexidade de problemas de otimização combinatória.

Com o aumento da complexidade, novos modelos e metodologias foram surgindo para determinar soluções factíveis. Uma visão ampla sobre as pesquisas desenvolvidas nesta área foi apresentada por Tien e Kamiyama [32]. Seguindo as linhas comuns dos vários trabalhos estudados, foi estabelecida uma série de modelos analíticos gerais baseados na seqüência de passos mencionada anteriormente.

A seguir, usando como fonte os estudos realizados por Tien e Kamiyama [32], serão comentados alguns dos aspectos referentes a cada passo para uma maior compreensão da modelagem geral estabelecida.

1.- Cálculo da força de trabalho temporal

Modelagem : Otimizar T(i,j), onde $T(i,j) \geq 0$.

Nesta primeira etapa, muitos dos algoritmos assumem este valor conhecido e os outros o consideram como um problema de otimização, onde tenta-se minimizar uma

função objetivo (usualmente a matriz T(i,j) como variável de decisão que, como mencionamos anteriormente, representa o número de trabalhadores necessários para um turno de trabalho *j*, no dia *i*, sujeito a padrões de demanda (exemplo: jornadas extras de trabalho, etc.), legislação trabalhista, limitações de recursos disponíveis (que inclui a força de trabalho total requerida) e outras restrições, como os limites inferior ou superior da matriz T(i,j)). Nesta formulação, a variável *I* representa o horizonte de planejamento e a variável *J* representa o número total de turnos de trabalho designados por dia.

2.- Cálculo da força de trabalho total

$$\text{Modelagem: } M = \left[\frac{\Sigma_i \Sigma_j T(i,j)}{C} \right]$$

onde:

[x] : Menor valor inteiro maior ou igual a x.

M : Número total de trabalhadores necessários para satisfazer requisitos temporais e outras restrições.

C : Dias trabalhados durante o horizonte de planejamento.

Neste segunda etapa, de forma similar ao item 1, normalmente assume-se um valor para M, sendo que outros calculam M (força total), através da relação de T(i,j) (força temporal), e uma constante C (número de dias designados para um trabalhador em um horizonte de planejamento). Por exemplo, para um período de 7 dias, C=5, embora pode ser menor devido a dias fora de trabalho como dias de aniversários, dias livres causados por doenças, entre outros.

Para estes dois primeiros passos e dependendo dos requisitos de cada problema, o método de cálculo do número de trabalhadores necessário varia de um para outro algoritmo. Burns e Carter [8], Burns e Koop [9] e Koop [19] apresentam alguns procedimentos, otimizando o número de trabalhadores necessário para cada período, dependendo das condições que limitam o problema tratado.

3.- Identificação dos períodos de descanso

Modelagem : Maximizar $\sum_t V(t)X(t)$, sujeito a : $\underline{A}\,\underline{X} \leq \underline{Y}$, $\underline{X} \geq 0$ e inteiro

onde :

V(t), Valor designado a um período de descanso de tipo *t* (t=1, 2,..., T).

$\underline{X} = (X(t))$; Vetor coluna com T elementos.

X(t), Número de trabalhadores com um período de descanso do tipo *t* durante um horizonte de planejamento *I*.

$\underline{A} = (A(i,t))$; Matriz de tamanho $I \times T$ onde:

$$A(i,t) = \begin{cases} 1, \text{ se o dia } i \text{ é um período de descanso de tipo } t. \\ \\ 0, \text{ caso contrário.} \end{cases}$$

$\underline{Y} = (Y(i))$; Vetor coluna com I elementos.

Y(i), Número de trabalhadores que podem descansar no dia *i* (*i*=1, 2,...I).

Aqui, identificam-se os períodos de descanso dentro da programação de trabalho. A partir das etapas 1 e 2, podemos determinar o número máximo de trabalhadores em folga cada dia da semana. Isto eqüivale ao número de vezes que o dia *i* aparece como dia de descanso no horizonte de planejamento.

Usando esta informação, pode-se atribuir valores subjetivos a cada um dos possíveis períodos de descanso e determinar o conjunto que satisfaz os requisitos de forma mais eficiente. É importante destacar que cada um dos períodos de descanso programado deve satisfazer às restrições impostas. Como resultado deste item, teremos o número de vezes (freqüência) que será usado um período de descanso no horizonte de planejamento.

4.- Identificação das seqüências dos períodos de descanso

Modelagem : Minimizar $\sum_m \sum_n c(m,n)\, f(m,n)$

Sujeito a :

(1) $\Sigma_n f(m,n) = 1$, para $m = 1,2...,X$

(2) $\Sigma_m f(m,n) = 1$, para $n = 1,2...,X$

(3) $\Sigma_{m\in Q}\Sigma_{n\in \overline{Q}} f(m,n) \geq 1, \forall Q$

(4) $\Sigma_m \Sigma_n D(m,n)f(m,n) = W$

(5) $f(m,n) = 0,1;\ m,n = 1,2,...X$

onde :

$X = \Sigma_t X(t)$, número total de períodos de descanso.

$f(m,n) = \begin{cases} 1, \text{ se o período de descanso } m \text{ é seguido por um período de descanso } n. \\ 0, \text{ caso contrário.} \end{cases}$

$\underline{D} = (D(m,n))$: Matriz de dimensão $X \times X$ onde D(m, n) é o número de dias de trabalho que separam os períodos de descanso m e n.

W : Número total de dias de trabalho dentro do horizonte de planejamento.

Q : Subconjunto de {1, 2,..., X}; $\overline{Q}$: Complemento de Q.

c(m,n) : Custo do período de trabalho que separa os períodos de descanso m e n.

Durante esta etapa, determina-se a seqüência dos períodos de descanso resultantes da etapa 3, notando que os tempos entre períodos de descanso correspondem a períodos de trabalho e também devem satisfazer às restrições do problema.

5.- Identificação das seqüências dos períodos de trabalho

Modelagem : Determinar a matriz S, sujeita aos requisitos da escala, relações de precedência e outras limitações.

onde :

$\underline{S} = (S(p,q))$; é uma matriz de dimensão P×Q, e S(p,q) é a escala designada a um trabalhador p (p=1,2,...P), no dia q (q=1, 2, ...Q).

Neste último passo, designam-se as seções ou turnos de trabalho, escalas correspondentes aos períodos de trabalho, tal que todos os requerimentos temporais possam ser cumpridos.

Desta forma, estabeleceu-se uma modelagem por partes para esta classe de problemas, e a tarefa seguinte é desenvolver as metodologias que, baseadas nestes modelos, sejam capazes de resolver os mais diversos problemas encontrados nesta área da engenharia.

2.3.2.- Metodologias de programação de escalas

Em conjunto com os modelos, diversas metodologias de programação de escalas foram apresentadas em Tien e Kamiyama [32], sendo que a maioria dos métodos usados são baseados em programação linear inteira.

Outras formas de atacar o problema são através do uso de modelos de fluxo em redes e programação usando módulos. Por exemplo, Balakrishnan e Wong [5] consideram o problema de escalonamento como um modelo de rede onde se trabalha com elementos importantes, como a identificação dos períodos de descanso, o seqüenciamento dos períodos de trabalho e/ou descanso e a geração das escalas de trabalho, tudo de forma simultânea. Para este caso, a solução ótima do problema corresponde ao caminho ótimo na rede, identificado via método de relaxação Lagrangeana.

Por outro lado, Burns e Koop [9] apresentam uma metodologia heurística usando programação em módulos. Neste caso, um dos objetivos é o cálculo do menor número de trabalhadores necessários para satisfazer todas as restrições, que incluem dois dias de descanso a cada semana, um número específico de fins de semana de descanso em qualquer número fixo de fins de semana, um número máximo de seis turnos diários consecutivos e diferentes demandas para cada tipo de turno. Uma vez determinado o número de trabalhadores necessários, o método constrói o escalonamento através da combinação de diferentes módulos preestabelecidos. Para isto, o algoritmo acessa uma

base de dados que contém um grande número de módulos, selecionando aquele módulo aplicável a uma determinada situação particular.

Com o intento de ampliar os campos de pesquisa na área, outros estudos vem sendo dirigidos no uso de novas heurísticas, técnicas metaheurísticas (e.g. técnicas de computação evolutiva) e combinação de ambas. Neste sentido, um modelo para a solução de um problema de *Set Covering* foi apresentado em Al-sultan *et at.* [2]. Outros resultados satisfatórios foram descritos em Filipec e Zupanic [12] e Van Bael *et at.* [34]. Problemas com características similares, tais como "*Time Table Problem*", "*Job Shop Scheduling Problem*" e "*Flow Shop Scheduling Problem*", vem sendo abordados usando estas linhas de pesquisas [23, 25, 26, 33, 34, 17]. Nestas aplicações, referentes ao uso de técnicas de computação evolutiva, geralmente uma programação de escalas é representada em forma de matriz, onde cada linha associa-se a um trabalhador e cada coluna a um dia de trabalho. Desta forma, é possível estabelecer uma codificação básica que permite o desenvolvimento destas aplicações.

2.4.- Aplicações práticas

Estudos iniciais na área de planificação de trabalho visavam estabelecer planos gerais de serviços, calculando a menor quantia de trabalhadores que, dada uma certa demanda, seria necessária e suficiente para cumprir as metas de produção, a um custo mínimo e sem violar os direitos e satisfações dos trabalhadores [4, 7, 8]. Outras aplicações eram direcionadas à busca de metodologias gerais, que pudessem resolver problemas com características e restrições similares, ou que pudessem ser adaptadas às condições propostas em cada aplicação [3].

Desde os anos 70, algumas aplicações usando metodologias de programação matemática em geral, e programação linear inteira em particular, já eram citadas na literatura, referindo-se principalmente a aplicações no setor público. Exemplos de aplicações em hospitais (programação de serviços de enfermaria), podem ser encontradas em Holmes [16] e Michael [24].

Outro setor muito enfatizado nesta área é o setor de transporte, incluindo transporte urbano, aéreo e ferroviário. Aparecido [1] apresenta um resumo de trabalhos nestas três áreas. Seu trabalho foi especificamente orientado para problemas de escalas no transporte ferroviário. Neste caso, considera-se o estudo e implementação de técnicas de preferência declaradas para resolver o problema de planejamento mensal de trabalho. Um dos principais objetivos é obter distribuições uniformes (equitativas) das satisfações dos trabalhadores. O planejamento de escalas é resolvido em duas etapas. A primeira procura obter uma escala cíclica otimizada, particionada em programações. Cada programação corresponde a uma seqüência de dias de trabalho entre duas folgas consecutivas. Duas propostas são apresentadas para resolver este problema: uma formulação matemática baseada no modelo *set covering* e um algoritmo heurístico baseado no problema de atribuição. A segunda etapa consiste em alocar as escalas de trabalho aos condutores, de forma que o nível de satisfação seja equitativamente distribuído. A atribuição é feita levando em consideração o nível de satisfação do histórico de trabalho de cada condutor, e também o nível de satisfação da escala futura.

Caprara *et at.* [10] também apresentam um trabalho aplicado em empresa de transporte ferroviário , neste caso na Itália. Uma abordagem também baseada em modelo *set covering,* usando um algoritmo heurístico. O modelo proposto tenta produzir uma escala cíclica de no máximo 30 dias. Caso o ciclo ultrapasse este limite, uma outra escala é produzida, também com esta limitação. Para cada escala de 30 dias um grupo de 30 condutores é selecionado. Uma escala é dividida em programações de 6 dias. Cada programação é constituída de cincos dias consecutivos de trabalho e uma folga, no sexto dia. A mesma escala de trabalho produzida é utilizada por um período de um ano. O objetivo principal foi descobrir um conjunto de escalas viáveis de no máximo 30 dias, cobrindo todas as atividades diárias e minimizando o número total de programações de 6 dias na escala. O número de condutores necessários para todos os dias é igual a 6 vezes o número total de programações. Assim, a minimização do número de programações implica na minimização do número global de condutores necessários.

Com relação a metodologias que utilizam metaheurísticas, mais especificamente as técnicas de computação evolutiva, várias pesquisas apresentam resultados promissores em diversas áreas da engenharia.

Filipec e Zupanic [12] desenvolvem uma nova metodologia de escalonamento baseada em técnicas de estratégias evolutivas. O algoritmo foi aplicado a uma indústria automobilística, em particular às linhas de produção da fábrica. O custo das escalas foi determinado pela qualidade da distribuição do consumo de energia nos períodos de maior demanda, minimizando o custo energético, analisando o efeito e adaptação do algoritmo ante situações não previstas.

Como mencionado, outros problemas típicos de escalonamentos, tais como *Flowshop* e *JobShop Scheduling Problems* foram resolvidos usando estas técnicas de computação evolutiva. Por exemplo, Ishibuchi e Murata [17] propuseram um algoritmo evolutivo, combinado com um procedimento de busca local, para resolver problemas multi-objetivos. A estratégia de evolução usa uma função de avaliação ponderada e cada um dos pesos são gerados aleatoriamente a cada geração para variar a direção de busca. As soluções durante cada geração são avaliadas em conjunto com algumas soluções vizinhas (busca local), evitando assim tempos de processamento grandes. O algoritmo foi aplicado para resolver um problema de *Flowshop*, sendo neste caso testados problemas com dois e três objetivos com bons resultados. Van Bael *et al.* [34] mostram que algoritmos evolutivos, combinados com técnicas de otimização local, podem resolver problemas de *Job Shop* mesmo para casos muito complexos.

Em outro exemplo, Tanomaru [31] apresenta um algoritmo evolutivo para planejamento de força de trabalho heterogênea. Neste caso, além dos operadores de reprodução e cruzamento, são implementados vários operadores heurísticos e estocásticos com o objetivo de diminuir certas penalidades que atuam em função das violações das restrições envolvidas. Este algoritmo mostrou bons resultados em problemas de dimensões moderadas.

2.5.- Resumo

Abordando as principais características históricas, recopiladas durante os estudos sobre escalonamentos gerais, tem sido possível adentrarmos na teoria metodológica de programação e nas peculiaridades desta classe de problemas.

Ao concluir esta etapa, dispomos de conceitos elementares que formam uma sólida base para elaborar esquemas direcionados a planejamento de escalas.

Durante as distintas seções deste capítulo, foram explicadas em detalhes as etapas necessárias para finalizar um bom programa de trabalho e, seja qual for o problema a tratar, parte do conhecimento básico tem sido exposto para este empenho.

Elementos básicos, como tipos de escalas, tipos de tarefas, períodos de trabalho e horizonte de planejamento, foram esclarecidos com o objetivo de compreender melhor as particularidades que formam o problema geral.

Por outro lado, explicitaram-se os modelos mais usados para representar esta classe de problemas. Em seguida, apresentou-se um resumo das metodologias usadas na busca de soluções, e exemplos de aplicações citadas na literatura foram mencionados.

Foi também estabelecida uma seqüência de passos como definição estrutural de um problema geral de escalonamento. É preciso destacar que toda aplicação referente ao contexto da programação de escalas de trabalho pode ser colocada através destas idéias, adaptando o conjunto de restrições envolvido a cada caso particular.

Segundo as análises realizadas a partir da bibliografia consultada, percebe-se a grande variedade de aplicações associadas aos problemas de escalonamento no sentido geral, seja de força de trabalho (recursos humanos) ou de máquinas, entre outros recursos. Cada problema apresenta um modelo e, partindo de cada modelo, encontramos diversos objetivos a analisar, assim como um conjunto diferente de restrições inerentes. Os modelos e métodos desenvolvidos variam conforme a concepção de cada problema. Os enfoques sempre são orientados pelas particularidades dos problemas ao invés de metodologias gerais que possam fornecer a soluções eficientes para todos ou para uma grande parte dos problemas desta natureza. Desta forma, a cada situação corresponde um modelo e um método mais adequado, uma vez que não existe uma ferramenta suficientemente robusta e geral para resolver todos os problemas.

Capítulo 3

Programação de Escalas em Transporte Ferroviário

3.1.- Introdução

O presente capítulo apresenta uma discussão sobre os modelos de programação de escalas aplicados na área de transporte ferroviário.

Grande parte dos conceitos relacionados nas seções seguintes estão baseados em informações proporcionadas por empresas ferroviárias integrantes do sistema de transporte ferroviário brasileiro.

A maioria das empresas de transporte ferroviário usa critérios e características similares com respeito às tarefas que são programadas em cada uma delas. Baseado nesta hipótese, poderemos estender os conceitos e generalizar as definições analisadas, dispensando qualquer tipo de especificação e falta de flexibilidade neste sentido.

Entre os principais elementos a serem tratados neste capítulo, incluem-se : a descrição e classificação das tarefas programadas por mês, horários de trabalho, estatutos e regulamentos, direitos dos trabalhadores e outros elementos práticos que podem influir na qualidade das programações. Por último, apresenta-se uma metodologia de programação de escalas utilizada para elaborar os planos de trabalho em algumas empresas. Esta metodologia é usada como referência para este trabalho.

3.2.- Características da programação de escalas em empresas ferroviárias

Publicações referentes à programação de escalas em empresas ferroviárias não são muito frequentes na literatura. Recentemente os trabalhos de Aparecido [1], Caprara *et al.* [10] e Gonçalves [14], apresentam uma contribuição da pesquisa operacional para tal propósito. Aparecido e Gonçalves, apresentaram um estudo detalhado do problema da geração de escalas em empresas de transporte ferroviário do Brasil, enquanto Caprara apresentou seus resultados para empresas de transporte ferroviário italiano. Estes sistemas são semelhantes no sentido operacional, ajudando a estabelecer considerações gerais relacionadas com conceitos e algumas especificações dos problemas de escalonamento no caso de empresas ferroviárias.

No Capítulo 1, quando apresentou-se a formulação do problema, mencionamos a frase : "Programação de escalas de equipagens nas empresas ferroviárias". Neste contexto, a *programação de escalas de equipagens* significa o *sequeciamento das tarefas* a serem realizadas pelas diferentes *equipes de trabalho* da empresa.

Geralmente, uma equipe de trabalho é composta por um maquinista e um auxiliar de maquinista, para executar cada operação de trabalho que caracteriza uma tarefa.

Usando esta analogia, nosso problema consiste basicamente em alocar as diferentes tarefas para cada equipe, geralmente em períodos mensais, de modo que possam ser satisfeitas todas as restrições, tanto as de caráter empresarial como aquelas relacionadas com os direitos dos trabalhadores, visando atingir objetivos relacionados com o balanço eqüitativo de carga de trabalho para cada equipe durante o período analisado.

As principais características presentes nos problemas de programação de tarefas, nas empresas ferroviárias, serão mostradas a seguir através dos diferentes elementos que compõem sua estrutura conceitual.

3.2.1.- Descrição das tarefas

Para ilustrar algumas das características de cada tarefa, pertencente aos programas de trabalho mensal de algumas empresas ferroviárias, considere a Tabela 3.1.

TAREFAS		
Código	Nome	Atividade
02	Manobra	Executa manobras na oficina de vagões. Equipe com maquinistas e auxiliar.
08	Manobra 1	Executa manobras para formação de trens nos pátios. Equipe com maquinista e auxiliar.
09	Prontidão-Retorno	Equipam os trens nos horários prefixados. A expressão "Retorno" indica que esta equipe não vai tirar descanso fora da sede. Equipe com maquinista e auxiliar.
10	Dividendo	Executam todas as tarefas com apresentações programadas. Funciona como reserva caso haja defasagem no dimensionamento. Equipe com maquinista e auxiliar.
12	Abastecimento	Manobra locomotivas para abastecimento de óleo diesel, cárter e areia. Eventualmente auxiliam na formação de trens. Equipe com maquinista e auxiliar.
06	Lastro	Equipam trens de manutenção da via. Normalmente são maquinistas fixos nesta tarefa. Equipe com maquinista e auxiliar.

Tabela 3.1. Exemplo de tarefas programadas em empresas ferroviárias.

Outros elementos presentes são considerados como extra-tarefas. Estes elementos também formam parte dos planos de trabalho realizados durante um determinado horizonte de planejamento. Exemplos de extra-tarefas são mostrados na Tabela 3.2.

EXTRA-TAREFAS		
Código	Nome	Atividade
50	Folga	Período de descanso do empregado na sua sede de lotação. Não deverá ser inferior a 24 horas.
56	Auxílio Doença	Afastamento pelo INSS, após 15 dias.
58	Punição (Suspensão)	Punição disciplinar.
59	Férias	Férias anuais.
60	Fora de Escala	Não tem programação para o dia.
64	Revisão Médica	Exame periódico anual.
65	Convocação Judicial	Atendimento a qualquer solicitação judicial.
69	Folga de Aniversário	Dia de folga atendendo a acordo coletivo.
72	Acidente de Trabalho	Afastamento por acidente de trabalho.
**	Intervalo na sede	Período de descanso (superior a 10 horas) do empregado na sua sede de lotação.
**	Descanso fora da sede	Período de descanso (superior a 10 horas) do empregado fora sua sede de lotação.

** Conceito utilizado no dimensionamento das escalas.

Tabela 3.2. Exemplo de extra-tarefas programadas em empresas ferroviárias.

Como podemos observar, tarefas e extra-tarefas apresentam três atributos básicos distintos : **código**, **nome** e **atividade.**

O **código** de cada tarefa representa um identificador ou parâmetro de referência que está relacionado com o tipo de tarefa e atividade a realizar. Assim, toda tarefa que realiza a mesma operação, possuirá o mesmo código. O **nome** da tarefa, como já é indicado, representa o tipo de tarefa e determina implicitamente a operação que vai ser realizada. Por último, a **atividade** não é mais do que a descrição da operação ou seqüência de operações a realizar na jornada de trabalho correspondente à tarefa em questão.

Para o grupo das extra-tarefas, a definição apresenta variações quanto à execução. Neste caso, devido às suas características, as extra-tarefas não representam uma operação de trabalho, mas entram no planejamento. Por exemplo, segundo a Tabela 3.2, a tarefa de código 50 representa um dia de folga e uma tarefa de código 60 representa uma fora de escala, ou seja que, o trabalhador não possui uma programação de trabalho nesse dia. Ambas as tarefas, apesar de serem da mesma classe, tem grandes diferenças quanto ao significado para um trabalhador. Devido a isto, podemos nos referir a cada extra-tarefa como um estado adicional dentro das escalas, representando outras especificações. Por exemplo, uma folga não apresenta horários fixos preestabelecidos e representa pelo menos um dia de descanso (24:00 h ou mais). O fato de não apresentar horários fixos preestabelecidos indica uma certa dinâmica nos horários, condicionada pela tarefa que termina antes da designação de cada folga. Ou seja, apesar de ter uma seqüência fixa de alocação, a folga varia seus horários de início e fim condicionados pelos horários finais das tarefas programadas no dia anterior à dita folga.

Diferentemente da folga, uma situação de fora de escala representa um estado em que o trabalhador fica sem programação em um determinado dia, causado pelas restrições de descanso entre duas tarefas consecutivas ou por algum outro critério adicional. Este estado "sem escala ou sem programação" tem sempre uma duração menor que 24:00 h.

Outras diferenças existem entre categorias de atividades. No caso, os critérios usados no processo de alocação são baseados em considerações distintas de cada caso. As figuras a seguir ajudam a compreender melhor estas diferenças:

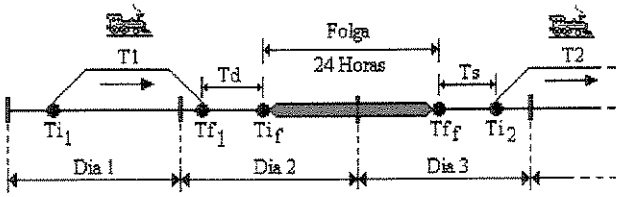


Figura 3.1. Representação de alocação de folgas.

Na Figura 3.1, observamos que a tarefa 1 (T1) inicia sua execução (Ti_1) no dia 1 e, devido à duração de sua jornada de trabalho, acaba no dia 2. Neste exemplo assume-se que, o dia 2, segundo os critérios estabelecidos para designar as escalas, é o dia correspondente a uma folga. Desta forma, partindo do horário final (Tf_1) da tarefa 1, e considerando também o tempo de descanso mandatório entre tarefas (o que, segundo a Figura 3.1, está representado por Td), a folga se iniciará em um tempo $Ti_f = Tf_1 + Td$ e terá uma duração de 24 horas. Uma vez concluído o tempo da folga, a tarefa a seguir, neste caso a tarefa 2 (T2), poderá começar imediatamente após o fim da folga, isto é, após Tf_f sempre e quando o término da folga não seja anterior às 5:00 h (considerando Ts como o tempo entre Tf_f e Ti_2). Caso contrário, a tarefa que continuaria na seqüência, deverá começar depois das 5:00 h da manhã. Mais detalhes serão fornecidos na seção (3.3.1).

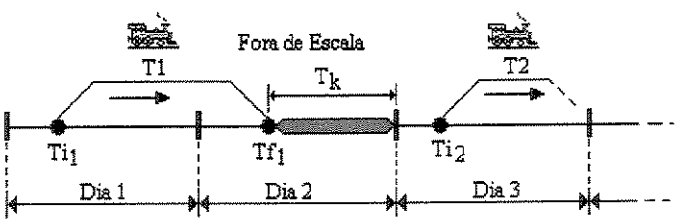


Figura 3.2. Representação de fora de escala (período sem programação).

Na Figura 3.2, observamos uma situação diferente da anterior. No exemplo mostrado vemos que a tarefa 1 também inicia sua operação no dia 1 concluindo a operação no dia 2. Mas agora, assume-se que o dia 2 não é o dia correspondente a um descanso. Sendo assim, o dia 2 torna-se um dia sem programação devido às restrições inerentes ao tempo de descanso, embora possam ser estabelecidos outros critérios adicionais para definir ou determinar um período sem programação. Segundo o exemplo mostrado na Figura 3.2, temos um tempo residual Tk, igual à diferença entre 24 h e o tempo final da tarefa

anterior (T1). Uma vez calculado o tempo residual Tk, este é comparado com o tempo de descanso obrigatório entre duas tarefas consecutivas. Caso o tempo residual Tk seja menor que o valor do tempo necessário para descansar entre duas tarefas consecutivas, o trabalhador fica sem programação para esse dia e só poderá reiniciar seu serviço após o período de descanso.

Estas duas diferentes situações são pontos fundamentais na programação de seqüências a cada trabalhador. Nas seções a seguir são apresentados outros detalhes adicionais que formam as análises de programação das folgas e dos dias onde o trabalhador fica sem programação.

3.2.2.- Dimensionamento das tarefas

O dimensionamento das tarefas é usualmente realizado ou verificado mensalmente na empresa devido à variação da demanda de trabalho. De acordo com a demanda, podem ocorrer mudanças nos horários de trabalho ou na quantidade de tarefas a programar. Eventualmente, nem todas as tarefas seriam necessárias ou suficientes para satisfazer a demanda de um determinado mês. Desta forma, não é possível estabelecer um planejamento geral a longo prazo (escalas cíclicas), pois o processo de alocação deve se adaptar a estas condições de variabilidade. A Tabela 3.3 mostra um exemplo de dimensionamento.

Código	Tarefa	Tempo Inicial [h]	Jornada [h]
02	Manobra	06:30	8:00
02	Manobra	23:00	8:00
:	:	:	:
08	Manobra 1	18:30	8:00
08	Manobra 1	23:01	8:00
:	:	:	:
09	Prontidão-Retorno	09:00	15:00
09	Prontidão-Retorno	23:00	15:00
:	:	:	:
10	Dividendo	05:01	15:00
10	Dividendo	06:00	15:00
:	:	:	:
12	Abastecimento	07:00	12
12	Abastecimento	07:01	12
:	:	:	:

Tabela 3.3. Exemplo de dimensionamento de tarefas.

Aqui, além dos códigos e nomes das tarefas, são relacionados a hora de início e a duração da jornada de trabalho.

Segundo os valores mostrados na Tabela 3.3, constatamos o discutido anteriormente no Capítulo 2 a respeito das características das tarefas e sua relação com o padrão genérico de divisão de trabalho mostrado na Figura 2.2. De fato, as tarefas apresentam hora de início variada durante o dia, e duração de jornadas de trabalho diferentes de 8 horas fixas.

3.2.3.- Classificação das tarefas

Segundo as análises realizadas em relação às tarefas, típicas de programas de empresas ferroviárias do Brasil, tomamos a iniciativa de caracterizar ou classificar cada tarefa usando dois critérios fundamentais, que nos permitissem estabelecer certa diferenciação entre elas.

No caso particular da geração de escalas na empresa ferroviária, as tarefas são classificadas segundo as suas características de execução e segundo os horários disponibilizados para cada uma delas.

É importante destacar que tal classificação não está tecnicamente estabelecida nas referências bibliográficas consultadas. Tais conceitos foram definidos como base para o desenvolvimento da metodologia adotada e a ferramenta computacional associada.

3.2.3.1.- Classificação das tarefas segundo as características de execução

Analisando as características de execução, como já foi mencionado no capítulo anterior, as tarefas da empresa foram classificadas em :

1. *Tarefas Fixas.*
2. *Tarefas Não Fixas.*

1.- Tarefas Fixas : Uma tarefa é classificada como fixa quando não existem possibilidades de variações na jornada de trabalho estabelecida, ou seja, quando a tarefa não depende de fatores imprevisíveis que possam provocar algum tipo de extensão na sua execução, garantindo assim o estrito cumprimento dos horários de trabalho estabelecidos. Portanto, as tarefas fixas são todas aquelas que são realizadas sem a possibilidade de atraso na execução.

2.- Tarefas Não Fixas : No sentido oposto, uma tarefa classificada como não fixa é aquela que está sujeita a variações no tempo de sua execução, possibilitando atrasos na jornada de trabalho.

Geralmente, estas características estão relacionadas com viagens de trens. Neste caso, algum dos principais fatores (perturbações) que podem provocar atrasos na jornada de trabalho incluem :

1. Defeitos ou falhas no trem,
2. Atrasos causados por interferências entre trens que circulam na mesma linha,
3. Condições operacionais inadequadas para o trânsito em vias deterioradas, entre outras.

Todas estas situações podem levar ao desequilíbrio de uma programação. Por isto, precisa-se de uma análises rigorosa na alocação das tarefas a serem incluídas em uma seqüência depois de uma tarefa da categoria não fixa. Na seção 3.3.1, apresenta-se outros detalhes baseados nestes fatos.

3.2.3.2.- Classificação das tarefas segundo os horários

Segundo os horários de trabalho pré-estabelecidos por uma empresa, cada tarefa pode ser classificada como :

1. *Tarefa Diurna,*
2. *Tarefa Vespertina,*
3. *Tarefa Noturna,*
4. *Tarefa Noturna Extensiva.*

De forma similar à classificação anterior, estas classificações foram estabelecidas para o desenvolvimento da ferramenta implementada e dependem muito da natureza do problema em particular.

1.- Tarefa Diurna. Em geral, o período diurno de trabalho fica estabelecido entre as 5:00 h da manhã até as 22:00 h. O resto das horas que completariam as 24:00 horas do dia correspondem ao período de trabalho noturno. As tarefas diurnas são aquelas cuja jornada de trabalho prefixada não atinge o período noturno estabelecido, ver Figura 3.3.

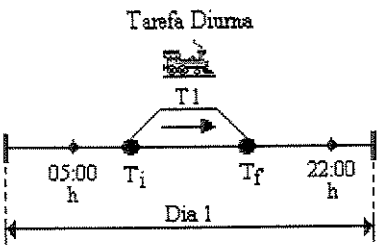


Figura 3.3. Representação de uma tarefa diurna.

2.- Tarefa Vespertina. As tarefas vespertinas podem atingir parte do período noturno, mas a jornada de trabalho se inicia e acaba no mesmo dia, Figura 3.4.

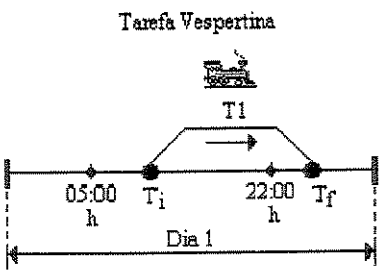


Figura 3.4. Representação de uma tarefa vespertina.

3.- Tarefa Noturna. São aquelas que podem atingir todo o período noturno, mas não provocam uma fora de escala no dia seguinte. Ou seja, a tarefa noturna é aquela em que mesmo iniciando num dia e acabando em outro, a equipe de trabalho terá tempo

suficiente para começar outra tarefa, sem violar as restrições de descanso entre tarefas consecutivas. Uma representação para este tipo de tarefa é mostrada a seguir.

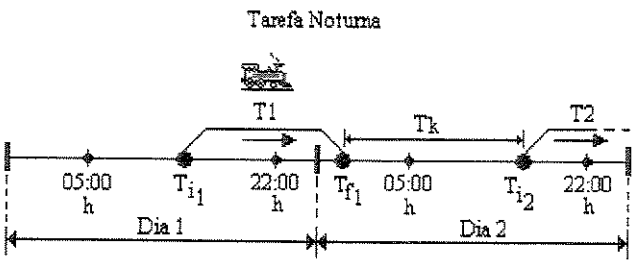


Figura 3.5. Representação de uma tarefa noturna.

Segundo a Figura 3.5, temos que $[\ T_k = Ti_2 - Tf_1\ ] \geq T_d$; onde T_k é o tempo resíduo entre as tarefas 1 e 2 e T_d é o tempo mínimo de descanso, regulamentado entre duas tarefas consecutivas.

4.- Tarefa Noturna Extensiva. As tarefas noturnas extensivas são aquelas que, além de serem noturnas, o horário final previsto implica que o trabalhador fique sem programação no dia seguinte, devido à restrição do período de descanso entre tarefas consecutivas, conforme ilustra a Figura 3.6.

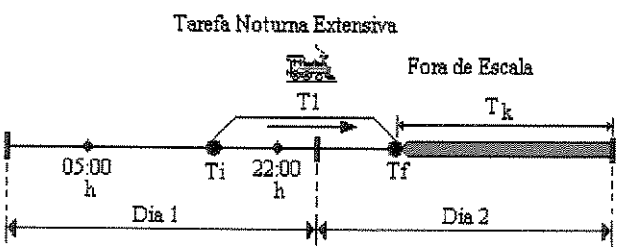


Figura 3.6. Representação de uma tarefa noturna extensiva.

Neste caso, a diferença entre as 24:00 horas do dia e o tempo final da tarefa (Tf), resulta em um valor inferior ao tempo mínimo de descanso estabelecido como restrição, ou seja, $[\ T_k = 24 - Tf\ ] < T_d$.

3.3.- Objetivos e restrições

Hoje em dia, um dos principais objetivos nas empresas consiste em elevar ao máximo a eficiência na produtividade, levando em consideração o nível de satisfação dos trabalhadores, estes em geral responsáveis pelo desempenho da produção. Neste caso, está em evidência a necessidade de elaborar escalas de tal forma que seja minimizado o custo de produção sem afetar o ritmo biológico dos seus trabalhadores. Para isto, precisa-se estabelecer uma análise extremamente rigorosa para obter o máximo de equilíbrio entre as cargas de trabalho programadas. No entanto, existem diversos fatores que limitam as possibilidades ideais de tal empenho. As restrições inerentes a cada problema geralmente influem e determinam um alto grau de complexidade dos processos de análises e elaboração das escalas.

A seguir, relacionamos, em ordem de prioridade, restrições típicas encontradas nas empresas ferroviárias brasileiras.

1.- O intervalo de tempo entre duas tarefas consecutivas geralmente é de 10 a 16 horas, mas, segundo os objetivos antes mencionados, sempre que a demanda não seja muito exigente, o valor desejável é, no mínimo, 16 horas de descanso.

2.- Para cada trabalhador devem ser designadas no mínimo 4 folgas em períodos mensais, sendo que uma delas deve ser alocada em um domingo em períodos de 45 dias.

3.- É desejável que o número máximo de tarefas noturnas repetidas não seja maior que 3.

4.- A cada 15 dias, cada trabalhador deve realizar um número de horas noturnas menor do que o número de horas diurnas. Geralmente, assume-se como horas noturnas aquelas entre as 22:00 e 05:00 horas, considerando o restante como horas de trabalho diurno.

3.3.1.- Outros critérios de análise para a alocação de tarefas

Além das restrições, existem outros fatores que influem e aumentam o grau de dificuldade da análise e elaboração dos planos de trabalho.

Os principais critérios giram em torno dos dias de folgas. Por exemplo, quando vai ser analisada a alocação de uma tarefa em um dia anterior ao dia de descanso (folga), não deve ser alocada uma tarefa noturna extensiva, ou uma tarefa noturna (seção 3.2.4) com jornada de serviço de mais de 8 horas de duração, evitando que o início da folga esteja em horários muito avançados do dia. Outro critério prático concerne às tarefas que devem ser alocadas depois de um dia de folga. Como já foi mencionado, os tempos de início e fim das folgas são dinâmicos e sempre dependem da finalização da tarefa que lhe antecede. Se o dia de folga analisado acaba depois das 05:00h, qualquer tarefa que comece depois desse horário poderá ser designada. Em caso contrário, ou seja, se o horário de término da folga for antes das 05:00h, só poderão ser designadas aquelas tarefas que tenham um horário de início posterior às 05:00h.

Por outro lado, para alocar qualquer tipo de tarefa após uma tarefa não fixa, devem ser analisados alguns critérios práticos, como por exemplo: designação de um tempo adicional ao tempo de duração (jornada) da tarefa, tempo que também pode ser visto como um tempo adicional ao descanso entre tarefas consecutivas. Isto tem o propósito de compensar o efeito de perturbações eventuais, como aquelas mencionadas na seção 3.2.3.1. Este tempo adicional é limitado pela demanda de trabalho, sendo que para demandas menores, atribui-se um intervalo de tempo maior como prevenção (e vice-versa).

3.4.- Metodologia usadas na programação de escalas

Atualmente, algumas empresas ferroviárias usam metodologias de programação não automatizadas (processamento manual) para gerar os planos de trabalho, geralmente elaborados a cada mês. Esta situação atual implica em algumas desvantagens, tanto para o pessoal encarregado de elaborar o escalonamento como para os trabalhadores.

Apesar de serem analisadas e realizadas por pessoal especializado, estas programações apresentam vários inconvenientes, entre eles a carga cognitiva exigida para se realizar de forma eficiente e detalhada a verificação de todas as restrições e a qualidade da escala.

De fato, a realização de uma programação mensal é tediosa e complicada, principalmente quando o número de trabalhadores e o número de tarefas é grande para ser resolvido através de um processamento não automatizado.

Apesar disto, algumas empresas realizam estes procedimentos através de uma metodologia simples, a qual demanda menos tempo para a geração das escalas. O método foi desenvolvido na prática a partir das próprias características do problema. Descreve-se a seguir a concepção do método, tentando esclarecer o porquê dos passos que o compõe.

Inicia-se com prévio conhecimento do número total de trabalhadores M. Este número de trabalhadores impõe um valor limite no número de passos em uma seqüência de tarefas a ser programada. Esta seqüência, chamada de seqüencial de tarefas, constitui o primeiro passo para a solução. As seqüências são estabelecidas em forma de um vetor linha de tamanho M, o número total de trabalhadores. Um exemplo deste esquema de seqüenciamento já foi discutido no capítulo anterior (2), seção 2.2.1.2. No entanto, será mostrado um novo exemplo para esclarecer pontos não abordados anteriormente.

Suponhamos que o número total de trabalhadores seja 12, e que o número de tarefas a programar seja 8. Suponha também que o horizonte de planejamento seja de 10 dias e que as restrições estejam relacionadas de forma igual à apresentada na seção 3.3.

O processo consiste em uma alocação seqüencial de tarefas, onde a posição de cada tarefa na seqüência depende das características da tarefa que lhe antecede. Isto significa que para alocar uma nova tarefa, o intervalo de tempo entre o início da tarefa a alocar e o final da tarefa anterior, deve ser maior do que o valor estabelecido para o descanso entre tarefas consecutivas. Caso este intervalo seja inferior, outra tarefa deverá ser considerada segundo o mesmo procedimento. Este processo é repetido até se conseguir alocar todas as tarefas, conformando finalmente um seqüencial de tarefas. Se por acaso todas as tarefas não puderem ser alocadas, devido às restrições, uma reestrutura dentro do seqüencial é realizada, tentando fazer uma troca na ordem das tarefas, que garantam uma melhor distribuição para que se possa designar as tarefas ainda não alocadas.

Como podemos verificar, a programação é razoavelmente simples, sempre que o número de tarefas e trabalhadores não é muito grande. Na prática, nem sempre se trabalha com conjuntos de variáveis e dados em forma reduzida. Na maioria dos problemas reais, o número de tarefas e trabalhadores pode aumentar conforme as

demandas. Outros fatores limitantes são as restrições, as quais limitam o número de combinações possíveis para gerar seqüenciais factíveis, desaconselhando qualquer metodologia não automatizada.

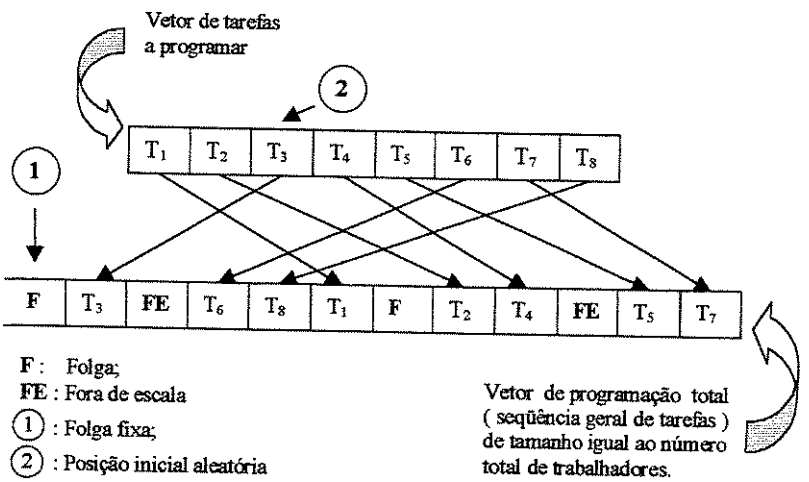


Figura 3.7. Geração do seqüencial de tarefas (escala de serviços).

Na Figura 3.7, observamos alguns detalhes já mencionados em seções anteriores. Geralmente, inicia-se o processo alocando uma folga. Isto permite ter um maior controle na alocação das folgas seguintes. A alocação da primeira tarefa (T_3) foi baseada em um critério relacionado com o horário de início desta tarefa. Neste caso, a tarefa T_3 deverá ter um tempo de início superior às 05:00h para evitar demoras no começo do serviço já que o dia anterior foi um dia de folga.

Segundo o seqüencial de tarefas (Figura 3.7), percebemos que as tarefas T_3 e T_4 , são tarefas noturnas extensivas e o resto podem ser diurnas, vespertinas ou noturnas.

Na etapa final do processo de alocação podem ocorrer situações não previstas. Pode acontecer que, dado um determinado número de tarefas e conhecendo o número de trabalhadores, a seqüência geral programada não complete o número de trabalhadores ou pelo contrário, pode ocorrer que o número de tarefas analisada na formação da seqüência total, supere o valor limite. Neste caso devem ser realizadas outras operações de reestrutura da seqüência. No Capítulo 4, se explicam mais em detalhes estas duas situações.

Uma vez confeccionado o seqüencial de tarefas, o próximo passo é a construção do programa de escalas propriamente dito. A Figura 3.8 mostra um exemplo do esquema utilizado para este propósito.

A geração de todas as escalas (seqüências de tarefas correspondentes a cada trabalhador) se inicia selecionando M posições no seqüencial (diferentes entre si). A partir de cada posição, são selecionadas M subseqüências de tarefas, truncando a seleção em um valor correspondente ao número de dias a programar (horizonte de planejamento). Finalmente, o conjunto de todas estas subseqüências conforma a programação total das escalas.

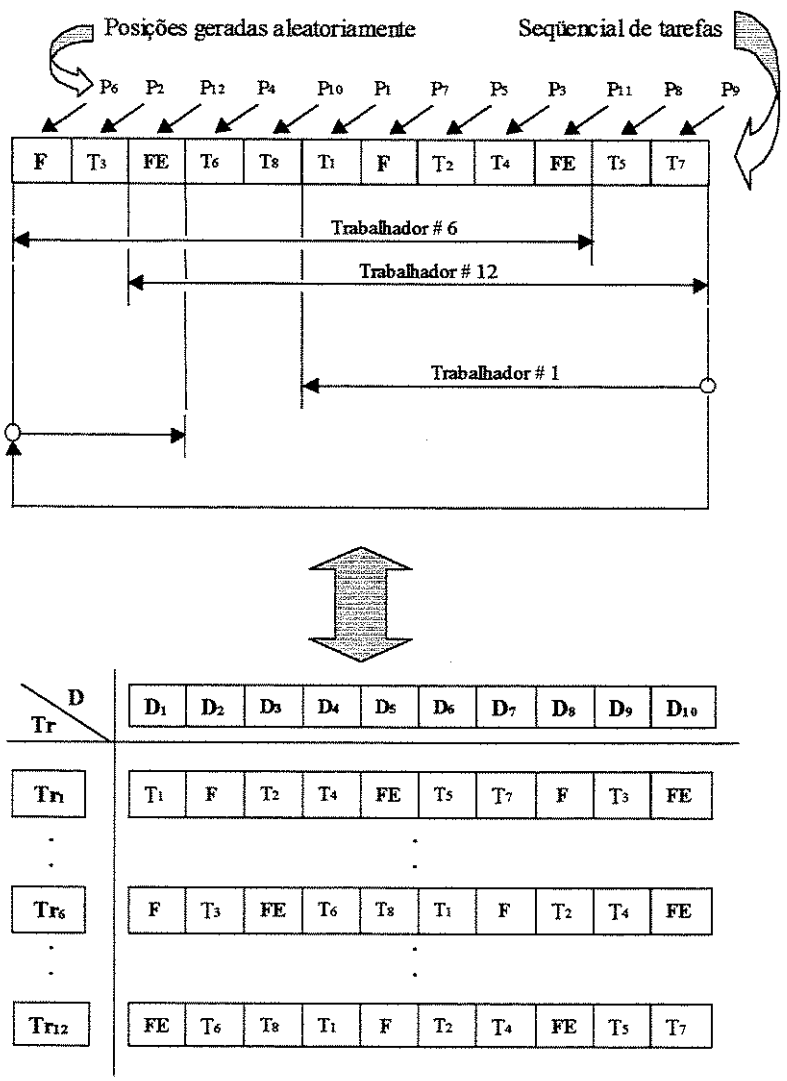


Figura 3.8. Programação das escalas mensais a partir do seqüencial de tarefas.

3.5.- Resumo

Diferentes características foram expostas durante este capítulo com o propósito de estabelecer e posicionar os conceitos, critérios e demais aspectos relacionados aos problemas de programação de escalas.

Neste capítulo, enfatizou-se o sistema de transporte ferroviário, em particular o sistema ferroviário brasileiro, um dos focos centrais deste trabalho.

O fato de ter que lidar com tarefas que possuem horários e jornadas irregulares implica em uma maior complexidade na elaboração dos programas de escala.

Apesar de ser usado um método manual, aparentemente simples, e de serem satisfeitas todas as restrições regulamentadas, os resultados observados nas escalas geradas não representam, necessariamente, um percentual elevado de satisfação dos trabalhadores envolvidos.

O problema é que, ao elaborar as escalas, é muito difícil estabelecer comparações com programações anteriores, de forma a balancear a escala atual e harmonizá-la com escalas anteriores.

Capítulo 4

Algoritmos Evolutivos na Geração de Escalas

4.1.- Introdução

O presente capítulo apresenta definições, conceitos e algumas formalizações concernentes a estruturas de algoritmos evolutivos. É estabelecido um plano comparativo e extensivo de cada etapa que compõe um algoritmo evolutivo, quando aplicado em problemas de natureza combinatória restrita. Nesta instância, aspectos básicos como a codificação dos cromossomos, os operadores genéticos de recombinação e a função de avaliação são os mais enfatizados.

A respeito da aplicação desenvolvida, a primeira grande barreira foi a questão da codificação das soluções do problema. Cada solução deve satisfazer várias restrições, além de alguns critérios e condições adicionais necessários para assegurar o bem estar dos trabalhadores.

O modelo de representação das soluções foi baseado em duas variantes de codificação. A primeira é obtida através de um algoritmo de busca em *Backtracking* com processamento heurístico, garantindo todos os requisitos, sejam restrições ou critérios adicionais, seção 3.3. Nesta primeira fase são gerados os seqüenciais de tarefas explicadas na seção 3.4. A segunda variante obtém a codificação das soluções através de um construtor de escalas, usando critérios aleatórios ou heurísticos, segundo o contexto

da aplicação. O resultado desta segunda etapa constitui a geração das escalas mensais, o objetivo principal do trabalho.

Outros estudos referem-se à função de avaliação. Uma primeira abordagem é modelada usando uma metodologia conhecida, o método de soma ponderada dos parâmetros de otimização. Uma outra abordagem considerada usa conceitos da Lógica *Fuzzy*. Nesta nova metodologia de avaliação, os parâmetros de otimização são considerados como conjuntos *fuzzy*, sendo que determina-se a confluência entre eles para tomar decisões. Neste caso, tais decisões referem-se à seleção dos indivíduos mais adaptados em cada população do algoritmo.

Assim, todos estes elementos se constituem no conjunto para formar a estrutura operacional do algoritmo evolutivo desenvolvido.

Algumas vantagens e desvantagens das metodologias desenvolvidas no trabalho serão comentadas, visando um maior entendimento das peculiaridades apresentadas.

4.2.- Histórico sobre algoritmos evolutivos

Os algoritmos de computação evolutiva (CE) são ferramentas de otimização de propósito geral, baseadas na Teoria da Evolução de Darwin. Estas estruturas têm a capacidade de produzir soluções otimizadas até mesmo quando a dimensão do problema é elevada. Por esta razão eles têm sido aplicados com sucesso em uma grande variedade de problemas, entre eles problemas de busca e otimização em aplicações industriais.

Um conjunto variado de estruturas de programação conforma os algoritmos de evolução em termos gerais. Entre eles podemos mencionar os Algoritmos Genéticos, introduzidos em meados de 1975 por John Holland e seus colaboradores da Universidade de Michigan [15]; a Programação Evolutiva, desenvolvida nos Estados Unidos, nos anos 60, por L. J. Fogel, A. J. Owens e M. J. Walsh [13] e as Estratégias Evolutivas, introduzidas na Alemanha, nos anos 60, por I Rechenberg [29] e H. P. Schwefel [30], seguidos por Rudolph Beyer e outros colaboradores.

Embora recentes, estas técnicas têm alcançado enorme sucesso na solução de problemas de otimização. Neste contexto, grande parte das referências consultadas

concernem aplicações de algoritmos genéticos e estratégias evolutivas em problemas técnicos [12, 17, 18, 22, 25, 33, 34].

4.3.- Aspectos básicos sobre algoritmos evolutivos

Freqüentemente, a literatura classifica os algoritmos evolutivos como Metaheurísticas. Estas metodologias têm superado algumas deficiências históricas dos algoritmos convencionais de busca, como por exemplo, o problema da convergência a pontos ótimos locais, que geralmente fornecem soluções muito distantes da melhor solução do problema tratado.

Um aspecto importante dos problemas práticos consiste em alguma forma de busca, onde, dada uma certa coleção de elementos, deseja-se encontrar um ou mais elementos que atendam a algumas condições previamente especificadas. O caso mais geral, entretanto, é a busca, num certo conjunto, de uma solução para um dado problema.

Cada um destes elementos, no contexto de algoritmos evolutivos, eqüivale a um cromossomo, que formará parte do espaço de soluções de um dado problema. Assim, cada membro da população (cromossomo) consiste em um número de genes (unidades de informação). Soluções são obtidas pela recombinação genética entre os membros da população para produzir novos filhos ou pela alteração dos existentes. Uma simulação da Seleção Natural acontece avaliando a qualidade de cada candidato à solução e definindo aqueles que sobreviverão para as próximas gerações.

De acordo com sua generalidade, os algoritmos de busca podem ser classificados como fracos, quando podem ser aplicados a uma gama variada de problemas, ou fortes, quando são projetados para aplicações específicas. À medida que aumenta o conhecimento sobre o espaço de busca, especializam-se os algoritmos de busca para este espaço, com o aumento de eficiência e perda de generalidade, ou seja, pode-se passar de algoritmos mais fracos e genéricos para algoritmos mais fortes e específicos. Geralmente, os algoritmos de busca devem apresentar, com um certo compromisso, duas características desejáveis : o algoritmo deve ser capaz de identificar em diversas regiões (no seu espaço de busca) soluções em qualidade e quantidade (etapa de exploração). Por outro lado, logo depois de localizar a região mais promissora, é necessário estudar cada

solução para obter um resultado eficiente, utilizando o mínimo de recursos (etapa de explotação). Assim um algoritmo de busca deve possuir a melhor combinação possível de suas capacidades de exploração e explotação.

Em problemas de otimização, para encontrar um extremo global (máximo ou mínimo), ou qualquer solução razoavelmente boa, requer-se uma boa capacidade de exploração do espaço de busca. Os algoritmos clássicos de programação matemática são freqüentemente projetados para problemas unimodais (apenas um extremo local e global), sendo essencialmente voltados ao aspecto da explotação. Quando aplicados a um problema multimodal, sua convergência, na maioria dos casos, se dará para um extremo local, que dependerá do ponto de partida utilizado.

Por estas razões, foi necessária a consideração de novas técnicas de busca e otimização. Em particular, os algoritmos de evolução têm como uma das suas principais características, a possibilidade de escapar dos pontos de mínimo local, pois conseguem explorar um maior número de pontos do espaço de soluções.

Em comparação com as técnicas e algoritmos matemáticos de otimização, existem alguns termos equivalentes, conforme resumido na Tabela 4.1.

Algoritmos evolutivos	**Algoritmos matemáticos**
População de cromossomos	Conjunto de soluções
Cromossomo	Solução
Gene	Componente de uma solução

Tabela 4.1. Similaridades entre os algoritmos evolutivos e matemáticos.

Todos os algoritmos evolutivos desenvolvidos até nossos dias apresentam características similares. A seguir, relacionam-se as principais etapas que os caracterizam.

4.4.- Etapas características de algoritmos evolutivos

- Encontrar uma codificação genética das soluções viáveis do problema (representação do cromossomo).
- Definição de parâmetros tais como o tamanho da população, critérios de recombinacão (probabilidade para efetuar mutação e crossover), critérios de seleção, entre outros.
- Determinação de uma população de cromossomos.
- Definição de uma função de avaliação.
- Definição de operadores genéticos para a reprodução de novos cromossomos.
- Determinação dos critérios de parada para a evolução das gerações.

4.4.1.- Codificação genética das soluções

Esta primeira etapa consiste em estabelecer um esquema de codificação (representação), para cada elemento do espaço de busca em questão, já que estes algoritmos não operam diretamente sobre as soluções do problema, e sim sobre uma codificação das mesmas.

Desta forma, uma solução de um dado problema está associada a um cromossomo c representado na forma de um vetor com n posições, onde cada elemento x_i representa um gene ou unidades de informação, Figura 4.1.

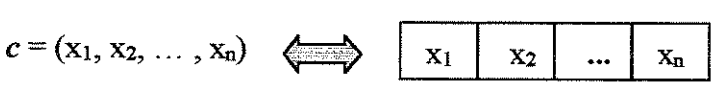


Figura 4.1. Representação típica de um cromossomo.

Conforme variam as características dos problemas, diferentes tipos de codificação são usualmente implementados. Em geral é usada a codificação binária, mas algumas aplicações requerem outras representações como codificação real, inteira, Figura 4.2.

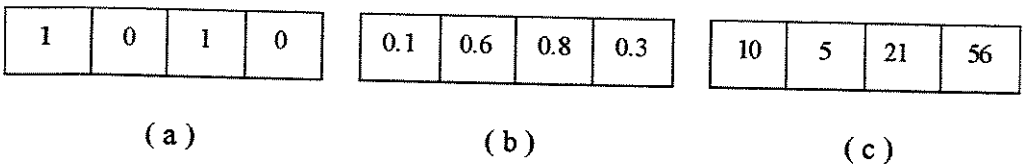


Figura 4.2. Tipos de codificação. (a) Binária. (b) Real. (c) Inteira.

Em aplicações de programação de escalas, geralmente são discutidas variadas formas para representar as soluções dos problemas. Cada representação depende das peculiaridades destes problemas, mas na maioria dos casos utiliza-se a codificação inteira. Por exemplo, Tanomaru [31] usa este tipo de codificação, onde cada gene que compõe o cromossomo é um identificador que contém a informação dos planos de trabalho designados a cada trabalhador em um determinado horizonte de planejamento.

Aplicações similares de escalonamento, como os problemas de *Job Shop Scheduling* e *Flow Shop Scheduling*, também usam identificadores inteiros para representar os genes do cromossomo [33, 17]. Van Bael *et al.* [34] representa cada *Job* (seqüência de tarefas) distribuído em vários genes, onde cada gene representa uma tarefa desta seqüência.

4.4.2.- Definição de parâmetros

Os parâmetros geralmente definidos no algoritmo são o tamanho da população (número de indivíduos) e as probabilidades para realizar operações de mutação e *crossover*. A seleção do valor para o tamanho da população não tem critério fixo. Em geral, este valor depende de um compromisso entre o tempo de convergência e o esforço computacional associado. As probabilidades de mutação e *crossover* geralmente são usadas em valores diferenciados devido ao efeito causado por cada uma nos processos de recombinação dos cromossomos. Costuma-se usar valores de probabilidade de mutação inferiores aos da probabilidade de *crossover* [25].

Outros critérios são referidos ao aumento da diversidade populacional, seleção de indivíduos das próximas gerações e seleção dos indivíduos que serão considerados para recombinação genética.

Na seleção dos indivíduos que formarão as próximas gerações são usadas diversas estratégias, tais como seleção elitista, seleção por roleta (*roulette wheel*), por torneio, seleção por heurísticas, ou simplesmente aleatória.

4.4.3.- População de um algoritmo evolutivo

A população de um algoritmo evolutivo constitui-se de determinados conjuntos de soluções viáveis, representadas através de cromossomos nas variadas formas de codificação.

Os procedimentos para gerar populações são mais simples que na maioria dos problemas de otimização, mas dependem muito das característica e da natureza do problema. Normalmente, utiliza-se desde procedimentos aleatórios até algoritmos heurísticos para este fim, caso sejam tratados problemas que envolvem restrições.

4.4.4.- Função de avaliação

A função de avaliação ou função objetivo é responsável pelo processo de seleção dos cromossomos. Através desta função obtém-se o nível de qualidade de cada cromossomo da população (candidato à solução), sendo sua forma dependente das características específicas de cada problema. No caso dos problemas de otimização, ela está intimamente ligada à função que se deseja minimizar ou maximizar. Referente a problemas variados de escalonamentos existem diversas formas de avaliar cada cromossomo (solução). Por exemplo, em Tanomaru [31], cada cromossomo recebe uma pontuação penalizando as suas características de não factibilidade. A função de avaliação calcula a qualidade invertendo essa pontuação. Assim, obtém-se aquele que possui as melhores características (mais factível) com respeito aos outros indivíduos analisados. Uckun *et al.* [33] calculam a média dos tempos de funcionamento das máquinas. Logo, selecionam o melhor indivíduo (melhor escala ou seqüência de tarefas realizadas em cada máquina) determinando aquela solução que apresenta o menor ou maior tempo, segundo a determinação da condição para ser o vencedor.

Desta forma, percebemos que a função de avaliação não tem uma estrutura genérica. Tudo vai depender sempre da natureza do problema, o qual vai determinar os requisitos e objetivos finais para a obtenção da qualidade das soluções tratadas em cada caso.

4.4.5.- Operadores genéticos

Os operadores genéticos estão fortemente ligados à representação de uma solução do problema original. Os procedimentos básicos para a recombinação genética são realizados via operadores de cruzamento e mutação. Os operadores de cruzamento, conhecidos como *crossover*, realizam transformações genéticas entre dois cromossomos para gerar um ou mais descendentes. Por outro lado, os operadores de mutação alteram as características no nível dos genes.

4.4.5.1.- Operador de crossover

O operador de *crossover* geralmente é usado para recombinar material genético de dois "cromossomos pais" e gerar um ou mais "cromossomos filhos".

O processo consiste em selecionar posições de cruzamento nos cromossomos pais e, dependendo das características do operador, gerar os filhos, de tal forma que cada filho contenha características genéticas de ambos os pais.

Por exemplo, sejam C1 e C2 dois cromossomos pais e p1 e p2 duas posições de cruzamento selecionadas convenientemente. Os filhos F1 e F2, podem ser obtidos através das diferentes estratégias de cruzamento mostradas na Figura 4.3. No caso das operações cruzadas (troca cruzada de gens) deve-se garantir a factibilidade dos cromossomos filhos resultantes. Outras variantes de *crossover* podem ser realizadas determinando dois pontos de cruzamento em cada cromossomo pai, mas estas operações são similares ou usam critérios baseados nas mesmas estratégias mostradas na Figura 4. 3.

Freqüentemente, a simples utilização destes operadores clássicos não conduz a uma solução factível. Por tal razão, é necessário incorporar critérios e regras adicionais que permitam viabilizar tais soluções.

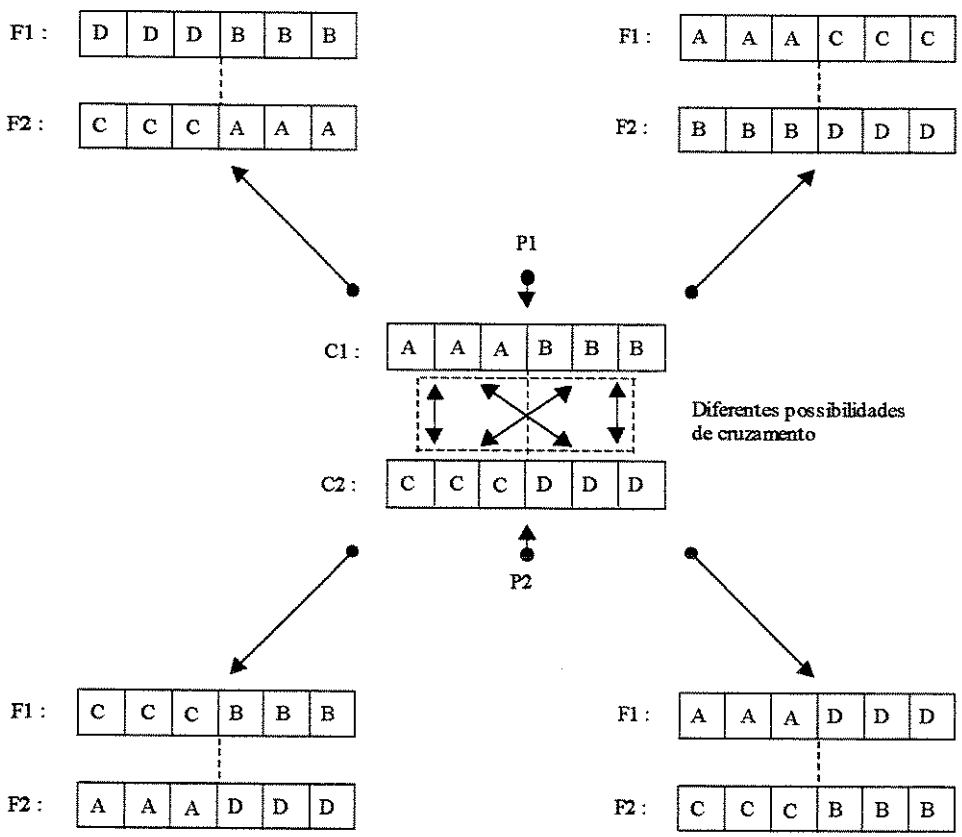


Figura 4.3. Variantes de cruzamento genético.

Em aplicações de programação de escalas, os operadores de *crossover* variam conforme o problema e a codificação das soluções. Na maioria dos casos, estes operadores funcionam em domínios específicos, devido às restrições inerentes aos problemas tratados. Uckun *et al.* [33] resolvem um problema de seqüenciamento de tarefas (*Job Shop Scheduling*) aplicando operadores ortogonais em que o contexto de cada operador está determinado pela representação das soluções, as quais definem o escopo de aplicação.

Contudo, operadores desta classe se apresentam em contextos muito específicos de aplicação, perdendo em generalidade, mas ganhando em factibilidade no caso de problemas altamente restritos, onde operadores de caráter mais genérico não conseguem bons resultados, ou simplesmente não são aplicáveis.

4.4.5.2.- Operador de mutação

O operador de mutação é o encarregado de realizar mudanças ou alterações de um ou vários genes de um cromossomo. Geralmente, esta operação baseia-se ou em uma seleção aleatória ou em algum critério predefinido.

O tipo de mutação também varia de um para outro problema. Esta operação está ligada à forma de representação dos cromossomos. Quando é usada a codificação binária, o caso típico de mutação refere-se á mudança de estado, de 0 para 1 ou vice-versa, Figura 4.4. Quando as soluções são representadas em codificação real, costuma-se adicionar um certo valor, em geral baseado em alguma regra (exemplo : adição de números aleatórios normalmente distribuídos) ou algum outro critério específico, heurístico ou simplesmente aleatório, Figura 4.5. No caso da codificação inteira, geralmente, a mutação exige a troca de posição para dois genes selecionados, Figura 4.6.

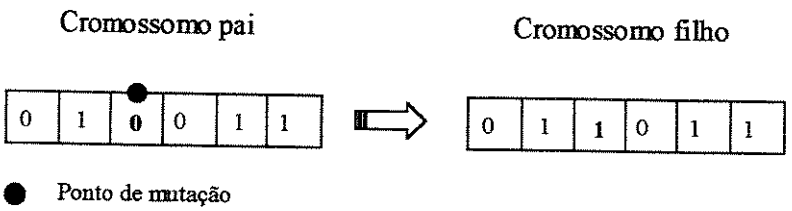


Figura 4.4. Mutação para codificação binária.

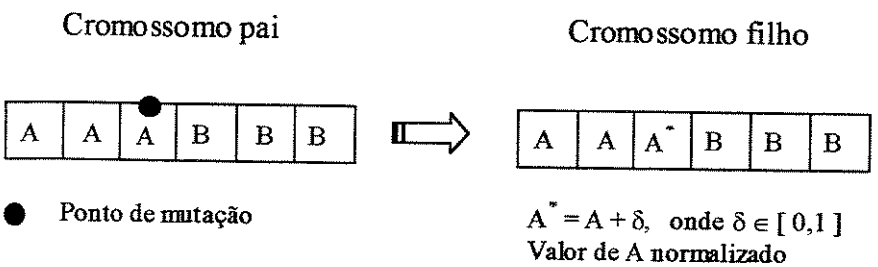


Figura 4.5. Mutação para codificação real.

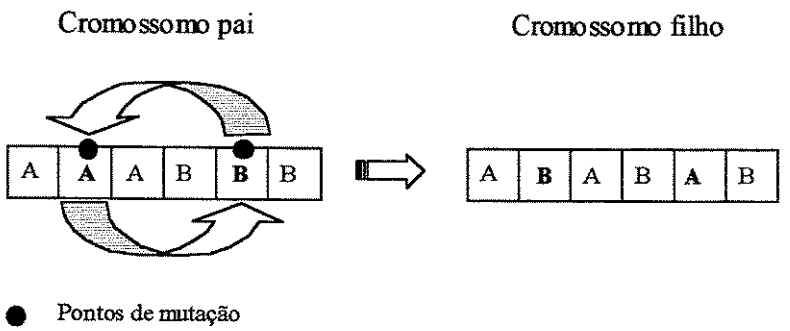


Figura 4.6. Mutação para codificação inteira.

Outros tipos de mutação são induzidas a partir destas análises, mas os procedimentos são basicamente baseados nas idéias apresentadas, Ishibuchi *et al.* [17].

Ambas as operações, ou seja *crossover* e mutação, apresentam outras dependências quando se trata de problemas que envolvem restrições. Neste caso, é necessário estabelecer certos critérios ou limites quanto às variações que serão realizadas em cada cromossomo, para que se garanta a factibilidade das soluções resultantes.

4.4.5.3.- Métodos de seleção

Nos algoritmos evolutivos, os métodos de seleção são inspirados na seleção natural que ocorre nos processos evolutivos dos organismos vivos. Estes mecanismos são geralmente usados depois dos procedimentos de recombinação e/ou para determinar aqueles cromossomos que serão objetos destas transformações.

Para a seleção de novas gerações populacionais, os métodos mais utilizados são a seleção via roleta (*roulette wheel)*, seleção elitista, seleção por torneio [25].

Na seleção dos cromossomos que sofrerão efeitos de recombinação ou mutação, podem ser usados métodos de seleção aleatória, heurística ou por algum dos métodos mencionados anteriormente.

4.5.- Descrição de um algoritmo evolutivo para geração de escalas

O algoritmo computacional implementado foi baseado em técnicas de estratégias evolutivas (EE) [25]. Neste sentido, o algoritmo permite a seleção de várias estratégias, tais como : (1+1)-EE, (μ+λ)-EE e (μ,λ)-EE. A principal diferença entre elas está no processo de recombinação e seleção dos indivíduos das próximas gerações. A estratégia de (1+1)-EE baseia-se na seleção de um cromossomo pai para gerar um filho. Logo é estabelecido um critério de sobrevivência para manter na população aquele que apresente maior adaptação (melhor avaliação). No caso da estratégia (μ+λ)-EE, μ cromossomos pais são selecionados para gerar λ cromossomos filhos. Neste caso, tanto os pais como os filhos competem pela sobrevivência. Por último, a estratégia (μ,λ)-EE apresenta a mesma relação de cromossomos pais e filhos, mas os cromossomos pais são sempre substituídos pelos cromossomos filhos.

No contexto de programação de escalas, o algoritmo evolutivo tem por objetivo resolver um problema de otimização sujeito a restrições. A idéia é balancear a carga de trabalho designada a cada trabalhador.

Como foi mencionado, como resultado da geração das escalas, cada trabalhador deve receber uma seqüência de tarefas correspondente ao horizonte de planejamento, sendo que o conjunto de todas as seqüências designadas para cada trabalhador compõe o planejamento total, ou mais especificamente, a programação mensal das escalas.

Em cada planejamento mensal, cada seqüência de tarefas apresenta diferenças a respeito dos horários de trabalho e descansos programados, sendo esta diferença o principal ponto de interesse para a estratégia de otimização.

Segundo o discutido no Capítulo 1, os objetivos fundamentais de otimização, dentro do problema de geração de escalas, envolvem : horas diurnas, horas noturnas, horas de descanso e número de pernoites programados para cada trabalhador. Mais adiante, serão discutidos os procedimentos de otimização utilizados.

A implementação do algoritmo desenvolvido apresenta algumas características pouco comuns. Devido às restrições do problema, desenvolveram-se algumas variantes de solução conforme foram sendo analisados o projeto e os procedimentos da ferramenta computacional.

O processo evolutivo realiza-se em duas fases. Na primeira fase, as populações são compostas por seqüenciais de tarefas, obtidos segundo os procedimentos explicados no Capítulo 3. A segunda fase, visa a construção das programações mensais de escalas, construídas a partir de cada seqüencial de tarefas gerado na primeira fase de evolução.

4.5.1.- Representação dos cromossomos

As duas fases evolutivas do algoritmo foram determinadas pela representação das soluções. Como mencionado, os cromossomos da primeira fase evolutiva foram os seqüenciais de tarefas, mostrados nas Figuras 4.7(a)(b). Neste caso, o tipo de representação é uma codificação indireta das soluções.

Na segunda fase de evolução estes seqüenciais são transformados em planos de trabalho mensal. Neste caso, os cromossomos são constituídos por estes planejamentos, representando assim uma codificação direta das soluções, Figura 4.7(c).

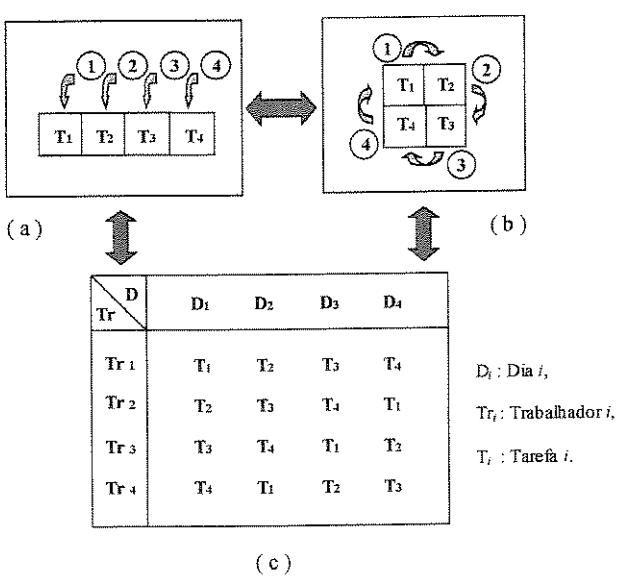


Figura 4.7. Formas de representação dos cromossomos. (a),(b). Representação indireta (seqüencial de tarefas). (c). Representação direta (programação mensal das escalas)

Nos cromossomos, cada gene é representado por um identificador de tarefas (Id), que sempre é um valor inteiro positivo. Na Figura 4.7, estes identificadores estão representados por T_i, i=1,..., 4.

Como foi visto na Tabela 3.2, os identificadores para os dias de descansos e dias sem programação foram 50 e 60 respetivamente. O resto das tarefas apresentam identificadores inteiros com certas peculiaridades, explicadas através do seguinte exemplo:

Supondo que uma determinada programação mensal engloba as seguintes tarefas :

- 5 tarefas com código 02,
- 10 tarefas com código 08,
- 15 tarefas com código 09.

No contexto da aplicação desenvolvida, a identificação destas tarefas foi estabelecida conforme mostra a Tabela 4.2.

No. Tarefa	Código	Identificador
T_1	02	11
T_2	02	12
⋮	⋮	⋮
T_5	02	15
T_6	08	21
T_7	08	22
⋮	⋮	⋮
T_{15}	08	210
T_{16}	09	31
T_{17}	09	32
⋮	⋮	⋮
T_{30}	09	315

Tabela 4.2. Estrutura dos identificadores segundo o número e o código da tarefa.

Segundo a Tabela 4.2, cada identificador contém duas informações. O primeiro dígito corresponde ao código da tarefa, definido desta forma para evitar seqüências contínuas de tarefas do mesmo código. No entanto elas equivalem a operações iguais de trabalho. Os dígitos restantes enumeram as tarefas do ponto de vista dos seus horários. Isto deve-se a um critério prático definido para evitar a repetição de uma tarefa dentro do seqüencial.

Assim, estes identificadores são alocados nestas seqüências, gerando os cromossomos, que correspondem aos indivíduos das populações desta primeira fase de evolução.

A seguir, daremos uma explicação detalhada sobre estas operações, conforme vão sendo apresentados os procedimentos utilizados para a elaboração da ferramenta computacional desenvolvida.

4.5.2.- Geração do seqüencial de tarefas

Um exemplo mais ilustrativo de um seqüencial de tarefas é mostrado na Figura 4.8. Junto com o detalhamento do exemplo, menciona-se a representação utilizada pela ferramenta computacional.

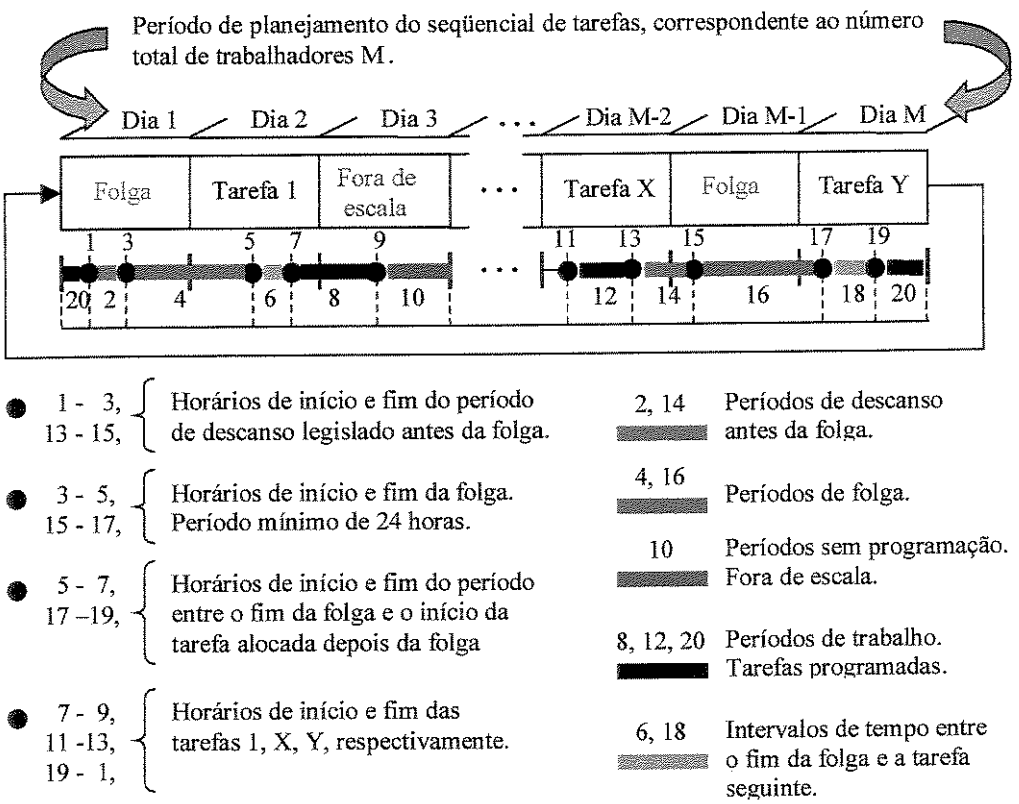


Figura 4.8. Exemplo típico de um seqüencial de tarefas.

Os dados principais são relacionados com as tarefas a programar. As tarefas são armazenadas em arquivos que contêm seus identificadores Id e seus horários iniciais e

finais. É bom destacar que, sempre que a tarefa não seja uma folga ou uma fora de escala, todos os identificadores são diferentes. Uma vez obtidos os dados, cria-se um vetor auxiliar de tarefas V_{AT} contendo somente os identificadores das tarefas a serem programadas.

O número de trabalhadores M, necessários para cada programação, é previamente estabelecido (valor fixado, dependente do quadro de funcionários da empresa).

Assim, todos os identificadores, contidos no vetor auxiliar anterior V_{AT}, devem ser alocados em um vetor de programação V_P, que conformará o seqüencial de tarefas.

Para finalizar este processo é preciso que o tamanho do vetor V_P seja igual a M, onde V_P corresponde ao número de atividades programadas e M corresponde ao número de trabalhadores. Geralmente, estes valores são diferentes. Por esta razão, são usados alguns critérios adicionais para complementar a geração e garantir a factibilidade de cada seqüencial resultante. Alguns destes critérios surgiram das primeiras experiências realizadas e serão logo explicados.

A criação do seqüencial de tarefas é um dos passos mais complexos e fundamentais dentro da programação de escalas em empresas ferroviárias. A seguir, relacionam-se as principais etapas de análises para a elaboração destas seqüências.

Etapas e critérios de elaboração de um seqüencial de tarefas

1. Início com folga fixa, seguida de uma tarefa com horário inicial avançado.
2. Identificação e alocação de tarefas noturnas extensivas.
3. Identificação e alocação do dia de folga.
4. Identificação e alocação de tarefas antes de um dia de folga.
5. Identificação e alocação de tarefas após um dia de folga.
6. Alocação das tarefas segundo o tempo de descanso e código das tarefas.
7. Reinício do processo de elaboração da seqüência, caso não seja possível a alocação de todas as tarefas.
8. Complementação da seqüência de tarefas obtida para compor o seqüencial de tarefas.
9. Validação do seqüencial de tarefas.

A ordem desta seqüência de passos influi e determina a qualidade do seqüencial, cumprindo todas as restrições explicadas no Capítulo 3.

A seguir, detalha-se os passos para a elaboração de cada seqüencial de tarefas.

1. Início com folga fixa, seguida de uma tarefa com horário inicial avançado.

A primeira alocação sempre é iniciada com uma folga fixa devido às condições que deve manter a repetição dos dias de descanso dentro das escalas. A alocação seguinte corresponde a uma tarefa que, por critérios práticos, foi estabelecida como uma tarefa com horário inicial marcado para as horas mais avançadas do dia. Isto está relacionado com as alocações desejadas para depois dos dias de folga, sendo que não deve ser designada uma tarefa cujo horário inicial seja anterior às 5:00 h.

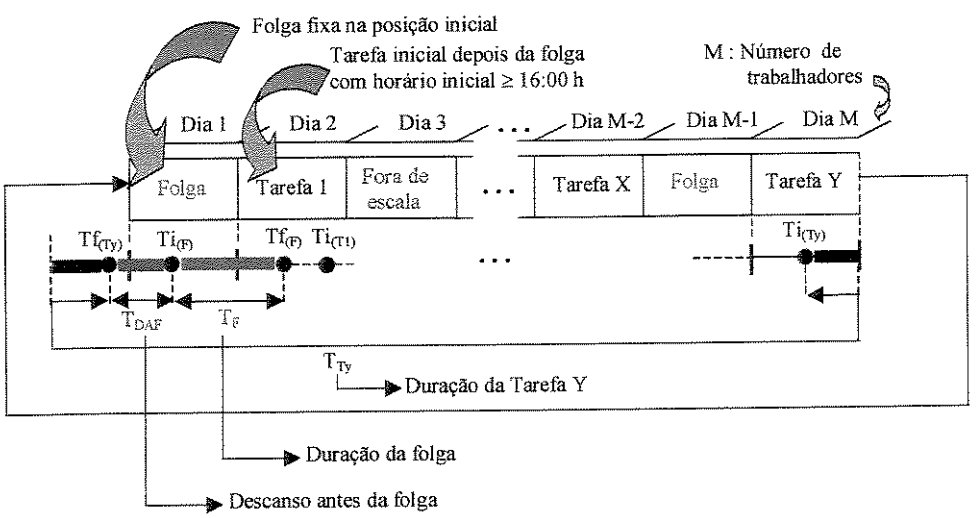


Figura 4.9. Interrelação entre as duas primeiras alocações e a última alocação no seqüencial.

A Figura 4.9 mostra alguns dos critérios práticos utilizados nas primeiras alocações no seqüencial de tarefas. O fato de alocar uma tarefa depois da primeira folga, com um horário inicial superior às 16:00 h, deriva-se das análises referentes à interrelação entre a última tarefa alocada na seqüência e a primeira tarefa da própria seqüência (neste caso a folga). Assim, evita-se uma possível violação do tempo de descanso estabelecido entre o

tempo final da tarefa designada antes da folga e o tempo inicial da tarefa a designar depois da folga.

A partir das duas primeiras alocações (folga e tarefa 1), é realizado todo um procedimento analítico que inclui a verificação de grande parte das restrições, assim como a verificação de critérios adicionais no sentido de obter soluções factíveis.

2.- Identificação e alocação de tarefas noturnas extensivas.

Dentro das análises de restrições, a primeira verificação concerne às tarefas noturnas extensivas. Este passo está determinado pelos critérios usados para a elaboração do seqüencial, sendo que sempre existirá uma tarefa na posição anterior e a posição atual analisada não vai se corresponder com um dia de descanso pois já foi alocado na primeira posição da seqüência.

Primeiro se verifica se a tarefa alocada depois da folga é uma tarefa cuja jornada induz uma fora de escala no próximo dia. Como já foi explicado no Capítulo 3, neste passo calcula-se a diferença entre 24 h e o tempo final da tarefa, uma vez determinado que este tempo final atinge o dia seguinte. Caso contrário serão analisados os passos que seguem na seqüência de elaboração mostrada. O valor da diferença calculado é comparado com um parâmetro ajustável, referido como Horas_tarefas_extensivas H_{TE}, inicialmente fixado em um valor maior que o tempo de descanso obrigatório entre tarefas consecutivas. O ajuste dinâmico deste parâmetro tem a ver com a complementação do seqüencial de tarefas. No item 8, explica-se em detalhes as características deste procedimento.

3.- Identificação e alocação do dia de folga.

Este passo resulta obrigatório dentro da seqüência de tarefas. Neste caso, o dia de descanso é marcado pelas características da restrição referente aos dias de folga. Como foi mencionado no Capítulo 3, nosso problema apresenta uma restrição que impõe 4 ou mais dias de descanso em períodos mensais, sendo que para cada período de 45 dias analisados, um descanso deve ser designado em um domingo. A partir desta idéia,

surgem algumas variantes para solucionar este problema. Uma primeira seria a designação dos dias de descanso a cada 6 dias de trabalho, Figura 4.10 (a). No entanto, esta seqüência de descansos as vezes não é conveniente para a empresa pois significariam muitos dias de descanso durante o período planejado. Uma outra variante considera descansos a cada 8 dias, Figura 4.10 (b). Esta variante apresenta alguns detalhes que devem ser flexibilizados para garantir as condições exigidas pela restrição. Desta vez, segundo a seqüência, não foi possível alocar uma folga em um domingo dentro do período de 45 dias. Por esta razão, e usualmente por acordo mútuo entre os trabalhadores e a empresa, aloca-se duas folgas consecutivas a cada período de 7 semanas.

Dias	Dom.	Seg.	Ter.	Qua.	Qui.	Sex.	Sáb.
1-7	F	-	-	-	-	-	F
8-14	-	-	-	-	-	F	-
15-21	-	-	-	-	F	-	-
22-28	-	-	-	F	-	-	-
29-35	-	-	F	-	-	-	-
36-42	-	F	-	-	-	-	-
43-49	F	-	-	-	-	-	F
50-57	-	-	-	-	-	F	-
58-63	-	-	-	-	F	-	-
64-70	-	-	-	F	-	-	-

(a)

Dias	Dom.	Seg.	Ter.	Qua.	Qui.	Sex.	Sáb.
1-7	F	-	-	-	-	-	-
8-14	-	F	-	-	-	-	-
15-21	-	-	F	-	-	-	-
22-28	-	-	-	F	-	-	-
29-35	-	-	-	-	F	-	-
36-42	-	-	-	-	-	F	-
43-49	-	-	-	-	-	-	F
50-57	F	-	-	-	-	-	-
58-63	-	F	-	-	-	-	-
64-70	-	-	F	-	-	-	-

(b)

Figura 4.10. Seqüencial de tarefas. (a) Com alocação de folgas a cada 6 dias. (b) Com alocação de folgas a cada 8 dias.

4.- Identificação e alocação de tarefas antes de um dia de folga.

Se a posição atual de análise não corresponde a um dia de folga, verifica-se a correspondência com um dia antes da folga, Figura 4.11. Esta verificação deve-se a um critério prático.

Caso esteja se analisando um dia antes da folga, deverá ser alocada uma tarefa diurna, vespertina ou, em última instância, noturna. A alocação de uma tarefa noturna extensiva implicaria uma fora de escala no dia seguinte, dia correspondente à folga, invalidando a seqüência fixada para os descansos.

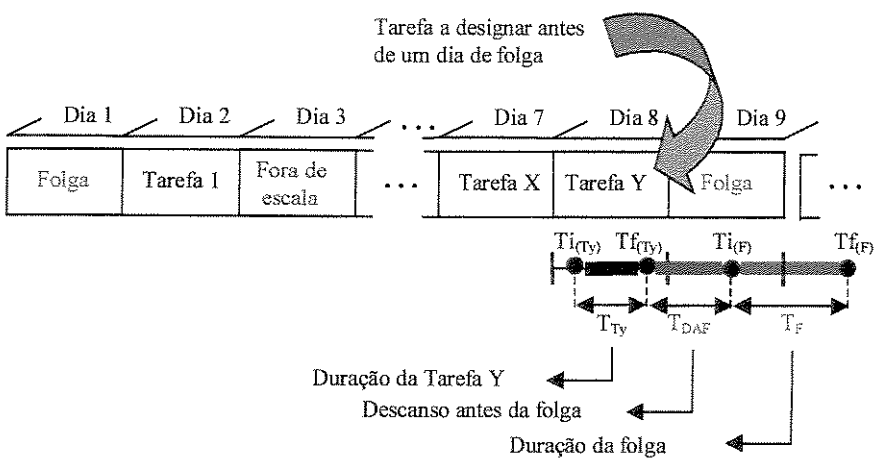


Figura 4.11. Alocação de uma tarefa antes de um dia de descanso.

5.- Identificação e alocação de tarefas após um dia de folga.

Quando efetua-se uma análise correspondente a um dia após o dia de folga, verifica-se a tarefa alocada no dia anterior à folga. Neste caso podem ocorrer duas situações. Uma primeira situação pode envolver uma fora de escala antes da folga, significando que a tarefa designada no dia anterior à fora de escala foi uma tarefa noturna extensiva. Uma segunda situação ocorre quando, anterior ao dia de folga, é designada qualquer outro tipo de tarefa, diurna, vespertina ou noturna.

Na Figura 4.12, podemos observar a primeira situação. Neste caso, as análises referem-se ao horário final da tarefa que determina esta fora de escala (Tarefa X). A partir do horário final da tarefa noturna extensiva $Tf_{(Tx)}$, adiciona-se um determinado tempo de descanso T_{DAF}. Este tempo é definido pela empresa, mas em geral, se faz corresponder com o tempo de descanso estabelecido entre tarefas consecutivas. A soma dos tempos T_{DAF} e $Tf_{(Tx)}$ determinam o horário inicial do dia de descanso $Ti_{(F)}$. Este horário inicial, por sua vez, determina o horário final do descanso $Tf_{(F)}$. A folga é legislada como sendo, no mínimo, de 24 horas. Por último, o horário final da folga determina o horário inicial da tarefa $Ti_{(Ty)}$ que poderá ser alocada um dia após uma folga, tendo em conta o critério adicional que estipula a alocação de tarefas com horário inicial superior às 5:00 h.

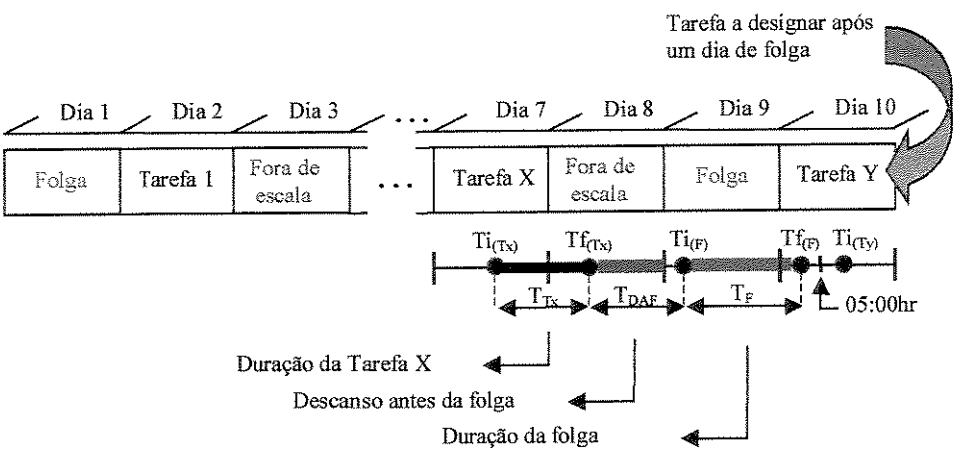


Figura 4.12. Alocação de uma tarefa após um dia de folga.

Quando no dia anterior à folga for designada uma tarefa diferente a uma tarefa noturna extensiva, é feita uma análise similar ao caso anterior. Neste caso, quando a tarefa antes da folga é noturna, o início da folga será em horas mais avançadas do dia. No entanto, isto não significa desconforto para o trabalhador ou equipe, pois antes de iniciar a folga o trabalhador recebe um tempo de descanso T_{DAF}.

decidido realizá-la como um processo adicional nas análises efetuadas para a elaboração de cada seqüencial de tarefas.

Assim, cada seqüencial obtido torna-se um indivíduo da população inicial da primeira fase do algoritmo desenvolvido. Esta primeira população será evoluída durante algumas gerações, até encontrar aquela programação que proporcione os melhores resultados, segundo os critérios de avaliação (seção 4.5.5).

4.5.3.- Geração da programação de escalas

Uma vez finalizada a primeira etapa evolutiva, procede-se à segunda fase, onde os novos cromossomos são construídos a partir de cada seqüencial de tarefas obtido. Desta vez, cada cromossomo será uma programação de escalas, objetivo básico deste trabalho.

Para construir cada programação de trabalho foram implementados dois métodos :

1.- Método de construção da programação de escalas sem análise da programação ou programações anteriores.

2.- Método de construção da programação de escalas com análise da programação ou programações anteriores.

Para o estudo do algoritmo desenvolvido, vários testes foram realizados usando o primeiro método, ajustando parâmetros e incorporando critérios adicionais que ajudassem a produzir um bom desempenho de modo geral.

1.- Método de construção da programação de escalas sem análise da programação ou programações anteriores.

A geração de uma programação de escalas sem considerar as análises correspondentes a uma programação ou programações anteriores é o caso mais simples. Neste caso, como foi discutido no Capítulo 3, as seqüências de tarefas correspondentes a cada trabalhador são selecionadas a partir dos seqüenciais gerados na primeira fase do algoritmo evolutivo. Para isto, somente é necessária a seleção aleatória de posições não repetidas dentro do

seqüencial de tarefas. Logo, a partir de cada posição selecionada, são extraídas as subseqüências de tarefas com um tamanho correspondente ao número de dias a programar (horizonte de planejamento). Assim, o planejamento é conformado pelas M subseqüências geradas. Na Figura 3.7, foi mostrado o resultado deste procedimento.

2.- Método de construção da programação de escalas com análise da programação ou programações anteriores.

A primeira suposição para este método é que já existe uma programação ou programações anteriores. Na prática, podemos considerar o trabalho efetivamente realizado pelas equipes ao invés das programações anteriores.

No término de cada programação de escalas, são armazenados os principais resultados que poderão ser utilizados na geração de programações futuras.

Entre os principais dados de um programa anterior constam as tarefas programadas no último dia com seus respectivos horários. Também é armazenado o último dia do período planejado em que cada trabalhador ou equipe recebeu um dia de descanso, e um vetor de estados que informa se o trabalhador ou equipe receberam um dia de descanso em um domingo nos últimos 14 dias planejados. Isto permite flexibilizar a restrição de um descanso dominical em períodos de 45 dias. Logo, serão fornecidos mais detalhes sobre estas condições.

Esta nova metodologia, com análise da programação anterior, engloba vários fatores que elevam a complexidade dos procedimentos de cálculo. Um deles é a variação do número de tarefas ou variação dos horários das tarefas a planejar. Outro fator está centrado na possibilidade de mudanças do número de trabalhadores.

Quando as tarefas, de uma programação para outra, são as mesmas, se facilita o processo de análise. Neste caso, tem-se a possibilidade de utilizar o seqüencial de tarefas da programação anterior, avaliado como a melhor solução encontrada ao longo das evoluções anteriores. Com isto, evita-se um excessivo esforço computacional, pois o primeiro processo evolutivo do algoritmo torna-se desnecessário.

Para criar os cromossomos nesta fase, deve-se levar em conta as tarefas programadas no último dia do programa anterior, o qual determina a tarefa a alocar no primeiro dia do

planejamento atual. Para isto, é referenciado o vetor que contém as últimas tarefas programadas a cada trabalhador ou equipe, assim como seus respectivos horários. Outra referência é feita ao último dia de descanso de cada trabalhador ou equipe, pois deve-se manter a seqüência de tarefas estabelecida entre dias de descanso para garantir a alocação de um deles em um domingo, durante 45 dias analisados.

Estas informações são importantes para iniciar uma nova programação, compatibilizando-a com a anterior e mantendo a factibilidade de cada planejamento elaborado.

O processo de compatibilização, entre a programação atual e a anterior é realizado passo a passo, tentando alocar cada tarefa do seqüencial atual, mas analisando o tempo de descanso estabelecido entre tarefas consecutivas, e os dias correspondentes às folgas, Figura 4.13.

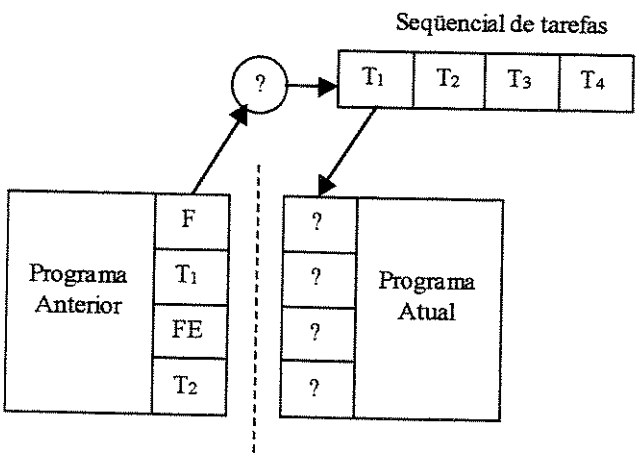


Figura 4.13. Análise do programa anterior para gerar o novo programa.

Nestas situações existem alguns elementos conflitantes. Nem sempre é possível que o conjunto de tarefas do seqüencial anterior seja compatível com o atual pois, além de cumprir o tempo de descanso entre tarefas, o primeiro dia de folga deverá estar em correspondência com a seqüência das folgas. Quando isto ocorre, o seqüencial gerado deixa de ser compatível com a programação anterior.

Por estas razões, o processo de elaboração dos programas quando são analisados programas anteriores, torna-se complexo, pois, além de atender às restrições típicas, deve-se garantir a compatibilidade entre uma programação e outra.

Uma vez determinados todos os seqüenciais que garantam as restrições e condições de factibilidade, são criados os programas mensais e iniciados os processos da segunda fase de evolução.

4.5.4.- Mecanismos e operadores de recombinação

A respeito da reprodução dos indivíduos, dois operadores foram implementados para realizar transformações nos seqüenciais de tarefas e nas programações mensais de escalas.

Durante a primeira etapa de evolução, foi usado um operador de mutação simples e na segunda fase utilizado um operador de mutação múltipla, equivalente a um *crossover* uniforme no nível de cromossomos.

4.5.4.1.- Mutação simples

O funcionamento deste operador é aparentemente simples, mas necessita de um pré-processamento de informação para efetuar as mudanças, o qual complica este processo.

Primeiramente, são selecionadas aleatoriamente duas tarefas dentro do seqüencial. A seguir, efetua-se um análise das restrições para garantir a factibilidade da troca dessas tarefas.

As análises realizadas no pré-processamento tem a ver com o descanso estabelecido entre tarefas consecutivas, sendo que a troca de duas tarefas implica em uma perturbação no equilíbrio dos horários anteriormente distribuídos no seqüencial. Para isto, devem ser analisados os tempos de descanso entre as duas tarefas que serão trocadas e as respectivas tarefas adjacentes a cada uma delas. Por exemplo, segundo a Figura 4.14, foram selecionadas as tarefas T_2 e T_4 para uma possível mutação. Se os tempos de descanso entre as tarefas T_1-T_4, T_4-T_3, T_3-T_2 e T_2-T_5 são iguais ou superiores ao tempo estabelecido para o descanso entre tarefas consecutivas, então as tarefas T_2 e T_4 poderão ser trocadas.

Outro aspecto é relacionado com as tarefas selecionadas para efetuar a troca de posição. Nesta instância, nem todas as tarefas podem ser trocadas pois existem alguns

fatores que invalidam este procedimento. Alguns critérios práticos de mutação não factível são relacionados a seguir :

- O dia de folga não permite efeitos de mutação com outra tarefa, ou deslocamento de posição, pois a restrição dos dias de folga impõe uma seqüência fixa dentro de seqüencial de tarefas.
- Tarefas noturnas extensivas não podem ser trocadas com outro tipo de tarefa por ocupar duas posições dentro do seqüencial de tarefas.
- A troca de uma tarefa T1, alocada depois de um dia de folga, por uma outra tarefa T2, só poderá ser efetuada se T2 apresenta um horário inicial igual ou maior do que a tarefa T1, caso contrário pode-se invalidar o descanso estabelecido entre a tarefa anterior à folga, somado além das 24 horas pertencentes ao dia de folga.

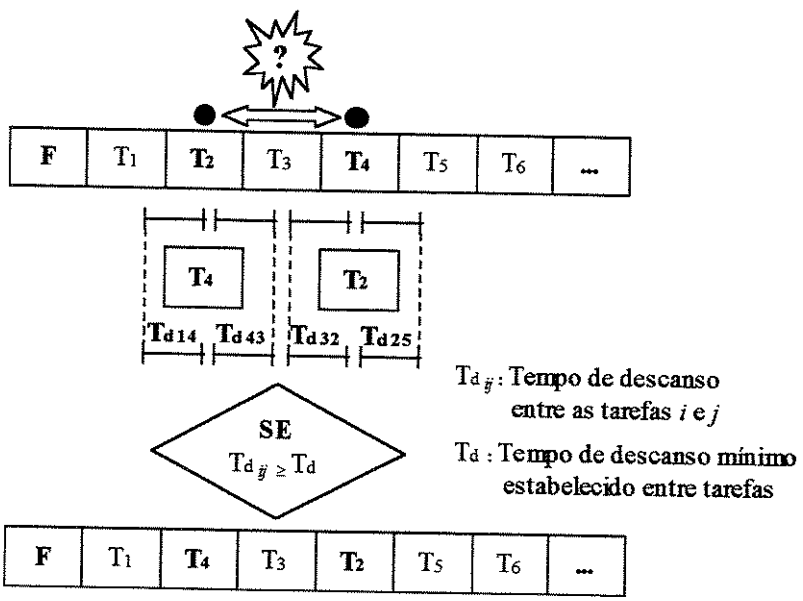


Figura 4.14. Análise dos tempos de descanso antes da troca de tarefas.

Como percebemos, a mutação simples entre tarefas combina etapas aleatórias (seleção das posições de troca) com etapas determinísticas (pré-análise das possíveis tarefas a trocar).

Uma vez conferida a validade do resultado da troca, efetua-se a operação, gerando um novo candidato à solução do problema.

4.5.4.2.- Mutação múltipla

A mutação múltipla somente opera sobre os cromossomos da segunda fase evolutiva. Este cromossomos, representados como matrizes de duas dimensões, contém M linhas (seqüências de tarefas de cada trabalhador) e D colunas (dias do horizonte de planejamento analisado).

A Figura 4.15 mostra um exemplo genérico deste tipo de transformação. Como observa-se, são geradas duas posições em forma aleatória, uma posição por linha e a outra por coluna (itens 1 e 2 da Figura 4.15). Desta forma são selecionadas uma determinada seqüência de tarefas e um determinado dia, dentro do horizonte de planejamento. Outra busca deve-se a um critério adotado para efetuar o cruzamento das seqüências a partir do dia de folga. Para isto, encontra-se o dia de folga mais próximo da posição selecionada (item 3). A seguir, determina-se a posição exata para efetuar o cruzamento (item 4).

Por outro lado, a partir da posição selecionada dentro das seqüências de tarefas são selecionadas aquelas seqüências que apresentam dias de folga no mesmo dia que a seqüência selecionada aleatoriamente (ponto 5). Com o conjunto formado, realiza-se uma análise das restrições (similar ao caso da mutação simples), entre as tarefas anteriores ao dia de folga de cada seqüência. Esta análise é feita fixando a primeira seqüência e selecionando, aleatoriamente, uma seqüência dentro do conjunto de seqüências formado. Logo depois de efetuar as correspondentes análises das restrições, é realizado o cruzamento das subseqüências correspondentes, a partir do ponto de cruzamento definido.

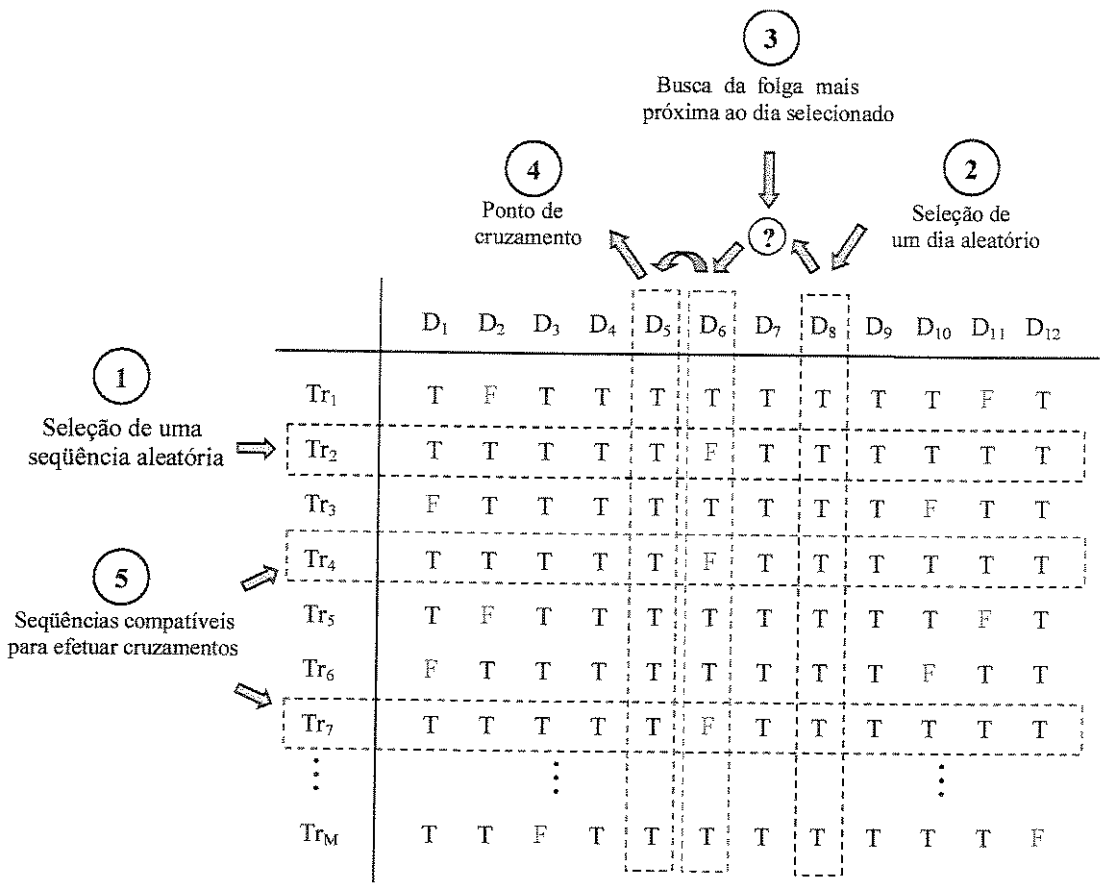
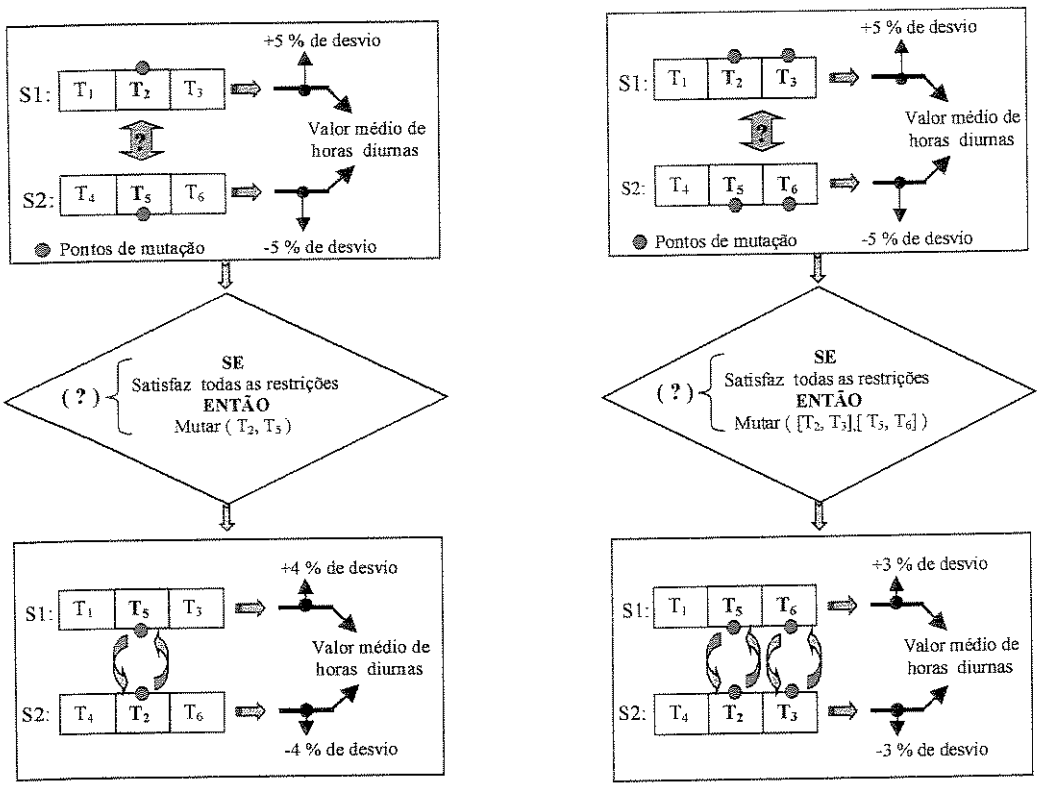


Figura 4.15. Passos para efetuar uma mutação múltipla de tarefas.

Na Figura 4.16 exemplifica-se o significado destas transformações baseado na seguinte discussão: sejam S1={T1, T2, T3} e S2={T4, T5, T6}, duas seqüências de trabalho, programadas para dois trabalhadores diferentes. Suponha que a soma das horas diurnas, programadas na seqüência S1, apresenta um desvio de +5 %, referente ao valor médio de horas diurnas, calculado a partir da programação total do mês. Por outro lado, suponha que a soma das horas diurnas programadas na seqüências S2 apresenta um desvio de –5 %, referente ao mesmo valor médio de horas diurnas. Se são cumpridas as condições para intercambiar as tarefas T2 e T3, da seqüência S1, pelas tarefas T5 e T6, da seqüência S2, poderíamos obter como resultado um +3 % e -3 % de desvio para as seqüências S1 e S2 respectivamente (Figura 4.16 (b)).



(a) (b)

Figura 4.16. Efeitos na mutação de tarefas. (a) Mutação simples, (b) Mutação múltipla.

Os resultados gerados a partir de ambas as operações (mutação simples e múltipla) têm como objetivo comum a minimização do desvio (erro) para cada um dos parâmetros analisados.

4.5.5.- Função e critérios de avaliação

Os critérios iniciais de avaliação foram similares para as duas fases evolutivas. Durante a primeira fase, os indivíduos (seqüenciais de tarefas) são avaliados a partir da construção dos programas mensais (programação das escalas), segundo os procedimentos já discutidos. A construção destes programas somente é realizada para o processo de avaliação. Os cromossomos das novas populações continuam sendo seqüenciais de tarefas.

Na segunda etapa, uma vez evoluída a população dos seqüenciais, são construídos os programas mensais, definidos como cromossomos da segunda fase evolutiva. Este procedimento, diferentemente do anterior, não é reversível. A partir do programa mensal modificado não é possível retornar ao seqüencial que originou o dito programa devido às características do método de construção dos programas mensais, a partir dos seqüenciais de tarefas. Assim, ao longo das evoluções, só serão tratados os programas mensais das escalas como representação direta das soluções do problema (cromossomos).

O processo de avaliação das soluções, tanto na primeira como na segunda fase evolutiva, depende da quantidade de tarefas e trabalhadores envolvidos na programação. Quando a programação requer um número elevado de tarefas e trabalhadores, o procedimento se torna complexo e induz um grande esforço computacional.

Segundo os parâmetros de otimização definidos como critérios de avaliação das soluções, foram estudados e implementados dois métodos fundamentais. O primeiro método é baseado em soma ponderada dos parâmetros. O segundo método é baseado em Lógica *Fuzzy* (mais precisamente, na teoria de Conjuntos *Fuzzy*).

Ambas as metodologias usam uma base comum nos procedimentos iniciais. Quando termina o processo de geração de uma programação mensal, são calculados os valores médios de cada um dos quatro parâmetros de otimização (horas diurnas, horas noturnas, horas de descanso e pernoites programados). Este valor médio é calculado segundo as equações (4.1) e (4.2),

$$Valor_{(i)}^{(p)} = \sum_{j=1}^{D} p_j^{(i)} \tag{4.1}$$

onde:

D : Horizonte de planejamento (número de dias).

$Valor_{(i)}^{(p)}$: Valor do parâmetro p, programado a um trabalhador i, durante todo o horizonte de planejamento.
$p \in$ {horas diurnas, horas noturnas, horas de descanso e número de pernoites }

$$ValorMédio^{(p)} = \frac{\sum_{i=1}^{M} Valor_i^{(p)}}{M} \tag{4.2}$$

onde : M é o número total de trabalhadores ou equipes de trabalho.

Os valores dos desvios (erros) entre o valor de cada parâmetro *p*, obtidos pela equação (4.1), e seus respectivos valores médios, obtidos pela equação (4.2), são calculados segundo a equação (4.3),

$$Desvio_{(i)}^{(p)} = \frac{\left| Valor_{(i)}^{(p)} - ValorMédio^{(p)} \right|}{ValorMédio^{(p)}} \qquad (4.3)$$

onde :

$Desvio_{(i)}^{(p)}$: Desvio do parâmetro *p*, para o trabalhador *i*, durante todo o horizonte de planejamento.

Desta forma, cada valor de desvio de cada parâmetro *p* constitui um elemento de otimização para o nosso problema. A tarefa de otimização é obter programações mensais de escalas que minimizem cada um deles. No entanto, estes objetivos são conflitantes e obter um compromiso aceitável não é uma tarefa simples.

Como mencionamos, as duas metodologias de cálculo, implementadas para avaliar as soluções (indivíduos), usam estas idéias.

4.5.5.1.- Avaliação das soluções usando um método de soma ponderada

Soma ponderada é um procedimento muito usado em problemas de otimização multiobjetivos [17, 25]. O fato de estabelecer coeficientes de ponderação nos permite influir na qualidade das soluções. Os coeficientes de ponderação determinam o peso ou grau de importância de cada parâmetro de otimização. Desta forma, a critério do especialista encarregado de elaborar as escalas, é permitido variar a qualidade da solução, tendo em conta a importância de um determinado parâmetro nas escalas produzidas.

A concepção deste método foi estabelecida segundo as seguintes condições :

Seja P o conjunto de parâmetros $P \in$ {HDi, HNo, HDe, Pe}, sendo HDi as horas diurnas, HNo as horas noturnas, HDe as horas de descanso e Pe os pernoites, também referidos como o número de repetições de tarefas noturnas.

Em cada programação mensal, todo trabalhador recebe uma determinada seqüência de tarefas a qual, implicitamente, apresenta estes parâmetros :

$$\mathrm{Tr}_i \Leftarrow \begin{cases} \mathrm{HDi}_i \\ \mathrm{HNo}_i \\ \mathrm{HDe}_i \\ \mathrm{Pe}_i \end{cases} \quad \text{com } i = 1,\ldots, M \text{; onde } M \text{ é o número total de trabalhadores.}$$

Como foi mencionado, o valor destes quatro parâmetros e seus respectivos valores médios, é calculado conforme as equações (4.1) e (4.2). Por tanto, a partir da equação (4.3), determinam-se os desvios (objetivos de otimização).

Para melhor compreensão dos procedimentos desenvolvidos na aplicação, mostramos a seguinte seqüência de cálculos :

Passo 1. Cálculo dos valores de cada parâmetro a partir das seqüências programadas para cada trabalhador, equações (4.4) a (4.7).

$$\mathrm{HDi}_i = \sum_{j=1}^{D} \underbrace{\mathrm{Tarefa}_j^{(i)}}_{[\mathrm{HDi}]} \quad \text{, Valor das horas diurnas programadas ao trabalhador } i. \tag{4.4}$$

$$\mathrm{HNo}_i = \sum_{j=1}^{D} \underbrace{\mathrm{Tarefa}_j^{(i)}}_{[\mathrm{HNo}]} \quad \text{, Valor das horas noturnas programadas ao trabalhador } i. \tag{4.5}$$

$$\mathrm{HDe}_i = \sum_{j=1}^{D} \underbrace{\mathrm{Tarefa}_j^{(i)}}_{[\mathrm{HDe}]} \quad \text{, Valor das horas de descanso programadas ao trabalhador } i. \tag{4.6}$$

$$\mathrm{Pe}_i = \sum_{j=1}^{D} \underbrace{\mathrm{Tarefa}_j^{(i)}}_{[\mathrm{Pe}]}. \quad \text{, Número de pernoites programados ao trabalhador } i. \tag{4.7}$$

onde :

$\underbrace{\mathrm{Tarefa}_j^{(i)}}_{[\mathrm{HDi}]}$ Valor correspondente às horas diurnas relativas à tarefa programada no dia j, para o trabalhador i.

D : Horizonte de planejamento (número de dias a programar).

Para cálculos posteriores, os valores são normalizados, estabelecendo iguais faixas de variação, segundo cada contexto.

Passo 2. Cálculo dos valores médios de cada parâmetro a partir da programação total de escalas (programa mensal).

$$\overline{\text{HDi}} = \frac{\sum_{i=1}^{M} \text{HDi}_i}{M} \quad \text{, Valor médio das horas diurnas programadas.} \tag{4.8}$$

$$\overline{\text{HNo}} = \frac{\sum_{i=1}^{M} \text{HNo}_i}{M} \quad \text{, Valor médio das horas noturnas programadas.} \tag{4.9}$$

$$\overline{\text{HDe}} = \frac{\sum_{i=1}^{M} \text{HDe}_i}{M} \quad \text{, Valor médio das horas de descanso programadas.} \tag{4.10}$$

$$\overline{\text{Pe}} = \frac{\sum_{i=1}^{M} \text{Pe}_i}{M} \quad \text{, Valor médio do número de pernoites programados.} \tag{4.11}$$

onde M representa o número total de trabalhadores ou equipes de trabalho.

Passo 3. Cálculo dos valores de desvios de cada parâmetro somando os desvios individuais de cada trabalhador ou equipe de trabalho, mediante as seguintes equações :

$$\text{Desvio\_HDi} = \sum_{i=1}^{M} \frac{\left|\text{HDi}_i - \overline{\text{HDi}}\right|}{\overline{\text{HDi}}} \quad \text{, Desvio das horas diurnas programadas.} \tag{4.12}$$

$$\text{Desvio\_HNo} = \sum_{i=1}^{M} \frac{\left|\text{HNo}_i - \overline{\text{HNo}}\right|}{\overline{\text{HNo}}} \quad \text{, Desvio das horas noturnas programadas.} \tag{4.13}$$

$$\text{Desvio\_HDe} = \sum_{i=1}^{M} \frac{\left|\text{HDe}_i - \overline{\text{HDe}}\right|}{\overline{\text{HDe}}} \quad \text{, Desvio das horas de descanso programadas.} \tag{4.14}$$

$$\text{Desvio\_Pe} = \sum_{i=1}^{M} \frac{\left|\text{HPe}_i - \overline{\text{HPe}}\right|}{\overline{\text{HPe}}} \quad \text{, Desvio do número de pernoites programados.} \tag{4.15}$$

Passo 4. Cálculo do valor de *fitness* (grau de avaliação da solução ou medida de adaptabilidade) da programação mensal total, através da equação (4.16),

$$\text{Fitness} = \sum_{i=1}^{No} \mathrm{K}_i * \text{Desvio}_i \text{ , Grau de avaliação da solução.} \tag{4.16}$$

onde :

K_i, Coeficiente de ponderação do objetivo i, onde $K_i \in [0,1]$ e $\sum_{i=1}^{No} K_i = 1$.

No, Número de objetivos (ou parâmetros) de otimização,

4.5.5.2.- Avaliação das soluções usando conjuntos fuzzy

Em ambiente *fuzzy*, um aspecto importante nas análises para tomada de decisão é a simetria que podemos assumir entre os objetivos e as restrições de um dado problema. Deste ponto de vista, os conceitos são definidos como conjuntos *fuzzy* no espaço de alternativas, e as funções de pertinência são usadas como funções de desempenho. Assim, um tratamento similar para ambos (objetivos e restrições) pode ser realizado na formulação da decisão.

Neste contexto, sugere-se que a decisão *fuzzy* seja induzida a partir da interseção entre os objetivos e restrições *fuzzy*. Bellman e Zadeh [6] formalizaram estas idéias como segue.

Sejam O e R, um objetivo *fuzzy* e uma restrição *fuzzy*, respectivamente, em um espaço de alternativas X. Assim, O e R são combinados, induzindo uma decisão D, a qual resulta da interseção de O e R, segundo a equação (4.17).

$$D = O \cap R \tag{4.17}$$

Conseqüentemente,

$$\mu_D(x) = \mu_O(x) \wedge \mu_R(x) \tag{4.18}$$

onde $\mu_{(i)}$, denota funções de pertinência e $\wedge$ denota a t-norma **min** [27], um operador que modela a interseção.

De forma similar, para No objetivos $O_1, \ldots, O_{No}$ e Nr restrições $R_1, \ldots, R_{Nr}$, obtemos :

$$D = O_1 \cap \ldots \cap O_{No} \cap R_1 \cap \ldots \cap R_{Nr} \tag{4.19}$$

Sendo assim,

$$\mu_D{}^{(x)} = \left[\overset{No}{\underset{i=1}{\wedge}} \mu O_i{}^{(x)} \right] \wedge \left[\overset{Nr}{\underset{j=1}{\wedge}} \mu R_j{}^{(x)} \right] \tag{4.20}$$

Como podemos observar, objetivos e restrições têm pesos similares neste processo de tomada de decisão. A interseção entre eles pode ser vista como a confluência de objetivos e restrições [6]. Neste sentido, as funções de pertinência são usadas para obter uma medida do grau de superposição entre condições. Estes graus serão usados como referências em um critério analítico para tomar decisões.

Baseado nestas idéias, a decisão, no contexto *fuzzy* proposto neste trabalho, pode ser formulada como em (4.21) :

$$\mu_D = \left[\overset{No}{\underset{i=1}{\mathrm{A}}} \mu_i(Objetivos) \right] \mathrm{A} \left[\overset{Nr}{\underset{j=1}{\mathrm{A}}} \mu_j(Restrições) \right] \tag{4.21}$$

onde :

$$Objetivos = \sum_{i=1}^{M} Desvio_i$$

$$Restrições = \frac{\sum_{i=1}^{M} Restrição_i}{M}$$

A, operador de agregação, considerado como mínimo.

M, é o número total de trabalhadores.

No, é o número de objetivos de interesse.

Nr, é o número de restrições relevantes. Na nossa aplicação, foi só considerada a restrição referente ao tempo de descanso entre tarefas consecutivas.

Esta decisão *fuzzy* é usada como uma medida de adaptabilidade dos indivíduos. Desta forma, a melhor programação mensal (indivíduo da população) será obtida quando for atingido o valor máximo de μ_D, ao longo de todas as gerações. Esta idéia é traduzida em (4.22).

$$S^* = \arg\max_{S \in \mathbf{S}} \mu_D(S) \tag{4.22}$$

onde :

S^*, é a melhor programação mensal gerada (indivíduo mais adaptado)

S, é o conjunto de todas as programações mensais factíveis para o problema.

Os conjuntos *fuzzy* são definidos segundo o escopo de abrangência das variáveis que eles caraterizam ou representam, ver Figura 5.8.

4.5.6.- Métodos de seleção

Durante todas as etapas de simulação da aplicação desenvolvida, foi usado o método de seleção elitista, selecionando sempre os melhores indivíduos para compor as novas populações.

A respeito da seleção dos cromossomos pais, as metodologias de seleção foram de natureza aleatória, mas com as devidas etapas de pré-processamento das restrições para garantir resultados factíveis.

4.6.- Resumo

Baseado em trabalhos anteriores e nas experiências adquiridas na prática, foi possível o estudo e desenvolvimento de uma nova metodologia de solução do problema de programação de escalas usando técnicas de inteligência computacional. Os principais procedimentos elaborados foram discutidos, enfatizando aqueles de maior complexidade e importância quanto ao modelo apresentado.

A respeito do algoritmo evolutivo desenvolvido, foram explicitados os elementos fundamentais que o constituem. Neste sentido, mostrou-se as duas fases de evolução, desenvolvidas a partir das codificações dos cromossomos.

O próprio processo de elaboração de cada programa mensal determinou as duas formas de representação das soluções, onde transformações mais simples podem ser efetuadas em nível de seqüenciais de tarefas e transformações de maior dimensão podem ser realizadas em nível da programação completa. Neste último caso, foi possível estabelecer troca de subseqüências de tarefas entre dois trabalhadores, tentando melhorar o balanço de carga de trabalho e descanso programado para cada trabalhador.

As análises de uma programação ou programações anteriores foi outro ponto fundamental deste capítulo. Este item engloba procedimentos não levados em consideração naquelas empresas onde a programação é elaborada manualmente. A principal razão está na elevada complexidade das análises requeridas, dado o grande número de combinações possíveis para se chegar a soluções factíveis deste problema.

Os critérios de avaliação das soluções obtidas foram baseadas em considerações práticas utilizadas em algumas empresas. Neste sentido, segundo os objetivos principais, foi definido inicialmente um método de soma ponderada, possibilitando a avaliação de cada programa segundo o grau de importância relativa dos objetivos de otimização. Outra forma de avaliar as soluções foi inspirada nas idéias da teoria de Conjuntos *Fuzzy*. Desta vez, a qualidade foi determinada usando a confluência (interseção) entre os diferentes objetivos e uma das restrições do problema, aquela referente ao tempo de descanso entre duas tarefas consecutivas. Em menor ênfase, mostraram-se os procedimentos de transformações locais efetuadas em cada fase da evolução. Os operadores de mutação simples e múltipla ajudam a re-alocar tarefas ou seqüências de tarefas nos seqüenciais e programações, respectivamente, fornecendo programas de escalas com horários de trabalho mais balançeados, considerando o período de planejamento.

Capítulo 5

Implementação do Algoritmo Evolutivo para Geração de Escalas.

5.1.- Introdução

Nas seções seguintes, apresenta-se a estrutura básica da ferramenta computacional desenvolvida para a programação de escalas em empresas ferroviárias.

Primeiramente, são fornecidos alguns detalhes sobre as principais classes implementadas e os principais elementos que as compõem. Explicam-se também os modos de funcionamento das interfaces de entrada/saída para garantir um uso adequado do sistema como um todo.

Para testar a viabilidade do algoritmo, os resultados experimentais são comparados com os resultados elaborados na prática, analisando os critérios de avaliação discutidos nos capítulos anteriores.

5.2.- Estrutura da ferramenta computacional.

O algoritmo foi desenvolvido usando as ferramentas da linguagem de programação JAVA, aplicando os paradigmas da programação orientada a objetos. Esta primeira

versão foi nomeada "*Gerador de Escalas por Algoritmos Evolutivos*" (*GESAE*). A arquitetura básica do algoritmo apresenta-se na figura a seguir.

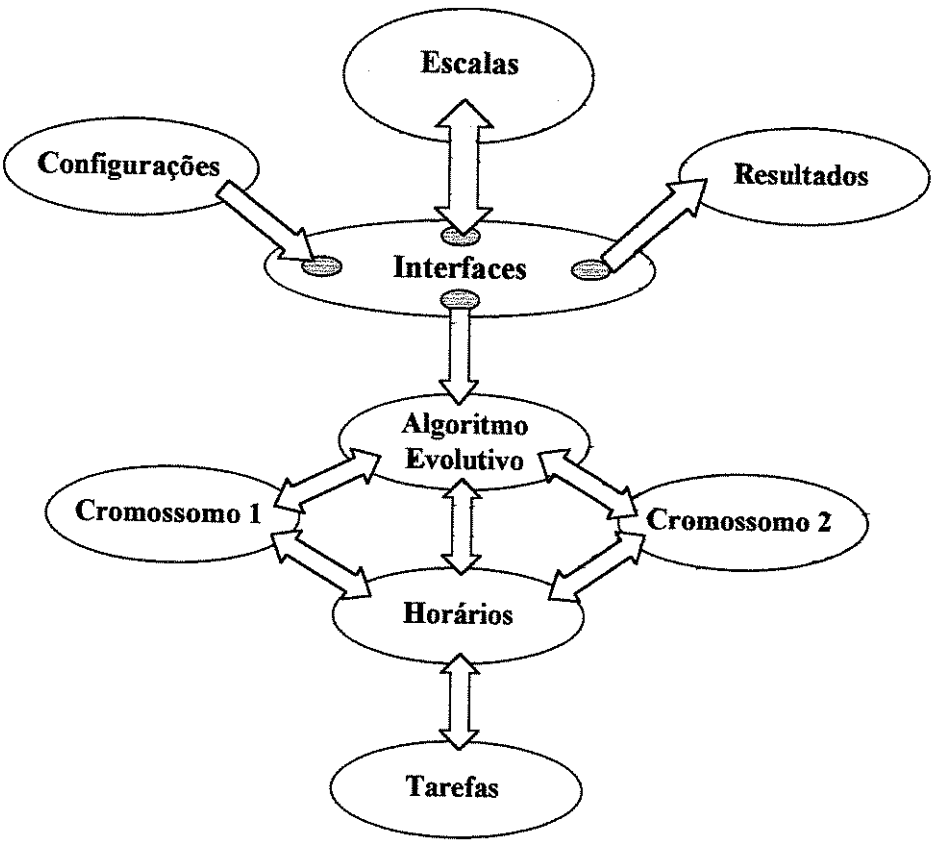


Figura 5.1. Estrutura operacional do sistema GESAE.

Como observa-se, o módulo **Escalas** representa a classe que gerencia o trabalho do sistema através das diferentes interfaces de entrada/saída interativas com o usuário.

O módulo **Interfaces** engloba todas as classes que são usadas durante as diferentes etapas de funcionamento do algoritmo. As interfaces de entrada incluem-se no módulo **Configurações** e as interfaces de saída estão incluídas no módulo **Resultados**.

Os módulos **Algoritmo Evolutivo**, **Cromossomo 1**, **Cromossomo 2**, **Horários** e **Tarefas** representam as principais classes do algoritmo evolutivo, implementadas para os processos de busca das soluções do problema.

O módulo **Configurações** envolve três classes fundamentais relacionadas com o dimensionamento de tarefas, a configuração dos parâmetros do algoritmo evolutivo e a configuração da programação de escalas.

O módulo **Resultados** refere-se às duas variantes implementadas para mostrar os resultados finais das simulações. Neste caso, tem-se a possibilidade de obter os resultados por tabelas ou de forma gráfica (histogramas).

5.3.- Descrição das principais classes implementadas.

A classe *Algoritmo Evolutivo* constitui o motor central dos procedimentos de cálculo. Nesta classe, encontram-se os métodos principais do algoritmo de evolução, tais como : criação das populações iniciais, evolução, métodos de mutação simples e múltipla, seleção e avaliação dos indivíduos. Uma vez em funcionamento, são enviadas as mensagens às classes *Cromossomo 1*, *Cromossomo 2*, *Horários* e *Tarefas*, em forma interativa, através dos diferentes objetos declarados. Durante a primeira fase de evolução, a classe *Cromossomo 1* é solicitada para a geração dos seqüenciais de tarefas (indivíduos das populações da primeira fase evolutiva). Em forma similar, a classe *Cromossomo 2* é solicitada para a geração dos indivíduos da população composta por programações de escalas.

A classe *Horários* facilita os cálculos e o envio de informação entre a classe *Tarefas* e as diferentes classes que usam os dados das tarefas. Na classe *Horários*, estão implementados todos os métodos referentes às tarefas, como por exemplo determinar se uma tarefa é noturna ou noturna extensiva.

A classe *Tarefas* é invocada durante os processos de simulação para fornecer informação sobre as tarefas que estão sendo alocadas. Nesta classe, são criados os objetos do tipo Tarefas, sendo que cada objeto contém informação referente ao identificador, código, descrição, horário inicial e horário final de cada tarefa.

No Apêndice I, relacionam-se as classes principais. Devido a sua grande quantidade, somente apresentam-se os principais atributos declarados e métodos implementados em cada classe.

5.4.- Descrição das interfaces.

Uma vez em funcionamento, o sistema apresenta seu ambiente de trabalho com o menu principal e os submenus correspondentes, conforme a Tabela 5.1.

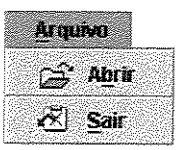
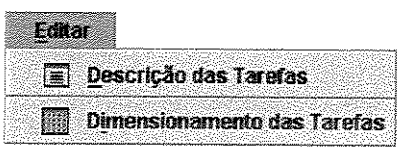
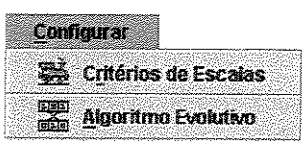
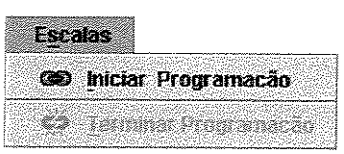
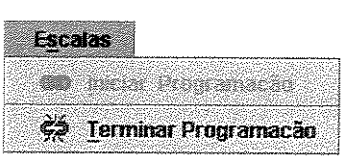
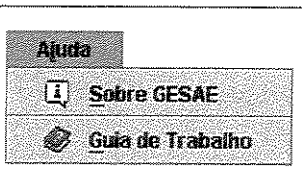
Menu	Submenus
Arquivo	
Editar	
Configurar	
Escalas	 
Resultados	
Ajuda	

Tabela 5.1. Menu principal e submenus correspondentes.

A cada início, somente estarão habilitados aqueles submenus que poderão ser selecionados em primeira instância pelo usuário. Isto forma parte de toda uma lógica seqüencial a seguir durante o uso da ferramenta.

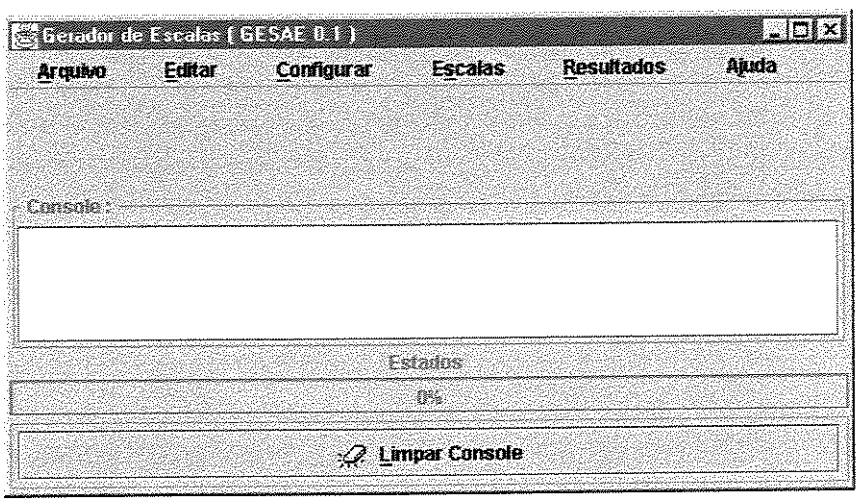


Figura 5.2. Ambiente inicial de trabalho.

Como observa-se na Figura 5.2, inicialmente são apresentadas a barra do menu principal e uma área de texto, onde são acompanhados os diferentes procedimentos do algoritmo para ir avaliando o seu desempenho em cada uma das etapas de evolução.

O menu "Arquivo" é instanciado com a finalidade de se ter acesso aos dados das tarefas a programar. Esta operação pode ser o primeiro passo a realizar, mas não é obrigatório dentro da lógica seqüencial das operações de trabalho, pois existirão outros submenus inicialmente habilitados. Outros submenus, pelo contrário, precisam dos dados contidos no arquivo antes mencionado, para efetuar sua função. Caso seja selecionado um arquivo errado, o sistema gera uma mensagem de erro ao tentar ler os dados das tarefas. Uma vez realizada a operação de leitura dos dados, ativam-se outros estados que garantam o acesso a novas opções do menu principal.

O menu "Editar" apresenta o submenu "Descrição das Tarefas" para mostrar a descrição de cada tarefa, baseado no arquivo lido no passo anterior. Esta opção constitui somente uma janela informativa. Em outra opção de edição tem-se o submenu

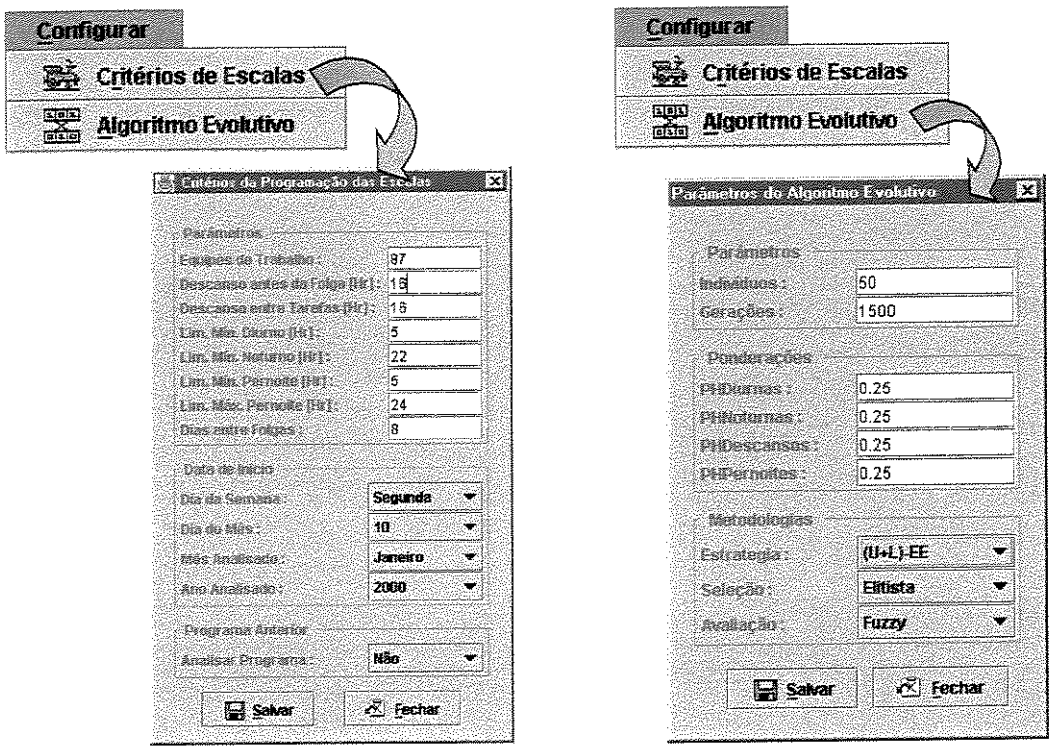
"Dimensionamento das Tarefas". Neste caso, são editadas as diferentes tarefas contidas no arquivo, assim como seus respectivos dados, Figura 5.3.

Outras opções podem se realizar através da janela de edição do dimensionamento. Neste sentido, é permitido ao usuário uma reestrutura das tarefas e seus dados, considerando a possibilidade de alterações tais como a incorporação de novas tarefas, eliminação de tarefas já existentes ou mudança nos horários inicialmente estabelecidos para as tarefas mostradas.

No	Código	Tarefa	Início [Hr]	Jornada [Hr]	Fim [Hr]
1	02	Manobra	06:30	08:00	14:30
2	02	Manobra	07:00	08:00	15:00
3	02	Manobra	07:01	08:00	15:01
4	02	Manobra	23:00	08:00	07:00
5	08	Manobra_1	15:00	08:00	23:00
6	08	Manobra_1	15:01	08:00	23:01
7	08	Manobra_1	18:30	08:00	02:30
8	08	Manobra_1	23:01	08:00	07:01
9	09	Prontidao	00:01	15:00	15:01
10	09	Prontidao	01:00	15:00	16:00

Figura 5.3. Menu "Editar", submenu "Dimensionamento das Tarefas".

Dentro do menu "Configurar", apresentam-se os submenus "Critérios de Escalas" e "Algoritmo Evolutivo". O primeiro deve ser ativado para a configuração dos parâmetros relacionados com a programação das escalas e o segundo permite a configuração dos parâmetros do algoritmo evolutivo.



(a) (b)

Figura 5.4. Menu "Configurar". (a) Submenu "Critérios de Escalas". (b) Submenu "Parâmetros do Algoritmo Evolutivo".

Como observa-se na Figura 5.4 (a), as configurações referentes à programação de escalas concernem às principais restrições e critérios utilizados em cada planejamento. Por outro lado, a Figura 5.4 (b) mostra os principais parâmetros do algoritmo evolutivo que podem ser configurados em cada corrida de simulação.

O menu "Escalas" apresenta os submenus "Iniciar Programação" e "Terminar Programação". A primeira opção começa a execução do algoritmo evolutivo e a segunda interrompe a dita execução caso o usuário determine desnecessária a continuação do processo de busca da solução.

O menu "Resultados" apresenta os submenus "Tabela" e "Gráficos". A seleção dos resultados por tabelas mostra uma janela como a mostrada na Figura 5.5.

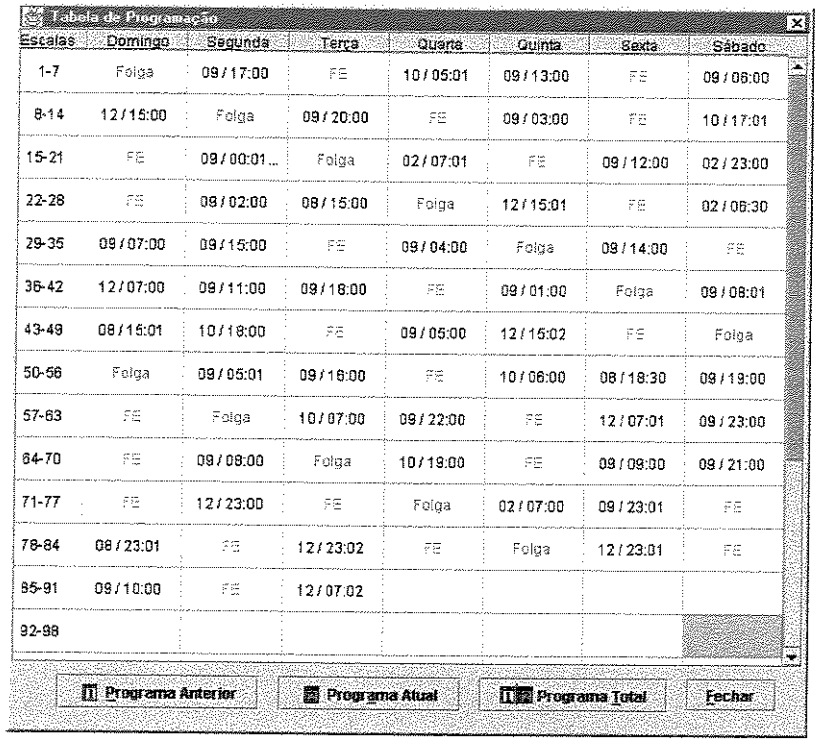
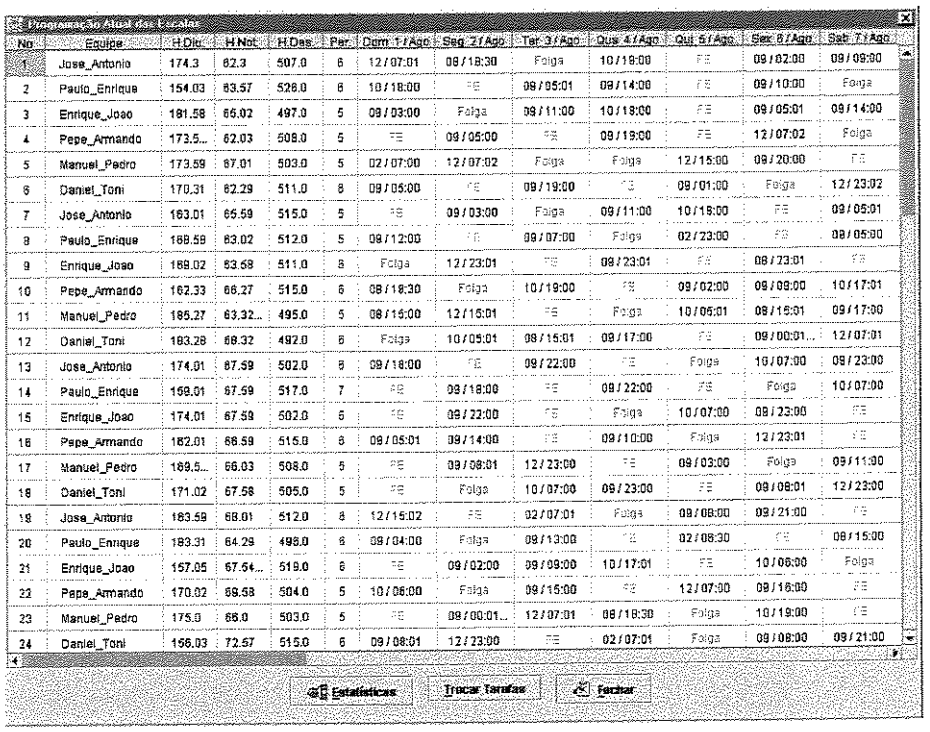


Figura 5.5. Exemplo de resultado por tabelas (seqüencial de tarefas).

Nesta janela, mostra-se o seqüencial de tarefas que obteve os melhores resultados segundo os critérios de avaliação considerados.

Caso a elaboração da programação de escalas tenha sido efetuada com análises de um programa anterior dispõe-se de três opções que mostrarão os resultados do programa anterior, do programa atual ou da programação total (programa anterior mais o programa atual). Por exemplo, a opção "Programa Atual" permite o acesso a uma outra tabela que mostra a programação obtida a partir do seqüencial mostrado e as diferentes transformações realizadas no algoritmo de evolução. Neste caso, relacionam-se as diferentes equipes de trabalho com as respectivas programações de tarefas, assim como os horários programados nas diferentes escalas, Figura 5.6.

Os resultados gráficos são mostrados em forma de distribuição de freqüências (histogramas) para avaliar os balanços dos horários programados. Na Figura 5.7, é mostrado um exemplo de um resultado gráfico. No exemplo, observa-se os desvios percentuais, calculados com respeito ao valor médio de cada parâmetro.



No	Equipe	H.Dia	H.Not	H.Des	Per	Dom 1/Ago	Seg 2/Ago	Ter 3/Ago	Qua 4/Ago	Qui 5/Ago	Sex 6/Ago	Sab 7/Ago
1	Jose_Antonio	174.3	62.3	507.0	6	12/07:01	08/18:30	Folga	10/19:00	FE	09/02:00	09/09:00
2	Paulo_Enrique	154.03	63.57	526.0	6	10/19:00	FE	09/05:01	09/14:00	FE	09/10:00	Folga
3	Enrique_Joao	181.58	66.02	497.0	5	09/03:00	Folga	09/11:00	10/19:00	FE	09/05:01	09/14:00
4	Pepe_Armando	173.5...	62.03	508.0	5	FE	09/05:00	FE	09/19:00	FE	12/07:02	Folga
5	Manuel_Pedro	173.59	67.01	503.0	5	02/07:00	12/07:02	Folga	Folga	12/15:00	09/20:00	FE
6	Daniel_Toni	170.31	62.29	511.0	6	09/05:00	FE	09/19:00	FE	09/01:00	Folga	12/23:02
7	Jose_Antonio	163.01	65.59	515.0	5	FE	09/03:00	Folga	09/11:00	10/19:00	FE	09/05:01
8	Paulo_Enrique	168.58	63.02	512.0	5	09/12:00	FE	09/07:00	Folga	02/23:00	FE	08/05:00
9	Enrique_Joao	168.02	63.59	511.0	6	Folga	12/23:01	FE	09/23:01	FE	08/23:01	FE
10	Pepe_Armando	162.33	66.27	515.0	6	08/18:30	Folga	10/19:00	FE	09/02:00	09/09:00	10/17:01
11	Manuel_Pedro	165.27	63.32...	495.0	5	08/15:00	12/15:01	FE	Folga	10/05:01	08/15:01	09/17:00
12	Daniel_Toni	163.28	68.32	492.0	6	Folga	10/05:01	08/15:01	09/17:00	FE	09/00:01...	12/07:01
13	Jose_Antonio	174.01	67.59	502.0	6	09/18:00	FE	09/22:00	FE	Folga	10/07:00	09/23:00
14	Paulo_Enrique	169.01	67.59	517.0	7	FE	09/18:00	FE	09/22:00	FE	Folga	10/07:00
15	Enrique_Joao	174.01	67.59	502.0	6	FE	09/22:00	FE	Folga	10/07:00	09/23:00	FE
16	Pepe_Armando	162.01	66.59	515.0	6	09/05:01	09/14:00	FE	09/10:00	Folga	12/23:01	FE
17	Manuel_Pedro	169.5...	66.03	508.0	5	FE	09/08:01	12/23:00	FE	09/03:00	Folga	09/11:00
18	Daniel_Toni	171.02	67.58	505.0	5	FE	Folga	10/07:00	09/23:00	FE	09/08:01	12/23:00
19	Jose_Antonio	163.59	66.01	512.0	6	12/15:02	FE	02/07:01	Folga	09/08:00	09/21:00	FE
20	Paulo_Enrique	163.31	64.29	498.0	6	09/04:00	Folga	09/13:00	FE	02/08:30	FE	08/15:00
21	Enrique_Joao	157.05	67.54...	519.0	6	FE	09/02:00	09/09:00	10/17:01	FE	10/06:00	Folga
22	Pepe_Armando	170.02	66.58	504.0	5	10/06:00	Folga	09/15:00	FE	12/07:00	09/16:00	FE
23	Manuel_Pedro	175.0	66.0	503.0	5	FE	09/00:01...	12/07:01	08/18:30	Folga	10/19:00	FE
24	Daniel_Toni	166.03	72.57	515.0	6	09/08:01	12/23:00	FE	02/07:01	Folga	09/08:00	09/21:00

Estatísticas | Trocar Tarefas | Fechar

Figura 5.6. Exemplo de resultado por tabelas (programação de escalas).

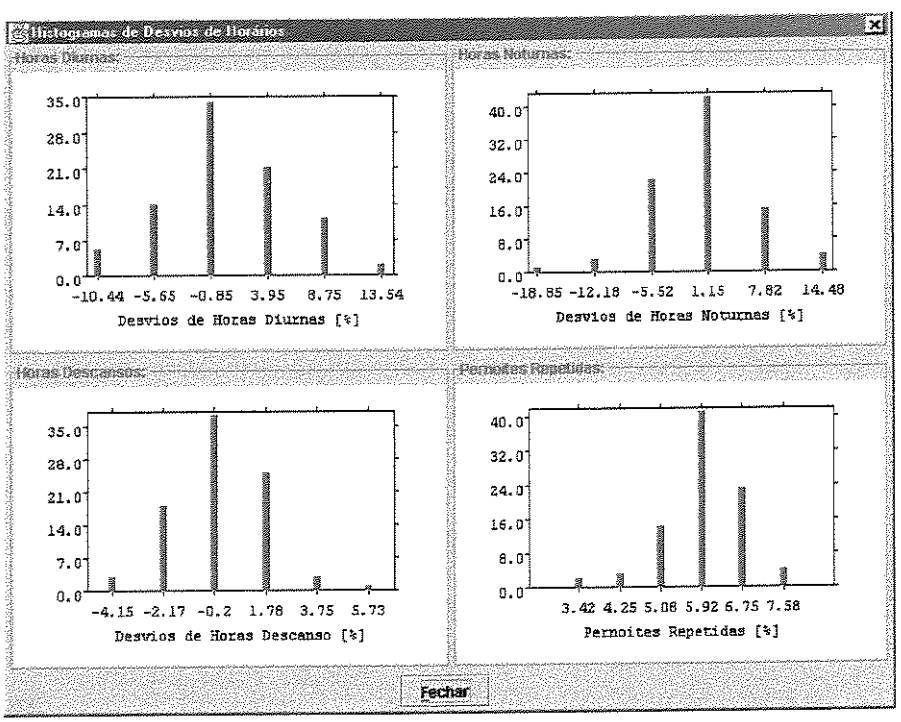


Figura 5.7. Exemplo de resultados gráficos (histogramas).

Por último, o menu "Ajuda" apresenta os submenus "Sobre GESAE" e "Guia de Trabalho", através dos quais obtém-se informação e ajuda sobre o funcionamento da ferramenta computacional.

5.5.- Exemplo de aplicação.

Com objetivos de comparação, usou-se como referência um resultado elaborado na prática por especialistas. Com igual conjunto de dados foi testado o algoritmo evolutivo usando as duas metodologias de avaliação: ponderada e *fuzzy*.

A Tabela 5.2 relaciona os conjuntos de dados e configurações utilizados em cada caso.

Escalas	# Trabalhadores	# Tarefas	# Indivíduos		# Gerações	Método de Avaliação
			μ	λ		
Prática	87	50	-	-	-	-
1	87	50	50	50	3000	Ponderado
2	87	50	50	50	3000	*Fuzzy*

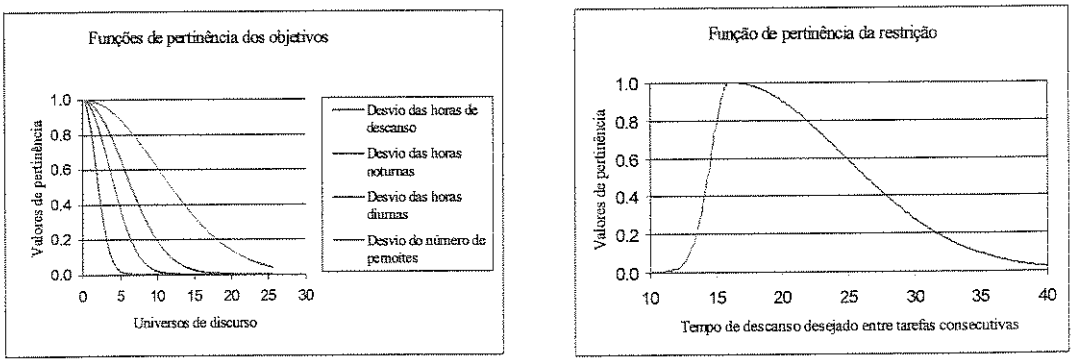
(Prática) - Escala obtida na empresa.

Tabela 5.2. Dados e configurações utilizados.

No caso do método de avaliação ponderado, os coeficientes de ponderação, para cada objetivo de otimização, foram fixados no valor 0.25.

Na metodologia *fuzzy*, a avaliação dos indivíduos está determinada pela confluência dos 4 objetivos de otimização (Capítulo 4) e a restrição do descanso entre duas tarefas consecutivas (Capítulo 3).

As funções de pertinência usadas para representar os diferentes conjuntos (objetivos e restrição) são mostradas na Figura 5.8.



(a) (b)

Figura 5.8. Funções de pertinência, (a) Objetivos, (b) Restrição do descanso entre tarefas.

Para determinar os graus de pertinência dos objetivos, primeiramente são calculados os valores para cada um deles usando as equações da (4.4) a (4.7), com prévia normalização dos parâmetros envolvidos, calculados mediante as equações (4.12) a (4.15) (Capítulo 4).

O valor máximo de cada objetivo é usado como referência para determinar seus universos de discurso. Como percebe-se na Figura 5.8(a), os valores máximos atingidos em cada função de pertinência são para desvios nulos significando um desvio ótimo nos valores de erros.

Por outro lado, a função de pertinência da restrição foi determinada usando a seguinte informação: se consideramos 16 horas como o tempo de descanso requerido ou desejado entre duas tarefas consecutivas, então os valores inferiores não serão desejáveis do ponto de vista dos trabalhadores e valores superiores não serão desejáveis para a diretoria da empresa. No entanto, os ganhos da empresa não são tão fortemente afetados por valores ligeiramente superiores se comparados com o que poderiam afetar valores inferiores na satisfação e produtividade dos trabalhadores.

5.6.- Análise de resultados.

As tabelas de resultados mostram as estatísticas concernentes ao valor médio, variância e desvio padrão dos parâmetros que definem os objetivos de otimização, tais

como : horas diurnas, noturnas, descansos e número de pernoites programados em cada escala. Por outro lado, os histogramas fornecem informação sobre a distribuição em freqüências dos horários programados. Neste caso, os desvios para as horas diurnas, noturnas e horas de descanso, são calculados em valores percentuais e o número de pernoites é mostrado exatamente como a quantidade de pernoites programados para cada trabalhador durante todo o horizonte de planejamento.

5.6.1.- Estudo da programação de escalas sem análise de programas anteriores.

O estudo da programação de escalas, sem analisar programas anteriores, realizou-se fundamentalmente para três casos. Um primeiro caso corresponde a resultados práticos (obtidos por especialistas de uma empresa). Os outros correspondem aos resultados do algoritmo evolutivo usando os dois métodos de avaliação : método ponderado e método *fuzzy*.

A Tabela 5.3 mostra as estatísticas dos resultados práticos e os resultados atingidos pelo algoritmo evolutivo, relativos a cada conjunto de dados relacionado na Tabela 5.2.

Escalas	Horas Diurnas			Horas Noturnas		
	M	V	DP	M	V	DP
Prática	171.78	123.34	11.11	65.52	22.61	4.76
1	171.78	98.64	9.93	65.52	25.06	5.00
2	171.78	185.72	13.63	65.52	12.12	3.48

Escalas	Horas de descanso			Número de pernoites		
	M	V	DP	M	V	DP
Prática	506.33	148.77	12.20	5.34	2.62	1.62
1	506.33	111.34	10.55	5.58	1.08	1.04
2	506.33	201.62	14.19	5.40	0.64	0.80

Tabela 5.3. Resultados obtidos para cada conjunto de dados da Tabela 5.2.

Os resultados gráficos correspondentes (histogramas) são mostrados nas Figuras 5.9, 5.10, 5.11 e 5.12.

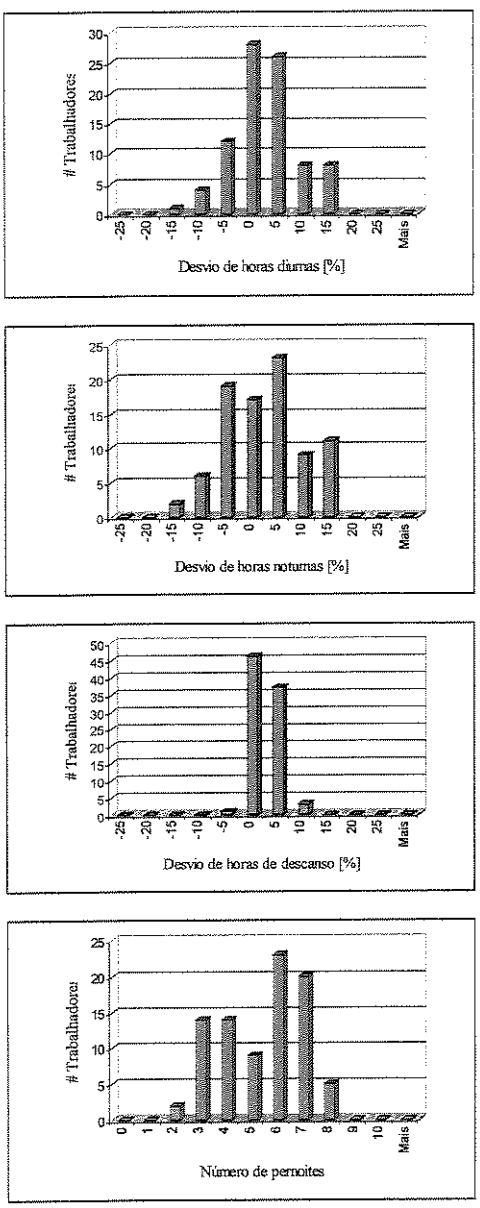


Figura 5.9. Resultados práticos obtidos por especialistas da empresa.

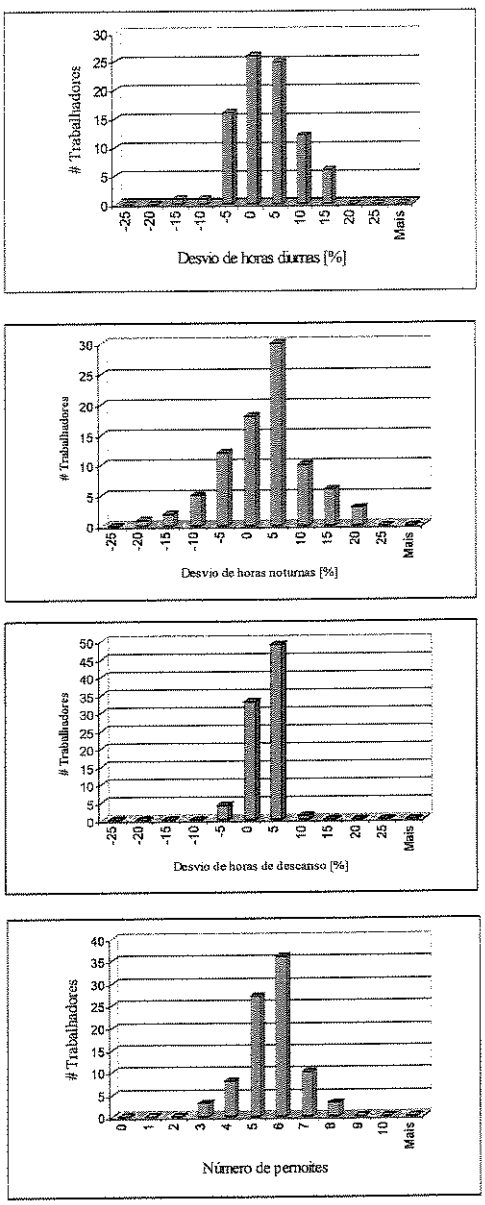
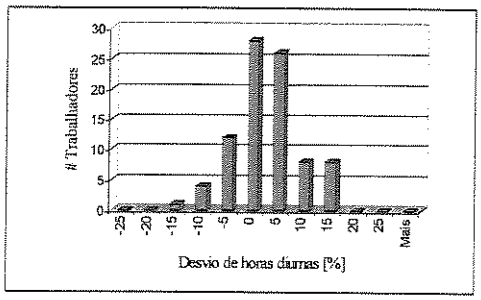
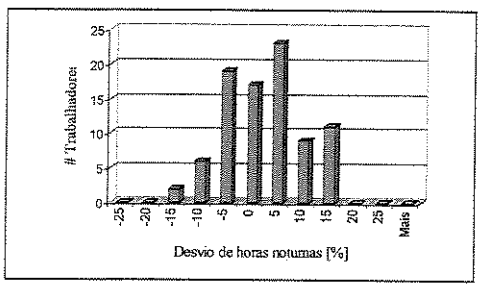


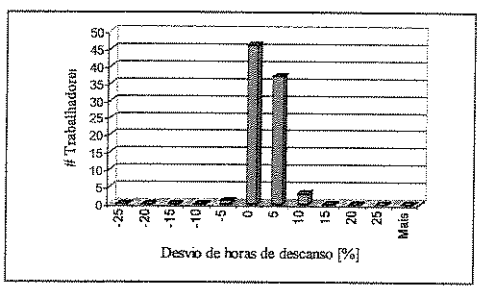
Figura 5.10. Resultados do algoritmo evolutivo usando o método de avaliação ponderada.



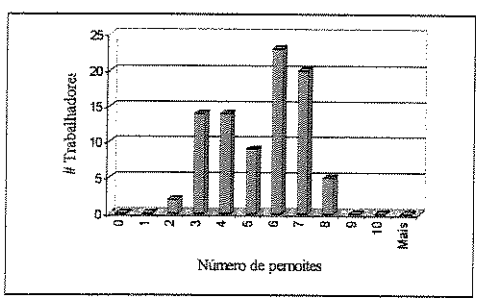
(a)



(b)

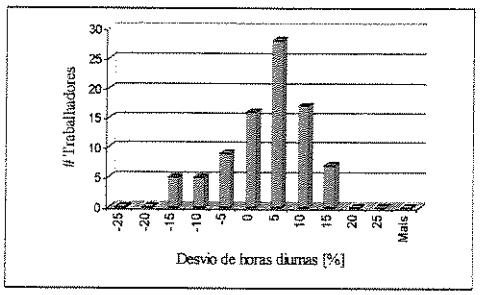


(c)

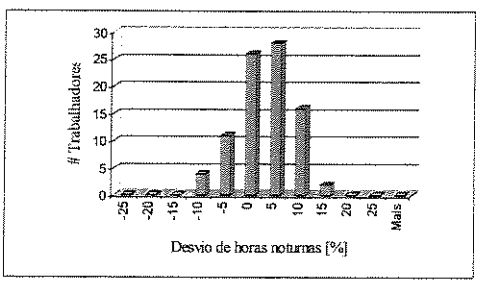


(d)

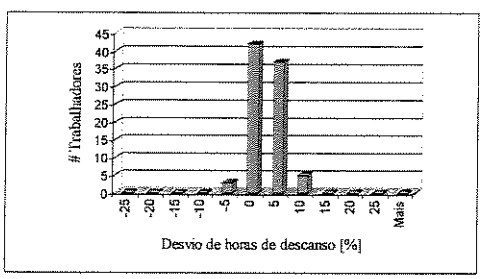
Figura 5.11. Resultados práticos obtidos por especialistas da empresa.



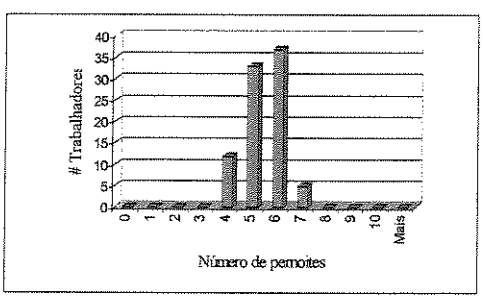
(a)



(b)



(c)



(d)

Figura 5.12. Resultados do algoritmo evolutivo usando o método de avaliação *fuzzy*.

Como observa-se, as estatísticas e os histogramas constatam a efetividade do algoritmo evolutivo quando seus resultados são comparados com os resultados práticos.

Segundo a Tabela 5.3, percebe-se uma menor dispersão a respeito do parâmetro mais crítico que é o número de pernoites programados. Com ambos os métodos de avaliação (ponderado e *fuzzy*), o algoritmo consegue enfatizar em maior dimensão a distribuição eqüitativa do número de pernoites designados para cada trabalhador. Evidentemente, este ponto é fundamental pois, em geral, as jornadas de trabalho noturno repetidas são as menos preferidas.

Outro parâmetro crítico é o número de horas noturnas. Neste caso, os resultados obtidos na prática e os gerados pelo algoritmo com avaliação ponderada, apresentam valores de dispersão com mínimas diferenças, como relacionado na Tabela 5.3. A avaliação *fuzzy* apresentou os melhores resultados neste sentido.

Os resultados por histogramas confirmam também estas conclusões. Segundo as distribuições mostradas nas Figuras 5.10 e 5.12, percebe-se um melhor balanço do número de pernoites programados (para ambos os métodos de avaliação). Por outro lado, a distribuição do número de pernoites mostradas pelo resultado prático (Figuras 5.9 e 5.11) apresenta maior dispersão, variando de dois a oito pernoites para diferentes equipes de trabalho. Isto significa que algumas equipes de trabalho apresentam seis pernoites a mais do que outras.

Os outros elementos (horas diurnas e horas de descanso), para os três casos analisados, mostram comportamentos similares nos resultados das programações efetuadas.

Sendo assim, através das diferentes transformações locais realizadas ao longo dos processos de evolução, melhores distribuições dos horários programados podem ser obtidas, atingindo soluções com bons compromissos (*trade-off*) entre objetivos que são altamente conflitantes.

Concernente à restrição do descanso entre tarefas consecutivas, o valor médio de horas de descanso atingido pelo algoritmo proposto, usando avaliação *fuzzy*, foi de 21.49 horas. Para o método de avaliação ponderada o valor atingiu as 21.83 horas e nos resultados práticos foi de 22.25 horas. Isto confirma a validade do uso da avaliação *fuzzy*, pois nos permite flexibilizar o descanso entre tarefas consecutivas, usando-o como um parâmetro adicional, com determinado grau de importância na determinação da qualidade

de uma solução. Desta forma, podem-se equilibrar as escalas gerando seqüências de tarefas que apresentam melhores compromissos entre jornadas de trabalho e descanso.

5.6.2.- Estudo da programação de escalas com análise de programas anteriores.

O estudo do comportamento do algoritmo evolutivo na geração da programação de escalas, quando analisam-se programas anteriores, foi realizado com o conjunto de dados e configurações relacionados na Tabela 5.4.

Escalas	**# Trabalhadores**	**# Tarefas**	**# Indivíduos**		**# Gerações**	**Programa Anterior**	**Método de Avaliação**
			μ	**λ**			
1	87	50	50	50	1500	N	Ponderado
2	87	50	50	50	1500	1	Ponderado
3	87	50	50	50	1500	2	Ponderado
4	87	50	50	50	1500	N	*Fuzzy*
5	87	50	50	50	1500	1	*Fuzzy*
6	87	50	50	50	1500	2	*Fuzzy*

(N) Sem analisar programa anterior, (1) Com análise de 1 mês anterior, (2) Com análise de 2 meses anteriores.

Tabela 5.4. Dados e configurações utilizados.

Inicialmente, elabora-se uma programação de escalas usando ambos os métodos de avaliação (ponderado e *fuzzy*). Este processo de elaboração de um programa inicial é realizado em um número menor de evoluções, provocando uma convergência prematura (solução menos otimizada). Isto é feito com o propósito de apreciar melhor o desenvolvimento do algoritmo na evolução das soluções posteriores, que precisaram da informação do programa anterior para o balanço dos horários no período total (horizonte de planejamento anterior mais o atual).

Uma vez gerada cada programação inicial, realizam-se as novas simulações (para ambos os métodos de avaliação), levando em consideração o conteúdo da programação inicial na geração de novas escalas. O objetivo é manter as distribuições dos horários eqüitativas ao longo do período total de planejamento, período este constituído pela soma

do período anterior e o atual. Assim, um novo programa é gerado, visando manter as distribuições dos horários da forma mais equânime possível.

Finalmente, para cada método de avaliação, geram-se duas novas programações. Desta vez, consideram-se as duas programações anteriores como histórico do passado, e efetua-se o balanço tentando manter o equilíbrio nas distribuições dos horários.

Os resultados estatísticos correspondentes são relacionados na Tabela 5.5.

Escalas	Horas Diurnas			Horas Noturnas		
	M	V	DP	M	V	DP
1	171.76	292.19	17.09	65.524	15.75	3.97
2	343.68	371.41	19.27	131.25	26.21	5.12
3	515.61	856.55	29.27	197.00	47.10	6.86
4	171.76	297.36	17.24	65.524	24.03	4.90
5	343.71	457.01	21.38	131.24	28.64	5.35
6	515.62	1005.8	31.71	196.99	70.38	8.39

Escalas	Horas de descanso			Número de pernoites		
	M	V	DP	M	V	DP
1	506.33	272.97	16.52	5.73	1.13	1.06
2	1012.7	350.69	18.73	11.73	2.01	1.42
3	1519.0	852.17	29.19	17.81	5.09	2.25
4	506.33	233.36	15.27	5.77	1.46	1.21
5	1012.7	360.64	18.99	11.85	1.94	1.39
6	1519.0	764.93	27.65	17.96	4.15	2.04

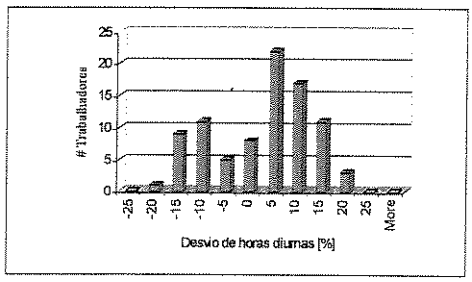
(M) Valor médio, (V) Variância, (DP) Desvio padrão.

Tabela 5.5. Resultados obtidos para cada conjunto de dados da Tabela 5.4.

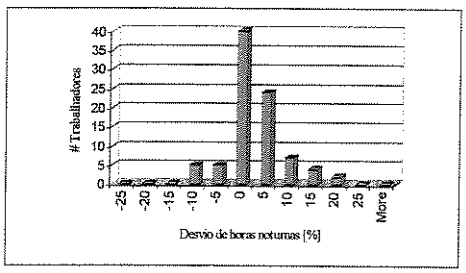
Segundo as estatísticas relacionadas na Tabela 5.5, percebe-se que os valores de dispersão não são muito distantes entre os resultados de uma programação e outra. Logicamente, como analisam-se períodos maiores (horizonte anterior mais o atual), e realizam-se transformações (mutações de tarefas) em períodos menores (horizonte atual), resulta difícil diminuir as dispersões com respeito a programas anteriores. No entanto, através dos processos de evolução é possível compensar os horários distribuídos anteriormente de forma menos eqüitativa, proporcionando uma redução dos efeitos indesejáveis das programações anteriores.

A seguir mostram-se os resultados gráficos (histogramas), correspondentes aos resultados relacionados na Tabela 5.5.

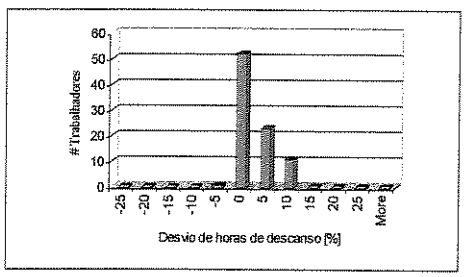
As Figuras 5.13 e 5.14 mostram os resultados da programação inicial e da programação gerada analisando a dita programação inicial (usando o método de avaliação ponderada).



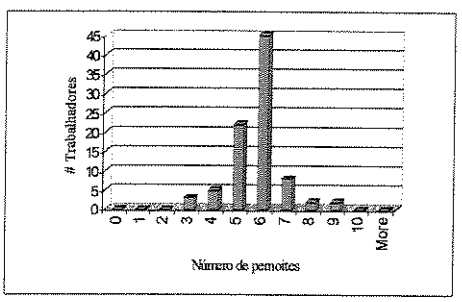
(a)



(b)

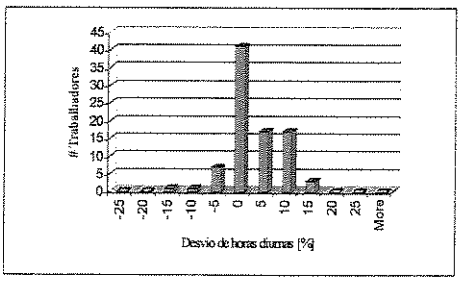


(c)

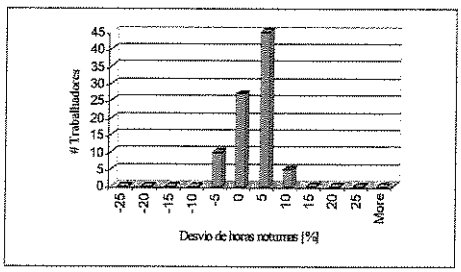


(d)

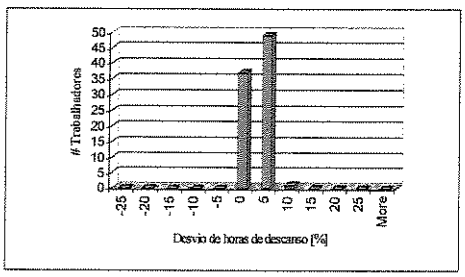
Figura 5.1. Resultados do algoritmo evolutivo usando o método de avaliação ponderada sem análise de programas anteriores.



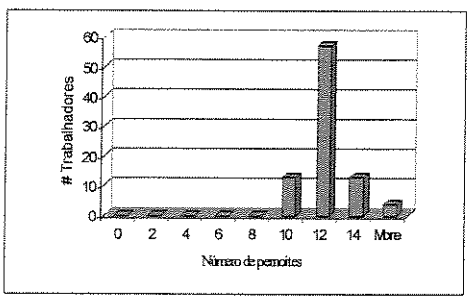
(a)



(b)



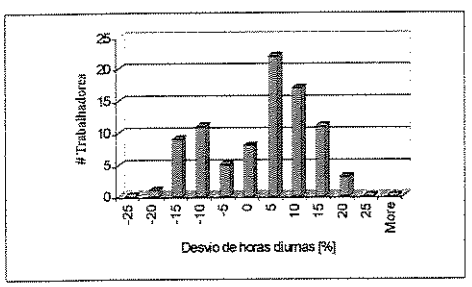
(c)



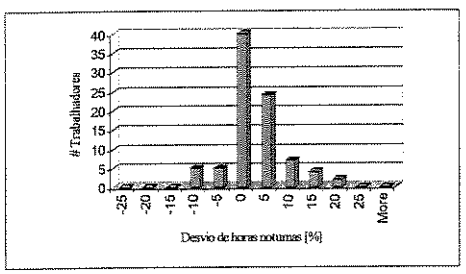
(d)

Figura 5.2. Resultados do algoritmo evolutivo usando o método de avaliação ponderada com análise de 1 mês anterior.

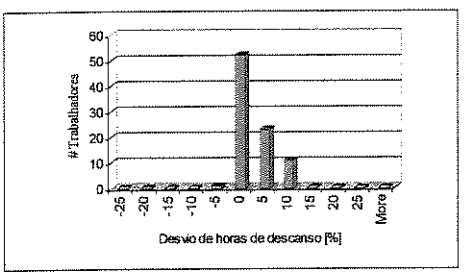
A Figura 5.15 mostra os resultados da programação inicial e a Figura 5.16 mostra a programação gerada analisando as duas programações anteriores, cujos resultados mostram-se nas Figuras 5.13 e 5.14, usando o método de avaliação ponderada.



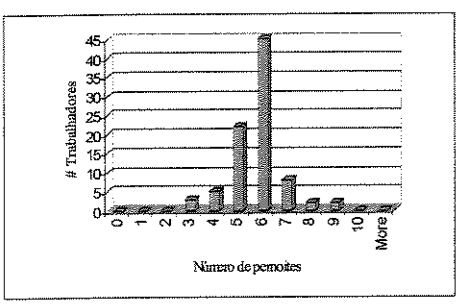
(a)



(b)

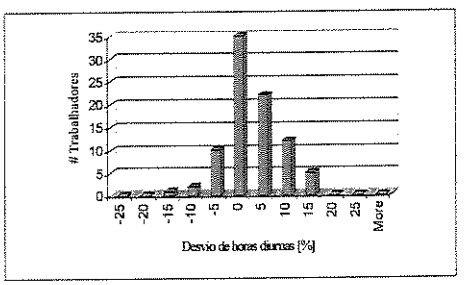


(c)

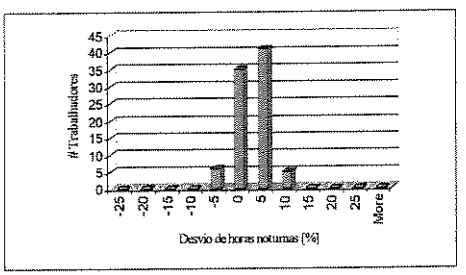


(d)

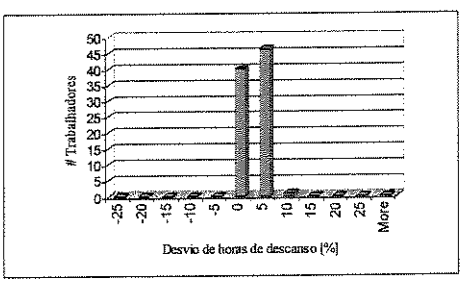
Figura 5.3. Resultados do algoritmo evolutivo usando o método de avaliação ponderada sem análise de programas anteriores.



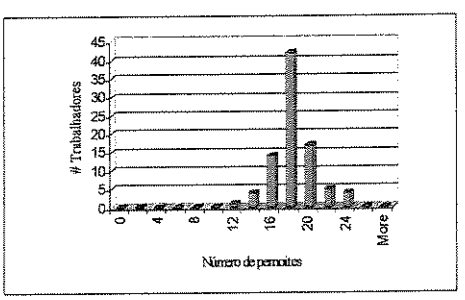
(a)



(b)



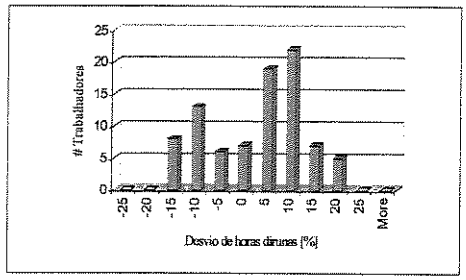
(c)



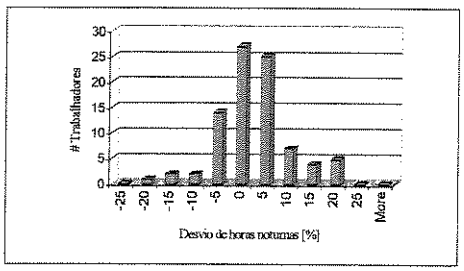
(d)

Figura 5.4. Resultados do algoritmo evolutivo usando o método de avaliação ponderada com análise de 2 meses anteriores.

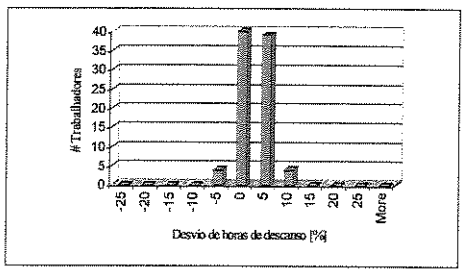
As Figuras 5.17 e 5.18 mostram os resultados da programação inicial e da programação gerada analisando esta programação inicial (usando o método de avaliação *fuzzy*).



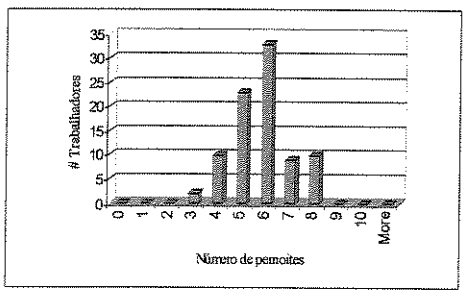
(a)



(b)

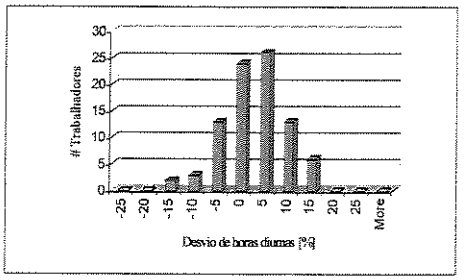


(c)

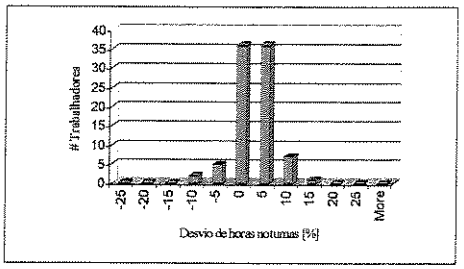


(d)

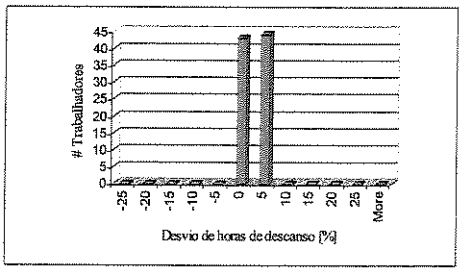
Figura 5.5. Resultados do algoritmo evolutivo usando o método de avaliação *fuzzy* sem análise de programas anteriores.



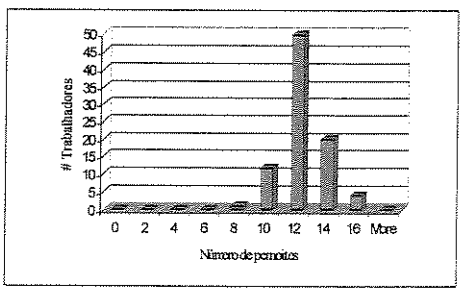
(a)



(b)



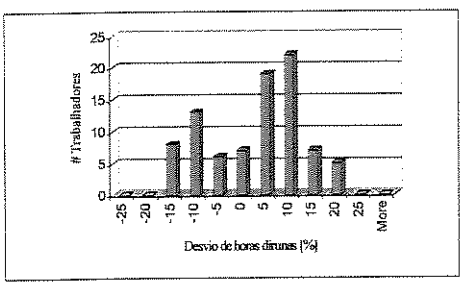
(c)



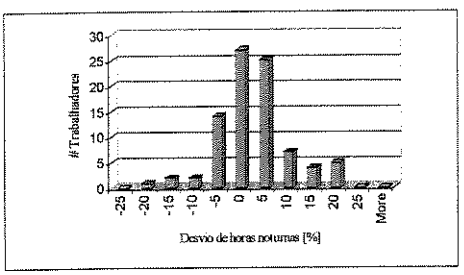
(d)

Figura 5.6. Resultados do algoritmo evolutivo usando o método de avaliação *fuzzy* com análise de 1 mês anterior

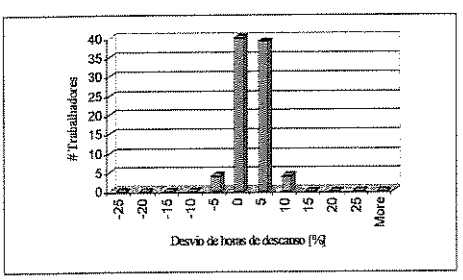
A Figura 5.19 mostra os resultados da programação inicial e a Figura 5.20 mostra os resultados da programação gerada analisando as duas programações anteriores (Figuras 5.17 e 5.18) usando o método de avaliação *fuzzy*.



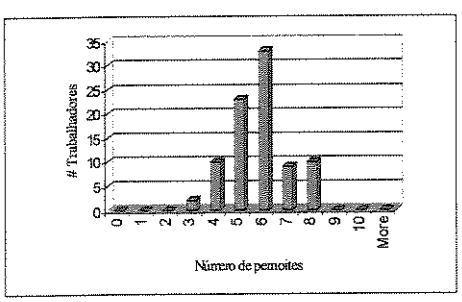
(a)



(b)

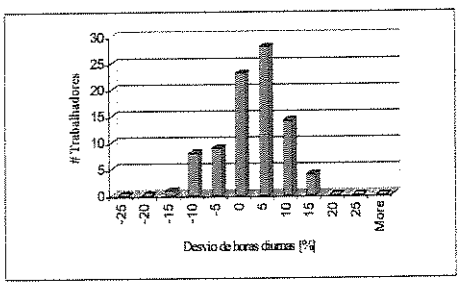


(c)

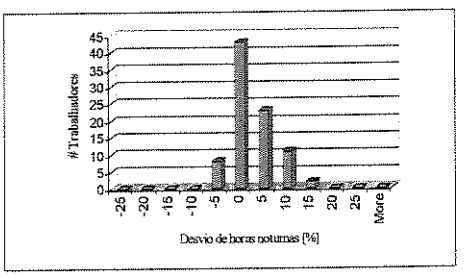


(d)

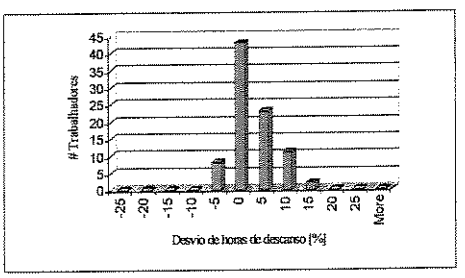
Figura 5.7. Resultados do algoritmo evolutivo usando o método de avaliação *fuzzy* sem análise de programas anteriores.



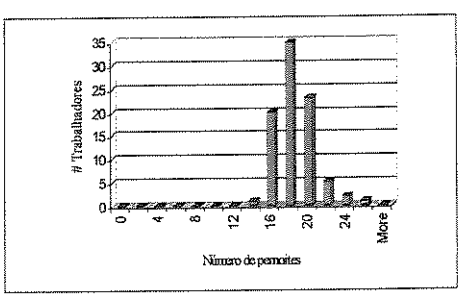
(a)



(b)



(c)



(d)

Figura 5.8. Resultados do algoritmo evolutivo usando o método de avaliação *fuzzy* com análise de 2 meses anteriores

Os histogramas correspondentes a cada uma das escalas mostradas nas figuras anteriores confirmam os efeitos produzidos para que a maioria dos trabalhadores recebam igual valor dos horários programados, dentro dos limites permissíveis, impostos pelo conjunto de dados e restrições analisadas.

5.6.3.- Estudo do comportamento do algoritmo evolutivo.

Pela análise das simulações realizadas, fixando-se o número de variáveis envolvidas (número de tarefas e trabalhadores), os resultados fornecidos pelo algoritmo evolutivo revelam que, com um número não muito elevado de gerações populacionais, pode-se encontrar soluções factíveis.

Os exemplos discutidos anteriormente (resultados da Tabela 5.5), foram obtidos para um total de 1500 gerações (ver Tabela 5.4) e apresentam bons compromissos quando comparados com os resultados práticos. Isto é justificado pelas imposições derivadas das análises das restrições, pois estas impõem fortes limites ao número de soluções factíveis do problema.

O desenvolvimento do algoritmo de evolução, referente aos exemplos das Tabelas 5.4 e 5.5, é mostrado nas Figuras 5.21 e 5.22 através do comportamento do grau de avaliação das soluções.

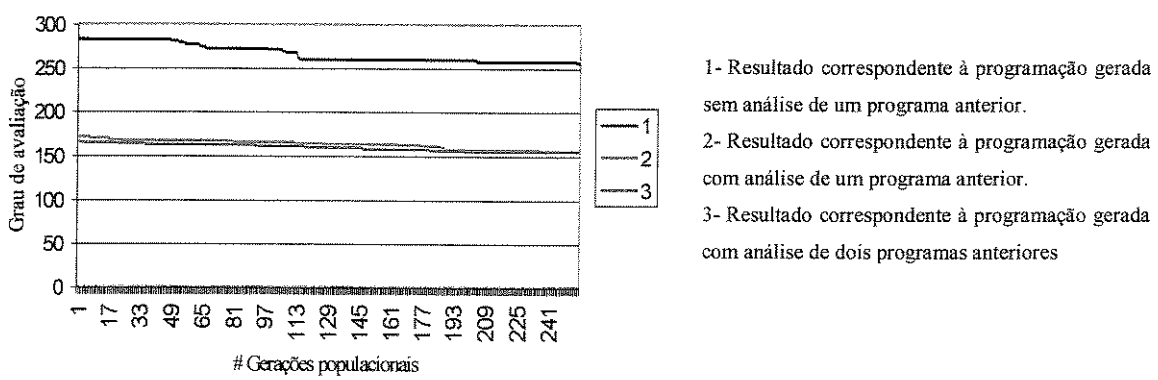
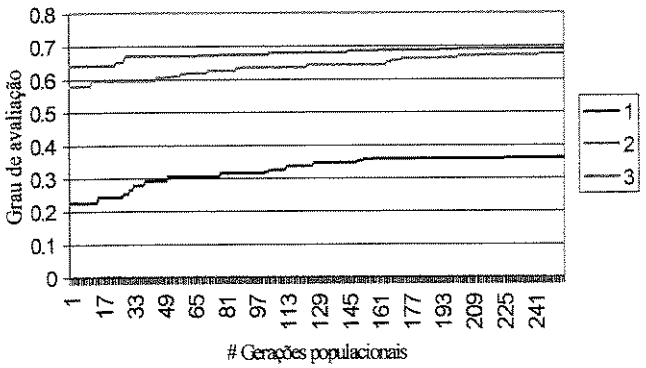


Figura 5.21. Desenvolvimento do algoritmo evolutivo usando o método de avaliação ponderada.



1- Resultado correspondente à programação gerada sem análise de um programa anterior.
2- Resultado correspondente à programação gerada com análise de um programa anterior.
3- Resultado correspondente à programação gerada com análise de dois programas anteriores

Figura 5.22. Desenvolvimento do algoritmo evolutivo usando o método de avaliação *fuzzy*.

Segundo a Figura 5.21, percebe-se que quando é usado o método de avaliação ponderada, o grau de avaliação é monotonamente decrescente, no sentido de diminuir a diferença (desvio) entre cada parâmetro calculado e seu respectivo valor médio (Capítulo 4). Contrário a este resultado, o valor da avaliação, quando é usado o método de avaliação *fuzzy* (Figura 5.22), apresenta um comportamento monotonicamente crescente, indicando o sentido de atingir o valor máximo de pertinência para cada objetivo integrante da formulação da função de avaliação estabelecida.

A respeito do tamanho da população, valores razoáveis observados para o número de indivíduos estiveram concentrados no intervalo de 50 e 100 indivíduos, incluindo pais e filhos. Para valores maiores, podem ser encontradas boas soluções, mas o custo computacional torna-se muito elevado, chegando a valores entre 4 e 5 horas de cálculos. Em contraste, para obter uma solução factível, sem ajuda dos procedimentos de evolução, são necessários em média entre 2 e 4 segundos de execução (usando processador *Pentium* a 300Mhz).

Quando aplicam-se os processos evolutivos, populações com 50 indivíduos induzem tempos de processamento entre 30 e 60 minutos, dependendo do número de gerações, sem aplicação de procedimentos de busca local dentro dos processos evolutivos.

Por outro lado, com busca local uma execução pode durar de 6 a 8 horas de cálculo, dependendo da quantia de soluções vizinhas a analisar. Os resultados analisados com busca local não forneceram melhores soluções apesar de afetar, e muito, o esforço

computacional requerido. Por tal motivo somente os resultados do algoritmo evolutivo sem busca local foram enfatizados.

As probabilidades para cada operação de mutação realizada, tanto simples como múltipla, foram mantidas sempre em valores elevados devido ao efeito causado pelas restrições. Estas restrições limitam fortemente as possibilidades de efetuar transformações que garantiriam a factibilidade das soluções resultantes deste processo.

5.7.- Resumo.

Neste capítulo, apresentou-se a ferramenta computacional desenvolvida para a geração de escalas em empresas ferroviárias. Foram explicadas, de forma sucinta, as principais classes implementadas e os aspectos operacionais das interfaces de entrada/saída, visando um correto uso do sistema.

Em detalhes, mostraram-se e discutiram-se os resultados obtidos das diferentes simulações, comparando-os com um programa elaborado na prática por especialistas desta área.

As análises realizadas confirmam a viabilidade do algoritmo evolutivo pois este, baseado nos critérios de avaliação definidos, oferece soluções que superam qualitativamente os resultados práticos.

Um aspecto importante foi a obtenção de balanços eqüitativos da carga de trabalho quando analisados períodos longos de planejamento, aspecto ignorado na prática atual.

Mostrou-se também a efetividade do método de avaliação *fuzzy*, se constituindo em uma nova metodologia que permite modelar o grau de adaptabilidade dos indivíduos (qualidade das soluções).

Desta forma, o algoritmo proposto constitui uma ferramenta adicional para a elaboração de planos de escalas em empresas ferroviárias, permitindo a geração de resultados factíveis do ponto de vista da satisfação dos trabalhadores e permitindo atender a demanda de transporte requerida em cada período de análise.

Capítulo 6

Conclusões e trabalhos futuros

6.1.- Conclusões

Na atualidade, os problemas de geração de escalas apresentam-se com variadas limitações, condições e critérios a cumprir para a determinação de soluções factíveis, tanto no sentido econômico como na satisfação social.

Nas primeiras etapas deste trabalho foram mostradas as peculiaridades e características envolvidas em problemas gerais de escalonamentos, mas enfatizando a elaboração de escalas de força de trabalho como ponto fundamental de interesse.

Os modelos e métodos estudados confirmam a grande variedade de requisitos nesta área, aspecto que limita o desenvolvimento de metodologias gerais que possam fornecer soluções eficientes para todos ou para uma grande parte destes problemas.

No setor ferroviário, características particulares impõem requisitos adicionais, motivo pelo qual, na atualidade, os resultados são gerados manualmente, onde na maioria das ocasiões ficam distantes dos melhores resultados possíveis.

A metodologia atualmente usada na prática (descrita no Capítulo 3) constitui um passo básico e muito importante na elaboração de escalas, mas precisa de outros mecanismos para complementar as soluções obtidas.

Sendo assim, os procedimentos implementados (baseados em idéias práticas), no algoritmo de evolução desenvolvido, conseguem atuar em pontos não abordados pelas

metodologias usadas na prática, complementando a busca de soluções superiores em vários sentidos.

A ferramenta computacional desenvolvida tem seu fundamento nos avanços das técnicas de computação evolutiva. Mesmo assim, foi necessário a implementação de métodos complementares como processamento heurístico e algoritmo de *bactracking*, indispensáveis na obtenção das soluções iniciais do problema.

Um aspecto importante foram os critérios de avaliação das soluções, para os quais foram definidos e implementados com resultados satisfatórios: um método de soma ponderada e uma metodologia usando conceitos da Teoria de Conjuntos *Fuzzy*.

Por outro lado, o algoritmo oferece soluções em menor tempo de processamento, sem grande esforço humano, com qualidade nas soluções e tornando o especialista da empresa um supervisor em todo este processo tão complexo.

Neste sentido, o algoritmo permite a geração automática das programações de escalas com bons resultados segundo os critérios de avaliação definidos e com a devida satisfação das restrições impostas, além de outros critérios adicionais.

As soluções mostraram balanços uniformes das cargas de trabalho e descanso, mesmo quando analisados períodos longos de planejamento.

Em termos operacionais, o sistema facilita a configuração dos dados através das interfaces de entrada e oferece também facilidade de processamento dos resultados, podendo ser visualizados por tabelas ou por gráficos de distribuição em freqüências (histogramas).

6.2.- Trabalhos futuros

As técnicas de inteligência computacional baseadas nas teorias da evolução ainda precisam se consolidar e ganhar em maturidade, sobretudo quando aplicam-se na solução de problemas de otimização combinatória e restrita.

Algumas questões por resolver referem-se ao desenvolvimento de operadores de recombinação entre cromossomos que possam permitir transformações entre diferentes soluções, aumentando assim a diversidade populacional. Outro aspecto importante deriva-se das formas de codificação das soluções. Neste sentido, fica em aberto as

possibilidades de implementação de novas formas de representação das soluções que permitam a codificação de tarefas que se apresentam com jornadas superiores a 24 horas.

Fatores relacionados com a velocidade de convergência incentivam implementações dos processos evolutivos em paralelo, aproveitando os recursos que cada dia oferecem maiores e melhores possibilidades neste sentido.

Segundo os resultados fornecidos pelo modelo de avaliação *fuzzy* evidencia-se a necessidade de implementação de metodologias que possibilitem a geração automática das funções de pertinência que modelam os conjuntos *fuzzy* dos objetivos e restrições do problema. Com isto, pode-se aumentar a flexibilidade do sistema, considerando as possibilidades de aplicação em empresas diversas. Em outra direção, podem ser incorporados coeficientes de ponderação na metodologia de avaliação *fuzzy* para estabelecer diferenciação entre os diferentes objetivos de otimização.

Finalmente, esperamos que os resultados desta dissertação possam incentivar novos estudos que permitam viabilizar o uso de recursos computacionais na automatização de processos de solução de problemas suficientemente complexos para serem tratados de forma eficaz por especialistas.

Referências bibliográficas

[1] A. Aparecido, "Otimização de Escala de Trabalho para Condutores de Trem: Sequenciamento de Tarefas e Alocação Baseada em Preferência Declarada ", Tese de Doutorado, UFSC, 1997

[2] K. Al-sultan, M. Hussain and J. Nizami, "A Genetic Algorithm for the Set Covering Problem", *Journal of the Operational Research Society*, 47, pp. 702-709, 1996.

[3] T. Aykin, "Optimal Shift Scheduling with Multiple Break Windows", *Management Science*, Vol. 42, No. 4, pp. 591-602, 1996.

[4] K. Baker, R. Burns and M. Carter, "Staff Scheduling with Day-Off and Workstretch Constraints", *AIIE Transactions*, Vol. 11, No. 4, pp. 286-292, 1979.

[5] N. Balakrishnan and R. Wong, "A Network Model for the Rotating Workforce Scheduling Problem", *Networks*, Vol. 20, pp. 25-42, 1990.

[6] R. Bellman and L. Zadeh., "Decision-Making in a Fuzzy Environment", *Management Science*, Vol. 17, No. 4, pp. 141-164, 1970.

[7] W. Brownell and J. Lowerre, "Scheduling of Work Forces Required in Continuos Operations Under Alternative Labor Policies", *Management Science*, Vol. 22, No. 5, pp. 597-605, January, 1976.

[8] R. Burns and M. Carter, "Work Force Size and Single Shift Schedules with Variable Demands", *Management Science*, Vol. 31, No. 5, pp. 599-607, May, 1985.

[9] R. Burns and G. Koop, "A Modular Approach to Optimal Multiple-Shift Manpower Scheduling", *Operations Research*, Vol. 35, No. 1, pp. 100-110, 1987.

[10] A. Caprara, M. Fischetti, P. Toth and D. Vigo, "Modeling and Solving the Crew Rostering Problem", *Operations Research*, Vol. 46, No. 6, 1998.

[11] H. Emmons, "Work-Force Scheduling with Cyclic Requeriments and Constraints on Days Off, Weekends Off, and Work Stretch", *IIE Transactions*, Vol. 17, No. 1, pp. 8-16. March, 1985.

[12] 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.

[13] L. Fogel, A. Owens and M. Walsh, "Artificial Intelligence through Simulated Evolution", John Wiley, NY, 1966.

[14] R. Gonçalves, "Sistemas Inteligentes para Planejamento de Escalas de Equipagens em Sistemas de Transporte : Aplicação a Sistemas Ferroviários". Tese de Doutorado, UNICAMP, 2000.

[15] J. Holland, "Adaptation in Natural and Artificial Systems ", The University of Michigan Press. Reprinted in 1992 by MIT Press.

[16] E. Holmes, "Nurse Scheduling Using Mathematical Programming", *Operations Research*, Vol. 24, No. 5, 1976.

[17] 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.

[18] C. Jordan, "A Two-phase Genetic Algorithm to Solve Variants of the Batch Sequencing Problem", *Int. J. Prod. Res.*, Vol. 36, No. 3, pp. 745-760, 1998.

[19] G. Koop, "Multiple Shift Workforce Lower Bounds", *Management Science*, Vol. 34, No. 10, pp. 1221-1229, October, 1988.

[20] H. Lau, "On the Complexity of Manpower Shift Scheduling", *Computers Ops. Res.*, Vol. 23, No. 1, pp. 93-102, 1996.

[21] H. Lau, "Combinatorial Approaches for Hard Problems in Manpower Scheduling", *Journal of the Operations Research*, Vol. 39, No. 1, pp. 88-98, 1996.

[22] O. Maimon and D. Braha, " A Genetic Algoritmh Approach to Scheduling PCBs on a Single Machine", *Int. J. Prod. Res.*, Vol. 36, No. 3, pp. 761-784, 1998.

[23] F. Maria, M. Jorge, C. Marco and M. Marley, "Optimização de Planejamento de Horários por Algoritmos Genéticos", *Anais do II Congreso de Redes Neurais*, Curitiba, Outubro, 1995.

[24] D. Michael, "Scheduling Nursing Personal According to Nursing Preference : A Mathematical Programming Approach", *Operations Research*, Vol. 24, No. 5, 1976.

[25] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolutionary Programs*, Springer-Verlag, London, 1992.

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

1DJ, 1995.

[27] W. Pedrycz and F. Gomide, "An Introduction to Fuzzy Sets", A Bradford Book, The MIT Press, Cambridge Massachussets, 1998.

[28] L. Ramírez and F. Gomide, "Railway Crew Scheduling using a Fuzzy-Evolutionary Strategy", accepted for the 8th Information Processing and Management of Uncertainty in Knowledge-Based Systems Conference (IPMU 2000), Madrid SPAIN, July, 2000.

[29] I. Rechenberg, "Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution", Frommann-Holzboog, Stuttgart, 1973.

[30] H. Schwefel, "Numerical optimization of computer models", John Wiley, Chichester, 1981.

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

[32] J. Tien and A. Kamiyama, "On Manpower Scheduling Algorithms", *SIAM Review*, Vol. 24, No. 3, pp. 275-287, 1982.

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

[34] Van Bael, P., Devogelaere, M. and Rijckaert, "A Hybrid Evolutionary Search Scheduling Algorithm to Solve the Job Scheduling Problem", *Proceedings of the Congress on Evolutionary Computation-CEC'99*, Washington D.C., U.S.A., Vol. 2, pp. 1104-1108, 1999.

Apêndice I. Descrição das classes principais

Elemento	Nome	Descrição
Classe	Escalas	Interface principal que contêm os menus e submenus correspondentes. Esta classe gerencia o funcionamento da aplicação através dos diferentes enlaces com as outras classes.
Atributos Principais	MenuArquivo	Elementos do menu principal.
	.	"
	.	"
	.	"
	MenuAjuda	"
	MAbrirArquivo	Elementos dos submenus.
	.	"
	.	"
	.	"
	MmanualUsuario	"
	Parametros.	Vetor que guarda os principais parâmetros de configuração usados na aplicação.
Métodos Principais	Escalas	Construtor da classe Escalas.
	LerArquivo	Recebe os dados das tarefas a programar lendo do arquivo selecionado pelo usuário através das interfaces de entrada.
	EditarDescricaoTarefas	Chamada à classe DescricaoTarefas.
	EditarDimensionamentoTarefas	Chamada à classe DimensionamentoTarefas.
	ConfiguraCriteriosProgramacao	Chamada à classe CriteriosProgramacao.
	ConfiguraAlgoritmoEvolutivo	Chamada à classe ParametrosAlgoritmoEvolutivo.
	GeraEscalas	Chamada à classe AlgoritmoEvolutivo para a geração das escalas.
	InterromperGeracao	Chamada à classe AlgoritmoEvolutivo para interromper o processo de geração das escalas.
	ResultadosTabela	Chamada à classe ResultadosTabela para mostrar na tela o sequencial de tarefas resultante.
	ResultadosGraficos	Chamada à classe ResultadosGraficos para mostrar na tela a programação mensal das escalas.
	SobreGESAE	Chamada à classe SobreGESAE para obter informação do sistema GESAE.
	GuiaUsuario	Chamada à classe GuiaUsuario para obter ajuda sobre os procedimentos de trabalho da aplicação.

Elemento	Nome	Descrição
Classe	AlgoritmoEvolutivo	Classe Algoritmo Evolutivo, responsável pela evolução dos seqüenciais de tarefas e as programações mensais de escalas.
Atributos Principais	VetorHorIndDiurnas	Horas individuais diurnas.
	VetorHorIndNoturnas	Horas individuais noturnas.
	VetorPernoites	Pernoites programadas.
	VetorDescansos	Horas de descanso programadas.
	MelhorCromo1	Melhor indivíduo da primeira fase evolutiva.
	PopInic1	População Inicial I.
	PopInter1	População Intermedia I.
	PopSelec1	População Selecionada I.
	Pop_Inic2	População Inicial II.
	PopInter2	População Intermedia II.
	PopSelec2	População Selecionada II.
	Corridas	Número de gerações do processo evolutivo.
	Ncromossomos	Número de cromossomos das populações
	Fitness1	Avaliação dos indivíduos da população da primeira fase evolutiva.
	Fitness2	Avaliação dos indivíduos da população da segunda fase evolutiva.
	VetorTarefas	Identificadores das tarefas a serem programadas.
Métodos Principais	AlgoritmoEvolutivo	Construtor da classe AlgoritmoEvolutivo.
	Inicio	Inicia o funcionamento da classe Escalas.
	AnalisaProgramaAnterior	Método de análise dos dados da programação anterior, caso deva ser gerada uma programação com análise da programação anterior.
	CreaPopulacao1	Geração dos indivíduos da primeira fase evolutiva. Neste método instancia-se a classe Cromossomo1.
	CreaPopulacao2	Geração dos indivíduos da segunda fase evolutiva. Neste método instancia-se a classe Cromossomo2.
	Evolucao1	Método para a primeira fase de evolução.
	Evolucao2	Método para a segunda fase de evolução.
	MutacaoSimples	Método parea efetuar mutações simples dentro do seqüencial de tarefas durante a primeira fase evolutiva.
	MutacaoMultipla	Método parea efetuar mutações múltiplas dentro das programações mensais das escalas
	SelecionaMelhor1	Método para avaliar e selecionar o melhor indivíduo da primeira fase evolutiva. Escolhe o melhor seqüencial de tarefas.
	SelecionaMelhor2	Método para avaliar e selecionar o melhor indivíduo da segunda fase evolutiva. Escolhe a melhor programação mensal das escalas.
	SalvarDados	Método para salvar os dados das programações nos respectivos arquivos de resultados.

Elemento	Nome	Descrição
Classe	Cromossomo1	Classe responsável pela geração dos seqüenciais de tarefas.
Atributos Principais	Parâmetros	Vetor de todos os parâmetros utilizados na aplicação. Estes parâmetros são configurados via interface com o usuário.
	HorTarefasNoturnasExtensivas	Número de horas consideradas para avaliar as tarefas noturnas extensivas.
	VetorTarefas	Vetor que guarda os identificadores das tarefas a programar.
	VetorAuxiliarTarefas	Vetor auxiliar das tarefas que vão sendo programadas, ordenadas aleatoriamente.
	VetorProgramacao	Vetor de programação de tarefas equivalente ao seqüencial de tarefas.
Métodos Principais	Cromossomo1	Construtor da classe Cromossomo1.
	GeraVetorProgramacao1	Inicia os procedimentos de geração de escalas elaborando os seqüenciais de tarefas.
	AnalisaTarefaNoturnaExtensiva	Método de análise da categoria de tarefa noturna extensiva para alocar um dia sem programação dependendo do retorno deste método. Retorno TRUE se a tarefa é noturna extensiva, FALSE caso contrário.
	AnalisaRestricaoDescanso	Método de análise da restrição de descanso entre tarefas consecutivas. Retorno TRUE se existem tarefas que podem ser alocadas no seqüencial, FALSE caso contrário
	AnalisaAntesFolga	Método de análise para a alocação de tarefas antes do dia de folga. Retorna TRUE se a existem tarefas que podem ser alocadas antes do dia de folga, FALSE caso contrário
	AnalisaDepoisFolga	Método de análise para a alocação de tarefas depois do dia de folga. Retorna TRUE se a existem tarefas que podem ser alocadas antes do dia de folga, FALSE caso contrário
	CompletaProgramacao	Método que completa o seqüencial caso a geração das tarefas no atinge o valor correspondente ao número de trabalhadores.
	AvaliaRepeticaoTarefasNoturnas	Método de avaliação do seqüencial a respeito da repetição de tarefas noturnas. Retorna TRUE se o número de tarefas noturnas consecutivas supera o valor configurado pelo usuário, FALSE caso contrário.
	CalculoHorasIndividuais	Método para o cálculo das horas diurnas e noturnas em cada tarefa.
	CalculoHorasDescanso	Método para o cálculo das horas de descanso.
	CalculoPernoites	Método para o cálculo do número de pernoites programadas em cada seqüencial.
	MutacaoSimples	Método para analisar e efetuar mutação no cromossomo caso se cumpram as restrições.
	RetornaCromossomo1	Método para retornar o seqüencial de tarefas gerado. Solução factível.

Elementos	Nome	Descrição
Classe	Cromossomo2	Classe responsável pela geração das programações mensais das escalas.
Atributos Principais	Cromo1	Melhor seqüencial obtido no programa dos mês anterior.
	TarefasDiaFinalProgramaAnterior	Vetor que guarda as tarefas do último dia do programa anterior.
	UltimaDiaFolga	Vetor que guarda o ultimo dia de folga no programa anterior para cada trabalhador.
Métodos Principais	Cromossomo2	Construtor da classe Cromossomo2.
	GeraProgramacaoMensal	Construção da programação mensal com ou sem análise da programação anterior.
	RetornaProgramacaoMensal	Retorno da programação mensal obtida.

Elementos	Nome	Descrição
Classe	Tarefas	Cria objetos a partir dos principais dados das tarefas a programar. Neste sentido, cada objeto resultante contêm informação do identificador, a descrição, o código, o horário inicial e o horário final de uma determinada tarefa.
Atributos Principais	Identificador	Identificador da tarefa.
	Codigo	Código da tarefa
	Descricao	Descrição da tarefa.
	HoraInicio	Hora Inicial da tarefa.
	HoraFim	Hora Final da tarefa.
Métodos Principais	Tarefas	Construtor da classe Tarefas.
	GetId	Método para retornar o identificador da tarefa.
	getCodigo	Método para retornar o código da tarefa.
	getDescricao	Método para retornar a descrição da tarefa.
	getHoraInicial	Método para retornar o horário inicial da tarefa.
	getHoraFinal	Método para retornar o horário final da tarefa.
	SetID	Método para atribuir um identificador.
	SetCodigo	Método para atribuir um código.
	SetDescricao	Método para atribuir uma descrição.
	SetHoraInicial	Método para atribuir um horário inicial
	SetHoraFinal	Método para atribuir um horário final.

Elementos	Nome	Descrição
Classe	Horários	Calcula os horários das tarefas e analisa categorias de tarefas noturnas e noturnas extensivas.
Atributos Principais	HorTarefasNoturnasExtensivas	Número de horas consideradas para avaliar as tarefas noturnas extensivas.
	VetorTarefas	Vetor que guarda os identificadores das tarefas a programar.
Métodos Principais	Horários	Construtor da classe Horários
	CalculoCodigo	Cálculo do código da tarefa em função do identificador.
	RetornoHoraInicial	Retorno do horário inicial da tarefa.
	RetornoHoraFinal	Retorno do horário final da tarefa.
	RetornoJornada	Retorno da jornada da tarefa.
	AnalisaTarefaNoturna	Método para analisar se a tarefa é noturna.
	AnalisaTarefaNoturnaExtensiva	Método para analisar se a tarefa é noturna extensiva.

Índice remissivo das referências bibliográficas

L. Fogel *et al.*	Artificial Intelligence through Simulated Evolution.	45
R. Gonçalves	Sistemas Inteligentes para Planejamento de Escalas de Equipagens em Sistemas de Transporte com Aplicação a Sistemas Ferroviários.	3, 29
J. Holland	Adaptation in Natural and Artificial Systems.	45
E. Holmes	Nurse Scheduling Using Mathematical Programming.	3, 24
H. Ishibuchi and T. Murata	A Multi-Objective Genetic Local Search Algorithm and Its Applications to Flowshop Scheduling.	4, 24, 26, 46, 49, 54, 76
C. Jordan	A Two-phase Genetic Algorithm to Solve Variants of the Batch Sequencing Problem.	46
G. Koop	Multiple Shift Workforce Lower Bounds.	20
H. Lau	On the Complexity of Manpower Shift Scheduling.	5
H. Lau	Combinatorial Approaches for Hard Problems in Manpower Scheduling.	5
O. Maimon and D. Braha	A Genetic Algoritmh Approach to Scheduling PCBs on a Single Machine.	5, 46
F. Maria *et al.*	Optimização de Planejamento de Horários por Algoritmos Genéticos.	3, 4, 24
D. Michael	Scheduling Nursing Personal According to Nursing Preference : A Mathematical Programming Approach.	3, 24
Z. Michalewicz	Genetic Algorithms + Data Structures = Evolutionary Programs.	24, 46, 49, 54, 55, 76
B. Paechter *et al.*	Two Solutions to the General Timetable Problem.	3, 4, 24

W. Pedrycz and F. Gomide	An Introduction to Fuzzy Sets.	79
L. Ramírez and F. Gomide	Railway Crew Scheduling using a Fuzzy-Evolutionary Strategy.	-
I. Rechenberg	Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.	45
H. Schwefel	Numerical optimization of computer models.	45
J. Tanomaru	Planejamento de Mão-de-Obra via um Algoritmo Genético.	26, 49, 50
J. Tien and A. Kamiyama	On Manpower Scheduling Algorithms.	5, 16, 19, 23
S. Uckun *et al.*	Managing Genetic Search in Job Shop Scheduling.	4, 24, 46, 49, 50, 52
Van Bael *et al.*	A Hybrid Evolutionary Search Scheduling Algorithm to Solve the Job Scheduling Problem.	4, 24, 26, 46, 49