DEPARTAMENTO DE COMPUTAÇÃO

D E C O M

Problema da Programação da Tripulação de um sistema de transporte público via Método de Pesquisa em Vizinhança Variável (VNS)

Euler Horta Marinho – 99.1.4024

Prof. Dr. Marcone Jamilson Freitas Souza

Relatório Técnico – (...)

U F O P

UNIVERSIDADE FEDERAL DE OURO PRETO

Resumo

Este trabalho aborda o Problema da Programação da Tripulação (PPT ou *Bus Crew Scheduling Problem*) de uma empresa que opera no sistema de transporte público. Tal problema consiste em atribuir as diferentes tarefas relacionadas à condução dos veículos de tal forma que as viagens das diferentes linhas da empresa sejam executadas com o menor custo possível. Neste trabalho, o PPT foi abordado utilizando o Método de Pesquisa em Vizinhança Variável (VNS) considerando como movimentos de vizinhança, para a Programação Diária, a inserção de tarefas a um conjunto de tripulações e a troca de tarefas entre duas tripulações. Para a Programação Mensal, foi considerado como movimento de vizinhança a troca de jornadas ou agrupamentos de jornadas entre duas tripulações. A função objetivo está voltada para a eliminação de inviabilidades, tal como a sobreposição de tarefas de uma mesma tripulação, e a redução dos custos operacionais. A redução dos custos se dá principalmente com a minimização do número de tripulações e a utilização adequada da mão-de-obra. O algoritmo foi testado com dados reais de uma empresa que opera no município de Belo Horizonte-MG.

Agradecimentos

Agradeço a Deus em primeiro lugar, aos meus pais, meus irmãos e à Flávia por todo apoio dado durante o desenvolvimento deste trabalho. Como não poderia deixar de mencionar agradeço, especialmente, ao Prof. Dr. Marcone pelo apoio e atenção prestados durante o desenvolvimento deste.

Índice

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 Motivação

O Problema da Programação da Tripulação (PPT) de um Sistema de Transporte Público consiste em determinar o número mínimo necessário de tripulações (motoristas e cobradores), tal que a programação dos veículos (conjunto de viagens atribuídas a cada veículo) seja realizada com sucesso. A resolução deste problema resulta no seqüenciamento das atividades de cada tripulação, gerando um conjunto de jornadas de trabalho cujo custo operacional total seja mínimo.

A necessidade cada dia maior de reduzir os custos das empresas tem feito com que elas busquem aprimorar cada vez mais seus processos produtivos, mantendo a qualidade de seus produtos e/ou serviços. Com as privatizações que ocorreram no sistema de transporte público, as companhias do setor entraram neste grupo procurando utilizar, de maneira eficiente, seus recursos materiais e humanos para se manterem lucrativas, sem, no entanto, comprometer a qualidade do serviço oferecido. Como a mão-de-obra operacional é umas das componentes que mais pesam na planilha de custos, uma pequena redução neste item pode significar um ganho considerável no custo total (Bouzada 2002), o que justifica qualquer trabalho no sentido de minimizar os custos com a mão-de-obra.

Embora o PPT tenha sido largamente estudado e aplicado nos países mais desenvolvidos, suas técnicas de resolução são pouco difundidas e raramente aplicadas à nossa realidade. Isso se deve, em parte, pelo estágio primário em que se encontram as empresas do setor de transporte público no Brasil. Por outro lado, as principais restrições presentes neste problema estão relacionadas ao cumprimento das legislações trabalhistas e às normas operacionais vigentes nas empresas que atuam no sistema. Tais fatores impedem que um modelo desenvolvido no Reino Unido, por exemplo, seja aplicado no Brasil. Ademais, as restrições operacionais vigentes nas empresas e as cláusulas contidas nas Convenções Coletivas de Trabalho tornam o problema complexo e intratável computacionalmente, inviabilizando sua solução por técnicas de programação matemática, ditas exatas.

O desenvolvimento de um método de solução eficiente para o PPT é, portanto, de grande importância.

1.2 Objetivo

Neste trabalho é proposta uma aplicação do método de Pesquisa em Vizinhança Variável (VNS) para solucionar o Problema da Programação da Tripulação (PPT) de uma empresa de transporte público urbano, bem como a disponibilização de uma ferramenta que forneça uma boa interação do usuário com o sistema.

2 Revisão Bibliográfica

Vários autores têm se dedicado ao desenvolvimento de modelos capazes de gerar, com menor custo possível, escalas de tripulações que satisfaçam às restrições impostas. Este é um problema permanentemente estudado uma vez que as realidades dos sistemas de transporte estão em contínua transformação e exigem cada vez mais uma gerência eficiente dos recursos disponíveis. Prova disso é do tema nos congressos especializados nas últimas décadas (Voβ & Daduna 2001, Wilson 1999, Daduna et al. 1995, Desrochers & Rousseau 1992).

Os sistemas heurísticos foram os primeiros a serem utilizados na resolução do PPT (Elias 1964). Tais sistemas consistem na automação do trabalho antes realizado manualmente, entretanto eles não foram capazes de detectar possibilidades de otimização. Manington & Wren (1975) iniciaram a inclusão de procedimentos de otimização neste tipo de sistema (Wren & Rousseau 1995). Com o surgimento de metaheurísticas tais como Algoritmos Genéticos (Goldberg 1989), Busca Tabu (Glover 1989, Glover 1990), Simulated Annealing (Kirkpatrick et al., 1983) entre outras, abriu-se um novo horizonte na resolução de problemas NP-difíceis como o PPT. Embora tais métodos não garantam a obtenção do ótimo global, eles permitem incluir com facilidade qualquer tipo de restrição. Dentre os trabalhos metaheurísticos mais relevantes que envolvem a programação da tripulação destacamos: Clement & Wren (1995), Wren & Wren (1995), Kwan, et al. (1999) e Shen & Kwan (2001).

2.1 Revisão dos métodos utilizados

2.1.1 GRASP (Greedy Randomized Adaptative Search Procedure)

GRASP (Procedimento de Busca Adaptativa Gulosa e Randomizada) é um método iterativo, proposto em (Feo & Resende, 1995), que consiste de duas fases: uma fase de construção na qual uma solução é gerada, elemento a elemento, e de uma fase de busca local na qual um ótimo local na vizinhança da solução construída é pesquisado. A melhor solução encontrada ao longo de todas as iterações GRASP realizadas é retornada como resultado. O pseudocódigo descrito na figura 3.2.1 ilustra um procedimento GRASP.

```
procedimento GRASP( f(.), g(.), N(.), GRASPmax, s);
1.  f* ← ∞;
2.  para (Iter = 1,2,...,GRASPmax) faça
3.      Construcao (g(.), α, s);
4.      BuscaLocal (f(.), N(.), s);
5.      se (f(s) < f*) então
6.          s* ← s;
7.          f* ← f(s);
8.      fim-se;
9.  fim-para;
10. s ← s*;
11. Retorne s;
fim GRASP
```

Figura 2.1.1.1: Algoritmo GRASP

Na fase de construção, uma solução é iterativamente construída, elemento por elemento. A cada iteração dessa fase, os próximos elementos candidatos a serem incluídos na solução são colocados em uma lista C de candidatos, seguindo um critério de ordenação pré-determinado. Esse processo de seleção é baseado em uma função adaptativa gulosa $g:C \rightarrow \Re$, que estima o benefício da seleção de cada um dos elementos. A heurística é adaptativa porque os benefícios associados com a escolha de cada elemento são atualizados em cada iteração da fase de construção para refletir as mudanças oriundas da seleção do elemento anterior. A componente probabilística do procedimento reside no fato de que cada elemento é selecionado de forma aleatória a partir de um subconjunto restrito formado pelos melhores elementos que compõem a lista de candidatos. Este subconjunto recebe o nome de lista de candidatos restrita (LCR). Esta técnica de escolha permite que diferentes soluções sejam geradas em cada iteração GRASP. Seja $\alpha \in [0,1]$ um dado parâmetro. O pseudocódigo representado pela figura 3.2.2 descreve a fase de construção GRASP.

```
procedimento Construcao ( g(.), α, s);
a)  s ← ∅;
b)  Inicialize o conjunto C de candidatos;
c)  enquanto (C ≠ ∅) faça
d)     t_min = min{g(t) | t∈ C};
e)     t_max = max{g(t) | t ∈ C};
f)     LCR = {t ∈ C | g(t) ≤ t_min + α (t_max – t_min)};
g)     Selecione, aleatoriamente, um elemento t ∈ LCR;
h)     s ← s ∪ {t};
i)     Atualize o conjunto C de candidatos;
j)  fim-enquanto;
k)  Retorne s;
fim Construcao;
```

Figura 2.1.1.2: Fase de construção de um algoritmo GRASP

Observamos que o parâmetro α controla o nível de gulosidade a aleatoriedade do procedimento *Construcao*. Um valor $\alpha = 0$ faz gerar soluções puramente gulosas, enquanto $\alpha = 1$ faz produzir soluções totalmente aleatórias.

Assim como em muitas técnicas determinísticas, as soluções geradas pela fase de construção GRASP provavelmente não são localmente ótimas com respeito à definição de vizinhança adotada. Daí a importância da fase de busca local, a qual objetiva melhorar a solução construída. A figura 3.2.3 descreve o pseudocódigo de um procedimento básico de busca local com respeito a uma certa vizinhança N(.) de s.

```
procedimento BuscaLocal ( f(.), N(.), s)
1.  s* ← s;  {Melhor solução encontrada}
2.  V = {s' ∈ N(s) | f(s') < f(s)};
3.  enquanto ( |V| > 0) faça
4.     Selecione s ∈ V;
5.     se ( f(s) < f(s*) ) então s* ← s ;
6.     V = {s' ∈ N(s) | f(s') < f(s)};
7.  fim-enquanto;
8.  s ← s*;
9.  Retorne s;
fim BuscaLocal;
```

Figura 2.1.1.3: Fase de busca local de um algoritmo GRASP

A eficiência da busca local depende da qualidade da solução construída. O procedimento de construção tem então um papel importante na busca local, uma vez que as

soluções construídas constituem bons pontos de partida para a busca local, permitindo assim acelerá-la.

O parâmetro α, que determina o tamanho da lista de candidatos restrita, é basicamente o único parâmetro a ser ajustado na implementação de um procedimento GRASP. Em (Feo & Resende, 1995) discute-se o efeito do valor de α na qualidade da solução e na diversidade das soluções geradas durante a fase de construção. Valores de α que levam a uma lista de candidatos restrita de tamanho muito limitado (ou seja, valor de α próximo da escolha gulosa) implicam em soluções finais muito próxima àquela obtida de forma puramente gulosa, obtidas com um baixo esforço computacional. Em contrapartida, provocam uma baixa diversidade de soluções construídas. Já uma escolha de α próxima da seleção puramente aleatória leva a uma grande diversidade de soluções construídas mas, por outro lado muitas das soluções construídas são de qualidade inferior, tornando mais lento o processo de busca local.

O procedimento GRASP procura, portanto, conjugar bons aspectos dos algoritmos puramente gulosos, com aqueles dos procedimentos aleatórios de construção de soluções.

Procedimentos GRASP mais sofisticados incluem estratégias adaptativas para o parâmetro α. O ajuste desse parâmetro ao longo das iterações GRASP, por critérios que levam em consideração os resultados obtidos nas iterações anteriores, produz soluções melhores do que aquelas obtidas considerando-o fixo (M. Prais & C.C. Ribeiro).

2.1.2 Método de Pesquisa em Vizinhança Variável (VNS)

O Método de Pesquisa em Vizinhança Variável (*Variable Neighborhood Search*, VNS) é um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança. Contrariamente a outras metaheurísticas baseadas em métodos de busca local, o método VNS não segue uma trajetória, mas sim explora vizinhanças gradativamente mais "distantes" da solução corrente e focaliza a busca em torno de uma nova solução se e somente se um movimento de melhora é realizado. O método inclui, também, um procedimento de busca local a ser aplicado sobre a solução corrente. Esta rotina de busca local também pode usar diferentes estruturas de vizinhança. O pseudocódigo do algoritmo é apresentado abaixo. Detalhes adicionais desse algoritmo podem ser encontrados em Mladenovic & Hansen (1997, 1999).

```
procedimento VNS
1.  Seja s0 uma solução inicial e r o número de estruturas diferentes de vizinhança;
2.  s ← s0;                    {Solução corrente}
3.  enquanto (Critério de parada não satisfeito) faça
4.     k ← 1;                  {Tipo de estrutura de vizinhança}
5.     enquanto (k ≤ r) faça
6.        Gere um vizinho qualquer s' ∈ N(k)(s);
7.        s'' ← BuscaLocal(s');
8.        se  f(s'') < f(s)
9.           então  s ← s'';
10.                 k ← 1;
11.          senão k ← k + 1;
12.       fim-se;
13.    fim-enquanto;
14. fim-enquanto;
15. Retorne s;
fim VNS;
```

Figura 2.1.2.1: Algoritmo VNS

Nesse algoritmo, parte-se de uma solução inicial qualquer e a cada iteração seleciona-se aleatoriamente um vizinho *s'* dentro da vizinhança $N^{(k)}(s)$ da solução *s* corrente. Esse vizinho é então submetido a um procedimento de busca local. Se a solução ótima local, *s''*, for melhor que a solução *s* corrente, a busca continua de *s''* recomeçando da primeira estrutura de vizinhança $N^{(1)}(s)$. Caso contrário, continua-se a busca a partir da próxima estrutura de vizinhança $N^{(k+1)}(s)$. Este procedimento é encerrado quando uma condição de parada for atingida, tal como o tempo máximo permitido de CPU, o número máximo de iterações ou número máximo de iterações consecutivas entre dois melhoramentos. A solução s' é gerada aleatoriamente no passo 6 de forma a evitar ciclagem, situação que pode ocorrer se alguma regra determinística for usada.

Método de Descida em Vizinhança Variável (VND)

O Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND) é um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada. O pseudocódigo desse algoritmo é apresentado pela Figura 5. Detalhes adicionais desse algoritmo podem ser encontrados em Mladenovic & Hansen (1997, 1999).

```
procedimento VND

1.  Seja s0 uma solução inicial e r o número de estruturas diferentes de vizinhança;
2.  s ← s0;            {Solução corrente}
3.  k ← 1;             {Tipo de estrutura de vizinhança}
4.  enquanto (k ≤ r) faça
5.      Encontre o melhor vizinho s' ∈ N(k)(s);
6.      se  f(s') < f(s)
7.          então  s ← s';
                   k ← 1;
8.          senão k ← k + 1;
9.      fim-se;
10. fim-enquanto;
11. Retorne s;
fim VND;
```

Figura 2.1.2.2: Algoritmo VND

3 Metodologia

Na seção 3.1 será descrito o problema da Programação Diária da Tripulação, na seção 3.2, o VNS aplicado ao problema da Programação Diária da Tripulação, na seção 3.3, o problema da Programação Mensal da Tripulação e na seção 3.4, o VNS aplicado ao problema da Programação Mensal da Tripulação.

3.1 Programação Diária da Tripulação

No transporte público, usualmente a programação da tripulação é feita após a programação dos veículos. Nesta as viagens são reunidas em *Blocos*. Um bloco apresenta a seqüência de viagens que um determinado veículo tem que realizar em um dia, começando e terminando na garagem. Cada bloco mostra também as *Oportunidades de Troca* (OT). A OT é um intervalo de tempo suficiente para haver a troca das tripulações. A cada oportunidade de troca está associado um *Horário de Troca* (HT) e um *Ponto de Troca* (PT), os quais representam, respectivamente, a hora e o local onde poderá haver uma troca de tripulação.

A partir do bloco de um veículo são criadas as *Tarefas*. Cada tarefa é um conjunto de viagens reunidas de maneira que haja apenas duas OT: uma no início e outra no final da tarefa. Assim, durante sua realização não é possível que haja troca de tripulação.

A programação de uma tripulação é formada por um conjunto de tarefas, chamado de *Jornada*. As jornadas são divididas em dois tipos: *Pegada Simples* ou *Dupla Pegada*. No primeiro tipo as tarefas são realizadas de uma única vez e os intervalos de tempo entre as tarefas são iguais ou menores que 2 horas. Caso ocorra um intervalo maior que duas horas a jornada é classificada como dupla pegada. Este intervalo não é contabilizado na remuneração da tripulação.

Ao se reunir as tarefas formando as jornadas, deve-se levar em conta inúmeras restrições operacionais e trabalhistas, tais como:

a) A troca da tripulação só pode ocorrer em pontos preestabelecidos;

b) A tripulação só pode operar veículos de um mesmo grupo de linhas;

c) A jornada diária de trabalho é de 7:10 horas, podendo ser acrescida de no máximo 2:00 horas extras;

d) Uma tripulação tem direito a 30 minutos de descanso durante sua jornada diária de trabalho, podendo este período ser fracionado em intervalos menores, desde que um deles seja maior ou igual a 15 minutos. Restrições deste tipo tornam problemas reais intratáveis computacionalmente, justificando a utilização de metaheurísticas na resolução dos mesmos.

3.2 VNS aplicado ao Problema da Programação Diária da Tripulação

3.2.1 A Função Objetivo

A função a ser minimizada, que avalia a programação da tripulação, tem os seguintes requisitos:

i) **Requisitos essenciais**:

(a) *Sobreposição de Tarefas*: uma tripulação não pode realizar mais de uma tarefa ao mesmo tempo;
(b) *Troca de Pontos Proibida*: uma tripulação não pode realizar duas tarefas consecutivas na qual o ponto de troca do final da primeira seja diferente do ponto de troca do início da segunda, e o intervalo entre elas seja menor do que o tempo necessário para ir de um ponto a outro;
(c) *Horas Excedentes*: por determinações trabalhistas uma tripulação não pode exercer mais que 9:10 horas de atividades diárias;
(d) *Folga Acumulada Deficiente*: durante a realização de uma jornada de trabalho a tripulação tem direito a uma folga de 30 minutos destinada à refeição. Por acordo feito pelos trabalhadores, esta folga pode ser dividida e distribuída no decorrer do dia, conquanto que uma das frações seja de pelo menos 15 minutos. Este sistema contempla automaticamente esse quesito, sendo que este não é contabilizado no cálculo da função objetivo;
(e) *Tempo entre Jornadas:* o tempo mínimo entre jornadas de trabalho é de 11:00. A jornada da tripulação que não atingir esse valor será penalizada pelo intervalo de tempo necessário para se atingir o tempo mínimo;
(f) *Troca de Linhas Proibida:* quando uma tripulação realiza uma tarefa de uma linha de um determinado grupo e, após o término desta, passa a realizar uma tarefa de uma linha de outro grupo.

A função que representa os requisitos essenciais é a seguinte:

$$f(s) = \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_i B_{ij} \qquad (1)$$

em que
s: solução corrente;
$f(s)$: função requisitos essenciais da solução s;
n: número de tripulações;
m: número de tipos de penalidades;
α_i: peso que reflete a importância do requisito i;
B_{ij}: Valor do requisito i na programação da tripulação j.

ii) **Requisitos não essenciais**:
(a) *Duplas Pegadas*: o algoritmo procura reduzir o número de tripulações que têm jornadas do tipo dupla pegada;
(b) *Horas Extras*: o número de horas que superam 7:10 horas e não ultrapassam 9:10 horas diárias é considerado hora extra. Este montante é minimizado;
(c) *Troca de Pontos Permitida*: uma troca de pontos é permitida quando uma tripulação realiza duas tarefas consecutivas, cujo ponto de troca do final da primeira é diferente do ponto de início da segunda, e o tempo para ir de um para o outro é menor que o intervalo entre elas. Apesar de permitida é desejável que não haja muitas trocas desse tipo;
(d) *Troca de Veículos*: quando uma tripulação realiza uma tarefa em um veículo e, após o término desta, troca de veículo e realiza uma outra tarefa;
(e) *Número de Tripulantes:* o número total de tripulações deve ser minimizado até o ponto que o número de horas extras não seja excessivamente grande;
(f) *Ociosidade:* a soma dos tempos que uma tripulação deixa de trabalhar nas 7:10 horas diárias, descontado o intervalo de 30 minutos para refeição;
(g) *Troca de Linhas Permitida:* quando uma tripulação realiza uma tarefa de uma linha de um determinado grupo e, após o término desta, passa a realizar uma tarefa de uma linha do mesmo grupo.

A função que representa os requisitos não essenciais é a seguinte:

$$g(s) = \sum_{i=1}^{p} \sum_{j=1}^{n} \beta_i C_{ij} + n \times peso + ndp \times peso \quad (2)$$

em que
- s: solução corrente;
- $g(s)$: função requisitos não essenciais da solução s;
- n: número de tripulações;
- ndp: excesso de tripulações com dupla pegada;
- p: número de tipos de custos;
- β_i: peso que reflete a importância do requisito i;
- Cij: Valor do requisito i na programação da tripulação j.

A fórmula representativa da função objetivo total é a seguinte:

$$FO(s) = f(s) + g(s) \quad (3)$$

em que
- s: solução corrente;
- $f(s)$: função requisitos essenciais da solução s;
- $g(s)$: função requisitos não essenciais da solução s;

3.2.2 Estruturas de Vizinhança

Dada uma solução *s*, para atingir uma solução *s'*, onde *s'* é dito vizinho de *s*, são usados dois tipos de movimento (Inserção e Troca) para definir duas estruturas diferentes de vizinhança, a saber: $N^{(1)}(s)$, $N^{(2)}(s)$.

O movimento de inserção (chamado movimento *1-optimal*) consiste em atribuir uma tarefa de uma tripulação *i* a uma tripulação *j*. Na Figura 3.2.2.1 o movimento é explicado detalhadamente. Em (a) temos as duas tripulações antes do movimento formando uma solução *s*. Em (b) ocorre o movimento, nesta figura mostra-se que a tarefa escolhida passa da tripulação *i* para a *j* e que, portanto, as ligações anteriores entre as tarefas deixam de existir (linhas tracejadas da figura). Já em (c) podemos ver as tripulações *i* e *j* constituindo a nova solução *s'* vizinha de *s*.

O conjunto de todos os vizinhos de *s* gerados através de movimentos apenas *1-optimal* define a estrutura de vizinhança $N^{(1)}(s)$.

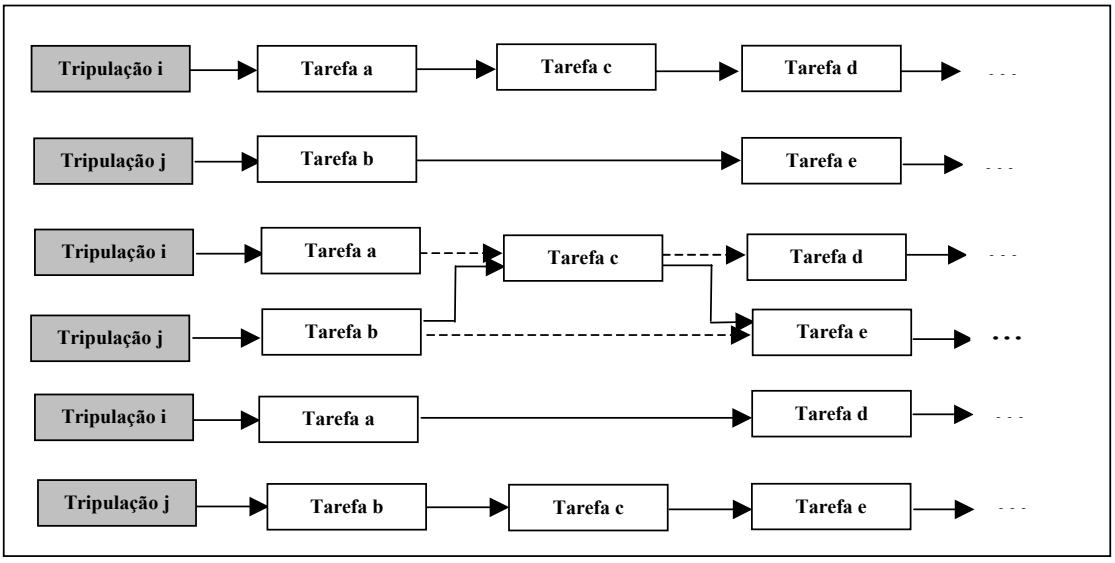


Figura 3.2.2.1: Movimento Inserir Tarefa

Já o movimento de troca considerado (chamado movimento *2-optimal*) consiste em trocar duas tarefas de duas tripulações. Para que a troca seja possível, é necessário que as tarefas a serem trocadas estejam compreendidas numa determinada faixa de tolerância, ou seja, que os seus horários de início e término difiram apenas de uma parcela x determinada, para mais ou para menos. Este tipo de movimento encontra-se ilustrado na Figura 3.2.2.2.

O conjunto de todos os vizinhos de *s* gerados a partir de movimentos do tipo *2-optimal* define a estrutura de vizinhança $N^{(2)}(s)$.

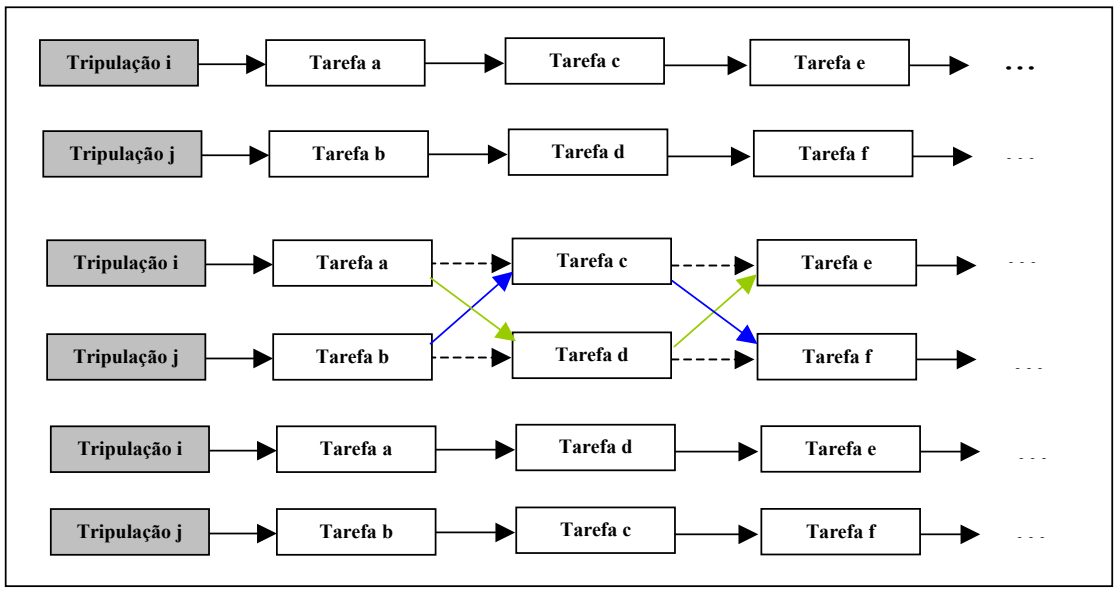


Figura 3.2.2.2: Movimento Trocar Tarefas

3.2.3 Geração de uma solução inicial

Uma solução inicial para o problema da programação da tripulação é gerada por um procedimento construtivo que segue as idéias da fase de construção do método GRASP (Feo & Resende, 1995).

Para a geração da solução inicial, o algoritmo trabalha com uma lista de candidatos restrita (LCR) de tarefas e uma lista de candidatos restrita de tripulações. O parâmetro para controle de aleatoriedade e gulosidade da LCR de tarefas é α e para a LCR de tripulações, β. As tarefas são ordenadas em ordem decrescente de duração (horário de fim da tarefa – horário de início da tarefa). A cada iteração, o algoritmo sorteia, aleatoriamente, um elemento dentro da LCR de tarefas. Para decidir para qual tripulação a tarefa será alocada, o algoritmo faz uma "simulação" da alocação da tarefa para cada um dos tripulantes. Após a "simulação", a lista de tripulações é ordenada em ordem crescente de função objetivo. Esse passo é necessário para a construção da LCR de tripulações. Em seguida, sorteia-se, aleatoriamente um elemento dessa lista, e ele recebe a tarefa sorteada anteriormente.

3.2.4 Funcionamento básico do algoritmo

Passo 1: Gerar as tarefas.

Tomando como base a programação dos veículos, o algoritmo particiona os blocos nas OT's, agrupando as viagens em tarefas.

Passo 2: Obtenção da solução inicial.

As tarefas são distribuídas aos tripulantes segundo a técnica da fase de construção do método GRASP.

Passo 3: Minimizar a função objetivo usando o método VNS.

O método VNS é usado para minimizar a função objetivo, eliminando as inviabilidades operacionais, minimizando os custos da programação e maximizando a utilização da mão-de-obra, ou seja, é usada para tentar atender os requisitos essenciais e não essenciais, fazendo com que o valor da função objetivo seja o menor possível. Como método de busca local para o VNS, foi utilizado o VND que utiliza as estruturas de vizinhança 1-optimal e 2-optimal.

Nos movimento 1-optimal e 2-optimal, para decidir quais as tripulações estarão envolvidas nos movimentos e eliminar o determinismo de começar o movimento pela primeira tripulação, utilizamos uma estrutura de controle que introduziu a aleatoriedade nesse passo.

3.3 Programação Mensal da Tripulação

No problema da Programação Mensal da Tripulação, as jornadas formadas na Programação Diária, terão algumas características próprias de acordo com o tipo do dia que está sendo programado. As jornadas de sábados, domingos e feriados, em menor número, só poderão ser do tipo *Pegada Simples* de acordo com as restrições operacionais da empresa de transporte público em questão.

A Programação Mensal é formada por um conjunto de jornadas, distribuídas ao longo de um determinado mês. Ao se fazer a distribuição das jornadas ao longo de um mês, deve ser levado em conta o tipo do dia que receberá a jornada – dia útil, sábado, domingo e

feriado, sendo que, domingo e feriado são semelhantes em termos de programação de tripulação. Outro fator que deve ser levado em conta são os dias de folga da tripulação.

Para facilitar a abordagem do problema da Programação Mensal da Tripulação, foi usada como estratégia a replicação de jornadas ao longo do mês, ou seja, de acordo com o tipo do dia, as jornadas dentro do mês seriam iguais. Dessa forma, a estrutura usada para a realização da distribuição de jornadas aos tripulantes seria um *agrupamento* de jornadas.

Mostra-se a seguir algumas restrições trabalhistas e operacionais que devem ser contempladas na Programação Mensal e que foram abordadas no presente estudo:

a) O intervalo de tempo entre jornadas que deve ser igual ou superior a 11 horas;

b) O número de trocas de tipos de pegadas entre jornadas, que deve ser reduzido;

c) Deve haver uma uniformidade na construção da escala mensal das tripulações de forma que exista um baixo desvio das horas trabalhadas no mês, por uma tripulação específica, em relação à média das horas totais trabalhadas pelas tripulações.

3.4 VNS aplicado ao Problema da Programação Mensal da Tripulação

3.4.1 A Função Objetivo

A função a ser minimizada, que avalia a programação mensal da tripulação, tem os seguintes requisitos:

i)**Requisito essencial**:

(a) *Tempo entre Jornadas:* o tempo mínimo entre jornadas de trabalho é de 11:00. O tempo entre jornadas que não for igual ou superior a esse valor será penalizado pelo intervalo de tempo necessário para se atingir o tempo mínimo;

Os requisitos essenciais são avaliados com base na seguinte fórmula:

$$f(s) = \sum_{j=1}^{n} \alpha B_j \qquad (1)$$

em que
- s: solução corrente;
- $f(s)$: função que avalia o não atendimento ao requisito essencial da solução s;
- n: número de tripulações;
- α: peso que reflete a importância desse requisito;
- B_j: Quantificação desse requisito na programação da tripulação j.

ii) **Requisitos não essenciais**:

(a) *Diferença do Tempo Médio de Trabalho*: o total de horas trabalhadas por uma tripulação ao longo do mês deve diferir tão pouco, quanto possível, da média de horas trabalhadas por todas as tripulações. Ocorrerá a penalização desse quesito se a diferença em relação à média ultrapassar um valor máximo estipulado pelo usuário. Esse valor será penalizado pelo intervalo de tempo para se atingir o valor máximo estipulado;

(b) *Número de Jornadas Diferentes*: uma jornada é considerada diferente da outra quando o conjunto de tarefas que as compõem for diferente. Este montante deve ser minimizado;

(c) *Troca de Períodos de Trabalho*: ocorre uma troca de período de trabalho quando a tripulação realiza duas jornadas consecutivas, onde a primeira começa em um determinado período do dia (por exemplo, manhã) e a outra em outro período do

dia (por exemplo, noite). É desejável que não haja muitas trocas desse tipo. A definição da hora do dia em que ocorre a mudança do período de trabalho é definido pelo usuário. Para efeito dos testes consideramos que o período da tarde se inicia quando a primeira tarefa de uma jornada começa a partir das 8 horas da manhã.

(d) *Troca de Tipos de Pegada:* quando uma tripulação realiza um conjunto de jornadas em dias úteis com um determinado tipo de pegada e, em outros dias úteis, passa a realizar um conjunto de jornadas com um outro tipo de pegada.

Os requisitos não essenciais são avaliados com base na seguinte fórmula:

$$g(s) = \sum_{i=1}^{p} \sum_{j=1}^{n} \beta_i C_{ij} \qquad (2)$$

em que
- s: solução corrente;
- $g(s)$: função que avalia o não atendimento aos requisitos não essenciais da solução s;
- p: número de tipos de custos;
- β_i: peso que reflete a importância do requisito i;
- C_{ij}: Medida do requisito i na programação da tripulação j.

A fórmula representativa da função objetivo total é a seguinte:

$$FO(s) = f(s) + g(s) \qquad (3)$$

3.4.2 Estruturas de Vizinhança

Para simplificar o tratamento do problema da Programação Mensal, as jornadas de dias úteis foram agrupadas por semana (vide Fig. 3.4.2.1). Por exemplo, se um mês, sem feriados, começa em uma quarta-feira, então, os dias de quarta, quinta e sexta pertencerão a um mesmo agrupamento, o qual receberá uma mesma jornada. Nesse exemplo o próximo agrupamento envolveria os dias compreendidos entre a segunda e a sexta-feira da semana subseqüente.

Assim, dada uma solução s, para atingir uma solução s', onde s' é dito vizinho de s, é usado o movimento de troca para definir três estruturas diferentes de vizinhança, $N^{(1)}(s)$, $N^{(2)}(s)$ e $N^{(3)}(s)$. A primeira estrutura utiliza movimentos de troca entre agrupamentos de dias úteis. A segunda, $N^{(2)}(s)$, faz uso de movimentos entre jornadas de sábados. Finalmente, a terceira estrutura, $N^{(3)}(s)$, trabalha com movimentos entre jornadas de domingos ou agrupamentos de domingos/feriados.

O movimento de troca considerado (chamado movimento *2-optimal*) consiste em trocar jornadas (ou agrupamentos) de duas tripulações. A troca só e possível se os dias das jornadas (ou agrupamentos) das tripulações envolvidas no movimento forem iguais e não haja folga nesse dia (ou conjunto de dias). Este tipo de movimento encontra-se ilustrado na Figura 3.4.2.2.

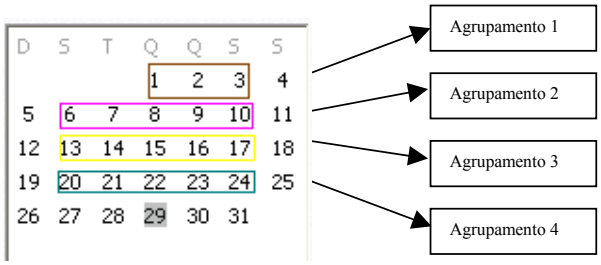


Figura 3.4.2.1: Agrupamentos de jornadas de dias úteis por semana.

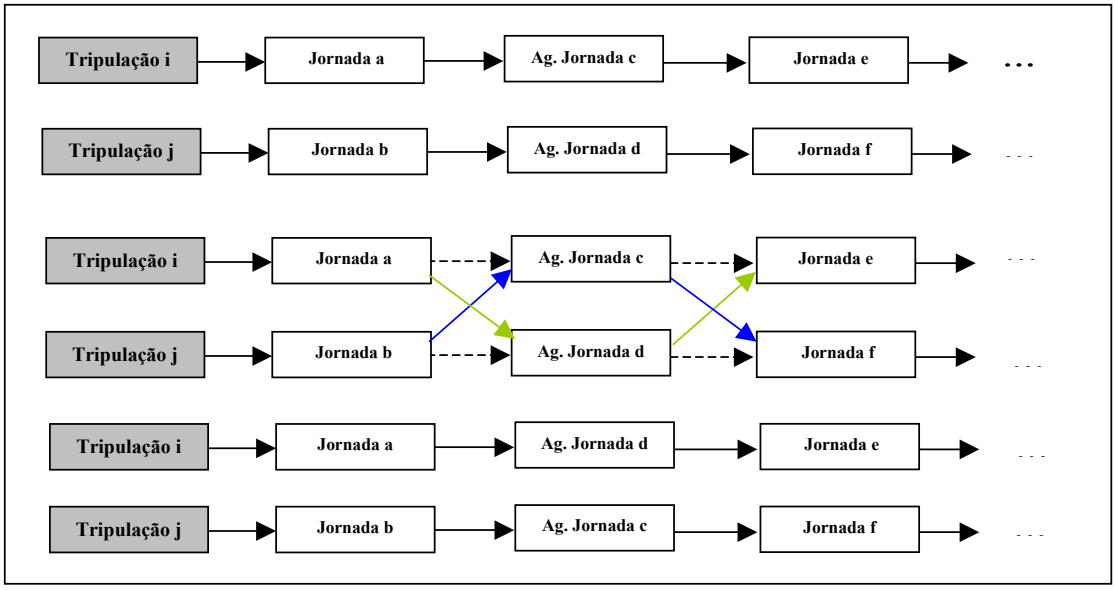


Figura 3.4.2.2: Movimento Trocar Jornadas (ou Agrupamentos).

3.4.3 Geração de uma solução inicial

A distribuição de jornadas aos tripulantes é feita aleatoriamente, mas com replicação das mesmas ao longo do mês, de acordo com o tipo do dia em questão (dia útil, sábado e domingo/feriado).

3.4.4 Funcionamento básico do algoritmo

Passo 1: Gerar estrutura do mês.

O algoritmo gera uma estrutura do mês a ser programado, com a identificação dos tipos de dias de trabalho. Os dias em seqüência de mesmo tipo (útil e domingo/feriado) são reunidos em agrupamentos.

Passo 2: Gerar o conjunto de jornadas.

Tomando por base as tarefas da Programação Diária (uma para dias úteis, uma para sábados e uma para domingos/feriados), o algoritmo agrupa as tarefas em jornadas.

Passo 3: Obtenção da solução inicial.

A solução inicial é obtida conforme seção 3.4.3.

Passo 4: Minimizar a função objetivo usando o método VNS.

O método VNS é usado para minimizar a função objetivo, eliminando as inviabilidades operacionais, minimizando os custos da programação e procurando equilibrar a quantidade de trabalho das tripulações, ou seja, é usada para tentar atender os requisitos essenciais e não essenciais, fazendo com que o valor da função objetivo seja o menor

possível. Como método de busca local para o VNS foi utilizado o VND, o qual utiliza as estruturas de vizinhança definidas na seção 3.4.2.

Em vista do elevado custo computacional para fazer uma avaliação completa de uma vizinhança, apenas um percentual da mesma é analisada. Dessa forma, para eliminar o determinismo de sempre começar pela primeira tripulação, utilizamos, tal como na programação diária, uma estrutura de controle que introduziu a aleatoriedade nesse passo.

4 Sistema Desenvolvido

4.1 Descrição do sistema

Para a implementação do sistema foi utilizada a linguagem C++ e o ambiente para desenvolvimento Borland C++ Builder 5.0 ®, versão Professional, rodando na plataforma Windows ®. A escolha dessa linguagem é justificada pelo seu desempenho altamente satisfatório.

O sistema foi totalmente desenvolvido segundo o paradigma de Orientação a Objetos, que favorece vários fatores, como por exemplo, a reusabilidade de código e alta modularização. Ele permite que o usuário faça alterações de valores que mudam freqüentemente como, por exemplo, os valores determinados pelas Convenções Coletivas de Trabalho, que mudam, praticamente, todo ano, apresentando assim uma boa flexibilidade de configuração.

O sistema consiste de dois módulos independentes que são: o módulo de Programação Diária e o módulo de Programação Mensal:

O funcionamento do módulo de Programação Diária pode ser melhor descrito em cinco etapas:

1) *Parâmetros da programação da tripulação*: O usuário preenche os parâmetros referentes às tripulações como, por exemplo, o tempo para troca das mesmas, a duração do intervalo de repouso/alimentação etc;

2) *Programação de veículos:* Nessa etapa, o usuário seleciona um arquivo que contém a programação de veículos;

3) *Entrada do número de tripulações*: O número de tripulações com o qual o algoritmo irá trabalhar é fornecido pelo usuário;

4) *Seleção do tipo da escala diária:* O usuário seleciona o tipo da escala diária que será programado: escala para dias úteis, escala para sábados e escala para domingos/feriados;

5) *Execução do VNS*: Nessa etapa o usuário executa o *VNS*.

Obs: Em qualquer momento antes da etapa 5, o usuário ainda pode alterar os valores dos parâmetros do GRASP – VNS/VND e das penalizações.

O funcionamento do módulo de Programação Mensal pode ser melhor descrito em três etapas:

1) *Parâmetros da programação da tripulação*: O usuário preenche os parâmetros referentes às tripulações: a diferença máxima do tempo médio trabalhado e a hora limite para troca de período de trabalho;

2) *Estrutura do mês:* Nessa etapa, o usuário identifica os feriados do mês a ser programado. Sábados e domingos são identificados automaticamente pelo sistema.

3) *Execução do VNS*: Nessa etapa o usuário executa o *VNS*.

Obs: Em qualquer momento antes da etapa 3, o usuário ainda pode alterar os valores dos parâmetros do VNS/VND e das penalizações.

4.2 Entradas do usuário e configuração de parâmetros

O sistema permite ao usuário, entrar com o número de tripulações, alterar os valores da programação da tripulação, das penalizações, do GRASP-VNS/VND etc. Isso é feito através das telas representadas pelas figuras a seguir:

4.2.1 Entrada do número de tripulações

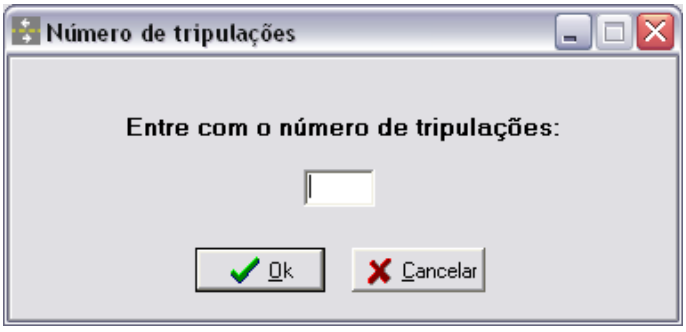


Figura 4.2.1: Entrada do número de tripulações

Clicando no botão "OK" da tela representada pela Figura 4.2.1, o número de tripulações será inserido no sistema.

4.2.2 Parâmetros da programação da tripulação

→Programação Diária

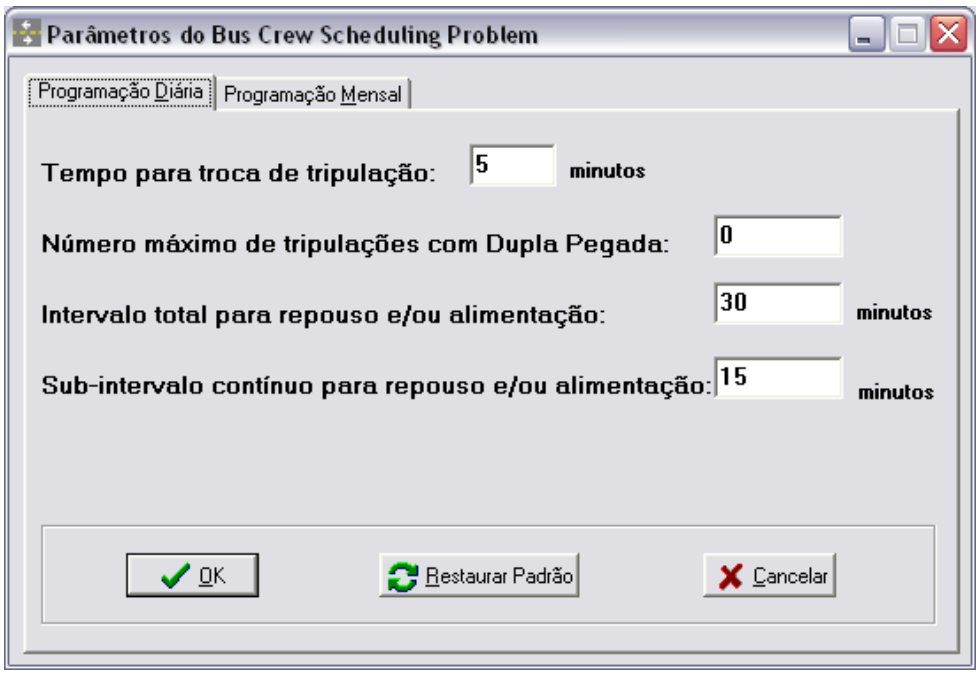


Figura 4.2.2: Parâmetros da Programação Diária da Tripulação

(a) *Tempo para troca de tripulação*: tempo necessário para realizar uma troca de tripulação. Esse valor é um número inteiro em minutos;

(b) *Número máximo de tripulações com Dupla Pegada*: número máximo de tripulações com dupla pegada. Esse valor é um número inteiro.

(c) *Intervalo total para repouso e/ou alimentação*: tempo mínimo que a tripulação deve ter para repouso e/ou alimentação. Esse valor é um número inteiro em minutos;

(d) *Sub*-intervalo contínuo para repouso e/ou alimentação: tempo contínuo mínimo que a tripulação deve ter para repouso e/ou alimentação. Esse valor é um número inteiro em minutos;

Clicando no botão "OK" da tela representada pela Figura 4.2.2, os parâmetros para o problema da Programação Diária serão alterados para os valores apresentados na tela.

Clicando no botão "Restaurar padrão" da tela representada pela Figura 4.2.2., os campos serão preenchidos com os valores padrão.

Clicando no botão "Cancelar" da tela representada pela Figura 4.2.2, as alterações realizadas pelo usuário não terão efeito sobre os parâmetros do problema da Programação Diária, e a tela será fechada.

→Programação Mensal

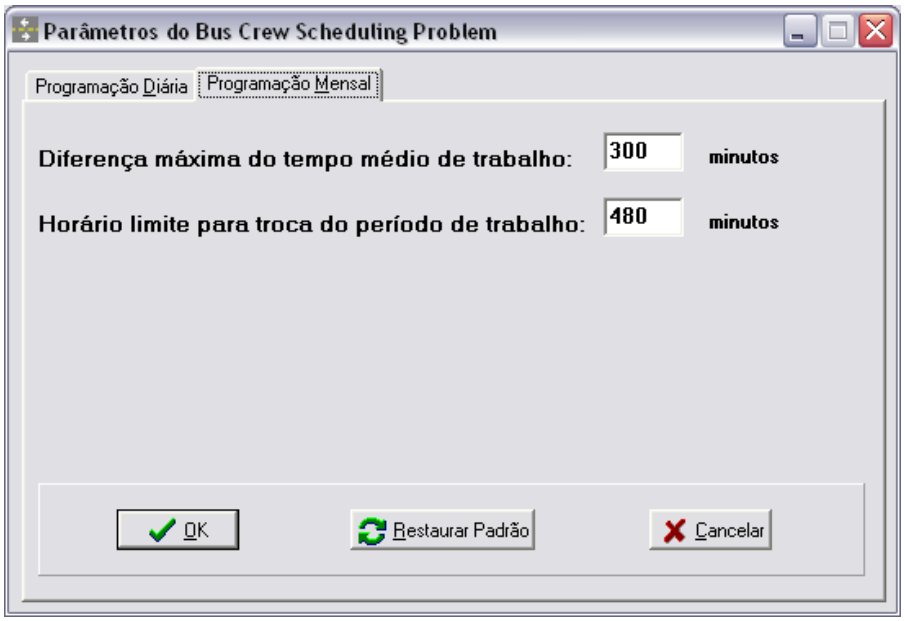


Figura 4.2.3: Parâmetros da Programação Mensal da Tripulação

(a) *Diferença máxima do tempo médio de trabalho*: variação máxima desejada da média de horas trabalhadas de uma tripulação em relação à média de horas trabalhadas de todas tripulações. Esse valor é um número inteiro em minutos;

(b) *Horário limite para troca do período de trabalho*: horário que caracteriza a separação entre turnos de trabalho. Esse valor é um número inteiro em minutos.

Clicando no botão "OK" da tela representada pela Figura 4.2.3, os parâmetros para o problema da Programação Mensal serão alterados para os valores apresentados na tela.

Clicando no botão "Restaurar padrão" da tela representada pela Figura 4.2.3., os campos serão preenchidos com os valores padrão.

Clicando no botão "Cancelar" da tela representada pela Figura 4.2.3, as alterações realizadas pelo usuário não terão efeito sobre os parâmetros do problema da Programação Mensal, e a tela será fechada.

4.2.3 Penalizações

→ Programação Diária

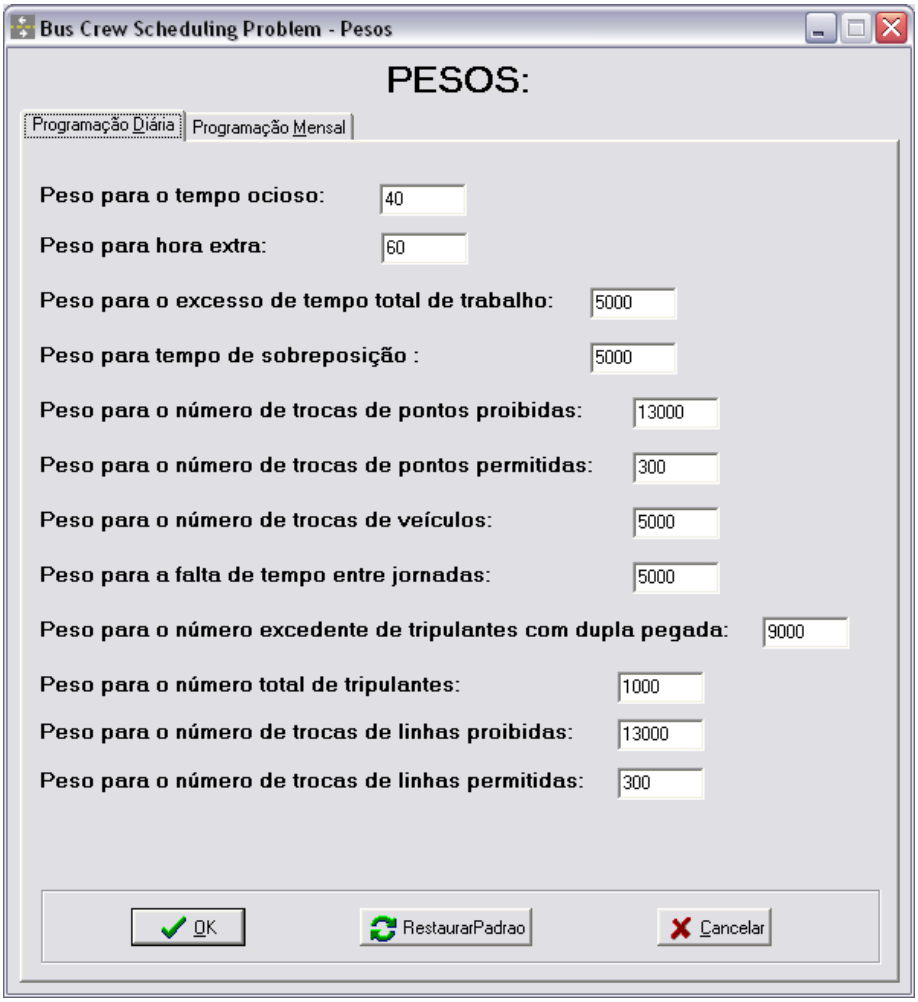


Figura 4.2.4: Pesos para a Programação Diária

(a) Peso para o tempo ocioso: valor utilizado para penalizar o tempo ocioso das tripulações;

(b) Peso para hora extra: valor utilizado para penalizar as horas extras das tripulações;

(c) Peso para o excesso de tempo total de trabalho: valor utilizado para penalizar o excesso no tempo total de trabalho das tripulações;

(d) Peso para o tempo de sobreposição: valor utilizado para penalizar o tempo de sobreposição das tripulações;

(e) Peso para o número de trocas de pontos proibidas: valor utilizado para penalizar o número de trocas de ponto proibidas das tripulações;

(f) Peso para o número de trocas de pontos permitidas: valor utilizado para penalizar o número de trocas de ponto permitidas das tripulações;

(g) Peso para o número de trocas de veículos: valor utilizado para penalizar o número de trocas de veículos das tripulações;

(h) Peso para a falta de tempo entre jornadas: valor utilizado para penalizar a falta de tempo entre as jornadas das tripulações;

(i) Peso para o número excedente de tripulações com dupla pegada: valor utilizado para penalizar o excesso de tripulações com dupla pegada;

(j) Peso para o número de tripulações: valor utilizado para penalizar o número de tripulações;

(k) Peso para o número de trocas de linhas proibidas: valor utilizado para penalizar o número de trocas de linhas proibidas das tripulações;

(l) Peso para o número de trocas de linhas permitidas: valor utilizado para penalizar o número de trocas de linhas permitidas das tripulações.

Clicando no botão "OK" da tela representada pela Figura 4.2.4, os valores para as penalizações da Programação Diária serão alterados para os valores apresentados na tela.

Clicando no botão "Restaurar padrão" da tela representada pela Figura 4.2.4, os campos serão preenchidos com os valores padrão.

Clicando no botão "Cancelar" da tela representada pela Figura 4.2.4, as alterações realizadas pelo usuário não terão efeito sobre as penalizações da Programação Diária, e a tela será fechada.

→ Programação Mensal

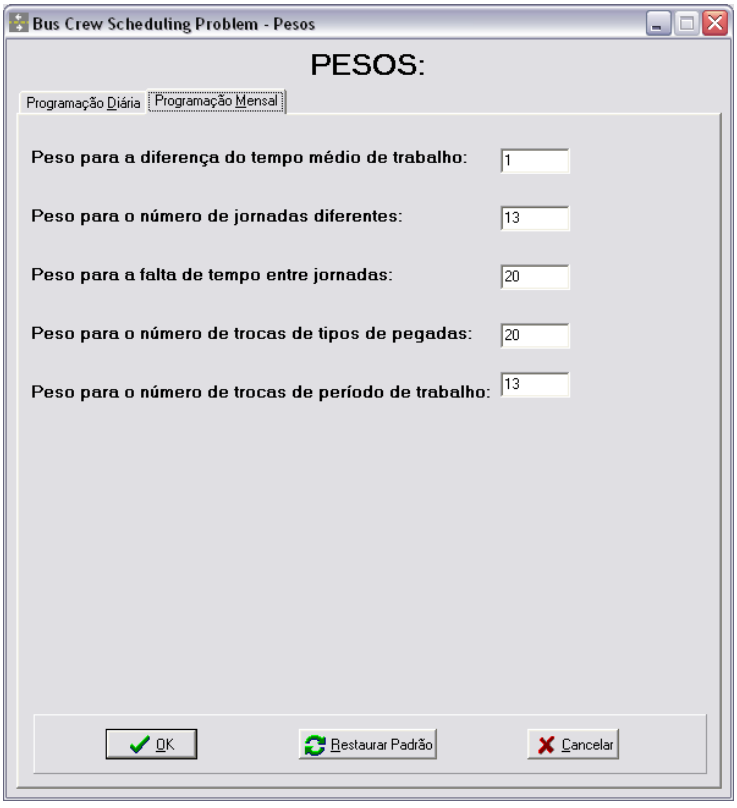


Figura 4.2.5: Pesos para a Programação Mensal

(a) Peso para a diferença do tempo médio de trabalho: valor utilizado para penalizar a diferença do tempo médio de trabalho de uma tripulação em relação à média de horas trabalhadas por todas tripulações;

(b) Peso para o número de jornadas diferentes: valor utilizado para penalizar o número de jornadas diferentes das tripulações;

(c) Peso para a falta de tempo entre jornadas: valor utilizado para penalizar a falta de tempo entre as jornadas das tripulações;

(d) Peso para o número de trocas de tipos de pegadas: valor utilizado para penalizar o número de trocas de tipos de pegadas das tripulações;

(e) Peso número de trocas de período de trabalho: valor utilizado para penalizar o número de trocas de período de trabalho das tripulações.

Clicando no botão "OK" da tela representada pela Figura 4.2.5, os valores para as penalizações da Programação Mensal serão alterados para os valores apresentados na tela.

Clicando no botão "Restaurar padrão" da tela representada pela Figura 4.2.5, os campos serão preenchidos com os valores padrão.

Clicando no botão "Cancelar" da tela representada pela Figura 4.2.5, as alterações realizadas pelo usuário não terão efeito sobre as penalizações da Programação Mensal, e a tela será fechada.

4.2.4 Parâmetros do GRASP – VNS/VND

→Programação Diária

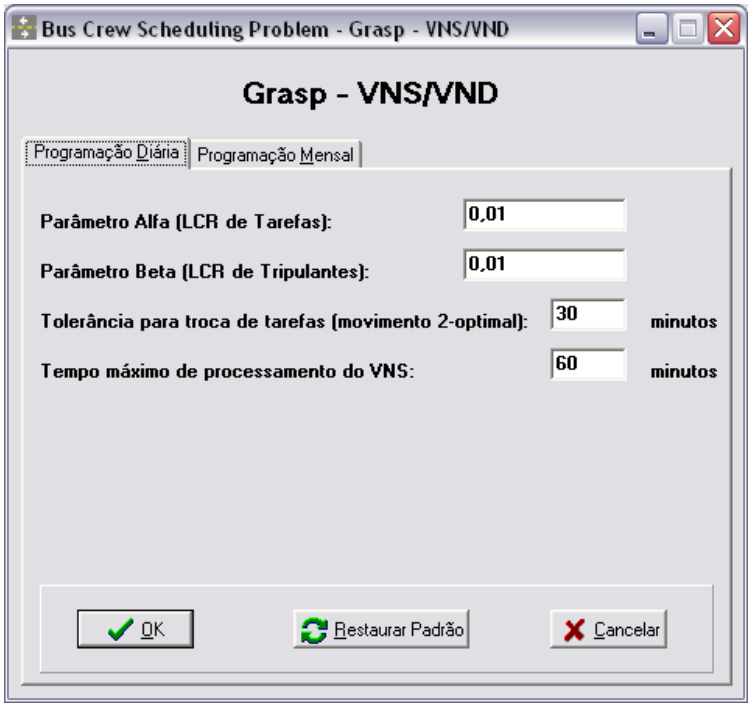


Figura 4.2.6: Parâmetros do GRASP-VNS/VND para a Programação Diária

(a) *Parâmetro Alfa (LCR de Tarefas)*: parâmetro que regula o tamanho da Lista de Candidatos Restrita de Tarefas;

(b) *Parâmetro Beta (LCR de Tripulações)*: parâmetro que regula o tamanho da Lista de Candidatos Restrita de Tripulações;

(c) *Tolerância para troca de tarefas (movimento 2-optimal)*: faixa de tolerância para que aconteça a troca de tarefas entre duas tripulações

(d) *Tempo máximo de processamento do VNS:* tempo durante o qual o algoritmo VNS irá rodar.

Clicando no botão "OK" da tela representada pela Figura 4.2.6, os parâmetros dos métodos GRASP e VNS/VND serão alterados para os valores apresentados na tela.

Clicando no botão "Restaurar padrão" da tela representada pela Figura 4.2.6., os campos serão preenchidos com os valores padrão.

Clicando no botão "Cancelar" da tela representada pela Figura 4.2.6, as alterações realizadas pelo usuário não terão efeito sobre os parâmetros dos métodos GRASP e VNS/VND, e a tela será fechada.

→Programação Mensal

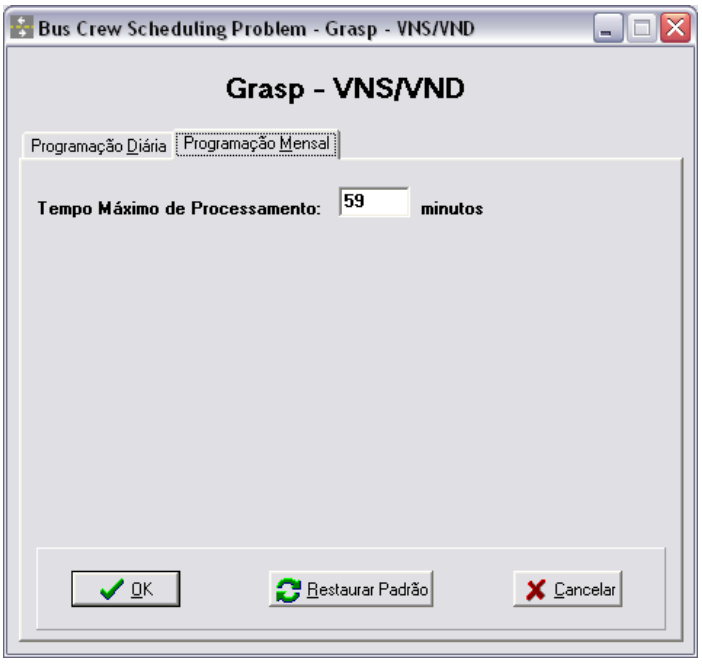


Figura 4.2.7: Parâmetros do VNS/VND para a Programação Mensal

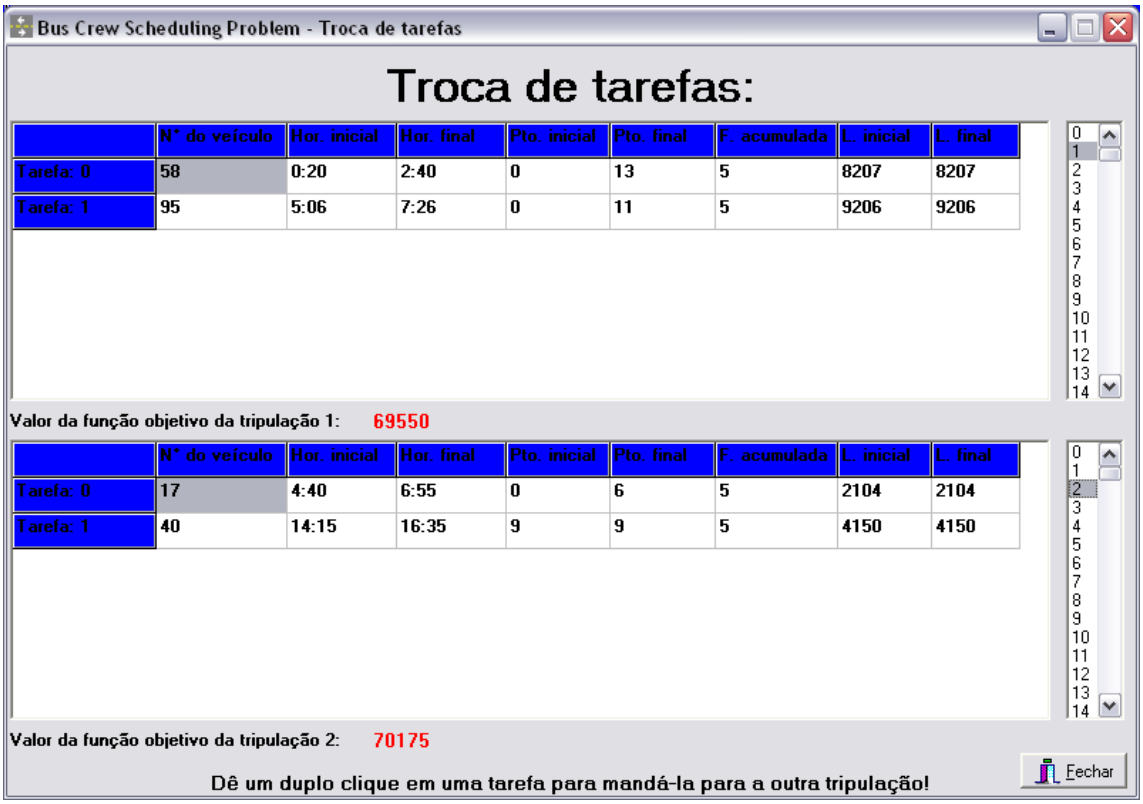
(a) *Tempo máximo de processamento do VNS:* tempo durante o qual o algoritmo VNS irá rodar.

Clicando no botão "OK" da tela representada pela Figura 4.2.7, os parâmetros do método VNS/VND serão alterados para os valores apresentados na tela.

Clicando no botão “Restaurar padrão” da tela representada pela Figura 4.2.7., os campos serão preenchidos com os valores padrão.

Clicando no botão “Cancelar” da tela representada pela Figura 4.2.7, as alterações realizadas pelo usuário não terão efeito sobre os parâmetros do método VNS/VND, e a tela será fechada.

4.2.5 Troca manual de Tarefas



	Nº do veículo	Hor. inicial	Hor. final	Pto. inicial	Pto. final	F. acumulada	L. inicial	L. final
Tarefa: 0	58	0:20	2:40	0	13	5	8207	8207
Tarefa: 1	95	5:06	7:26	0	11	5	9206	9206

	Nº do veículo	Hor. inicial	Hor. final	Pto. inicial	Pto. final	F. acumulada	L. inicial	L. final
Tarefa: 0	17	4:40	6:55	0	6	5	2104	2104
Tarefa: 1	40	14:15	16:35	9	9	5	4150	4150

Figura 4.2.8: Troca manual de tarefas

O sistema permite ao usuário trocar tarefas (manualmente) entre tripulações. Para isso, na Figura 4.2.8, basta selecionar duas tripulações (nas listas à direita das tabelas), depois dar um duplo clique em alguma tarefa de uma tabela e ela será enviada para a outra tabela, ou seja, para outra tripulação.

4.2.6 Tipo de escala para a Programação Diária

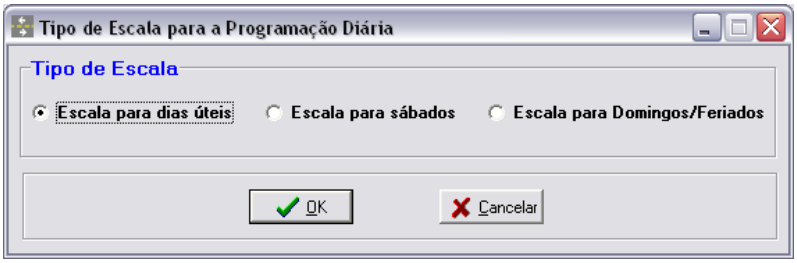


Figura 4.2.9: Tipo de escala para a Programação Diária

O usuário escolhe um tipo de escala para a Programação Diária e clica no botão “OK” para confirmar a sua opção.

Clicando no botão “Cancelar” da tela representada pela Figura 4.2.9 a tela será fechada.

5 Resultados obtidos

5.1 Programação Diária da Tripulação

Para testar o algoritmo, consideramos os dados relativos a um mês do ano de 2002 de uma empresa do sistema de transporte público do município de Belo Horizonte. O sistema foi executado em um PC Athlon 850MHz, com 128MB de RAM, sob o sistema operacional Windows 2000.

A tabela 5.1.1 mostra o valor da solução final obtido em 5 execuções do sistema a essa massa de dados. Cada execução partiu de uma semente diferente de números aleatórios. O tempo de CPU de cada teste foi de 75 minutos.

	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5
Função Objetivo	1.034.280	1.016.540	1.045.340	1.089.060	1.040.780

Tabela 5.1.1: Valores das soluções finais.

A tabela 5.1.2 reúne as informações relativas à melhor solução obtida (Teste 2).

	Valor
Número de tripulações	219
Número de tarefas	688
Tempo total trabalhado	1584:19
Tempo que falta entre as jornadas	0:00
Tempo total de sobreposição	0:00
Horas excedentes	0:00
Hora extra total	97:57
Tempo ocioso total	138:38
Nº de tripulações com dupla pegada	20
N° de trocas de veículos	21
N° de trocas de ponto proibidas	0
N° de trocas de ponto permitidas	14
N° de trocas de linha proibidas	0
N° de trocas de linha permitidas	10
Função objetivo total	1.016.540

Tabela 5.1.2: Informações da melhor solução.

A tabela 5.1.3 mostra uma análise da média dos valores das funções objetivo obtidos em cada teste, bem como o desvio apresentado em relação à melhor solução conhecida.

Melhor Função Objetivo Conhecida (*)	1.010.880
Média das Funções Objetivo	1.045.200
Desvio (%)	3,4

Tabela 5.1.3: Análise de desempenho do método heurístico.

$$\textbf{Desvio} = \left(\frac{FOMedia - FOMelhor}{FOMelhor} \right) \times 100$$

* Relatório técnico DECOM 02/2003, de autoria de Geraldo Regis Mauri, pela aplicação da técnica *Simulated Annealing* a esse problema.

Parâmetros e pesos adotados nos testes

Parâmetro do PPT	**Valor**
Número de tripulações	219
Tempo de troca de tripulação	5 minutos
Intervalo total para repouso e/ou alimentação	30 minutos
Sub-intervalo contínuo para repouso e/ou alimentação	15 minutos
Número máximo de tripulações com dupla pegada	20

Tabela 5.1.4: Parâmetros do PPT.

Peso	**Valor**
Para a falta de tempo entre jornadas	5000
Para o número de tripulações	1000
Para o tempo ocioso	40
Para o tempo total de trabalho	5000
Para as horas extras	60
Para o tempo de sobreposição	5000
Para o número de trocas de ponto proibidas	13000
Para o número de trocas de ponto permitidas	300
Para o número de trocas de linha proibidas	13000
Para o número de trocas de linha permitidas	300
Para o número de trocas de veículos	5000
Para o número máximo de tripulações com dupla pegada	9000

Tabela 5.1.5: Pesos.

5.2 Programação Mensal da Tripulação

A tabela 5.2.1 mostra o valor da solução final obtido em 5 execuções do sistema a uma massa de dados proveniente da Programação Diária. Os dados relativos aos dias úteis foram obtidos a partir de dados reais da empresa considerada para análise. Já os demais dados relativos a sábados, domingos e feriados foram extraídos de uma programação de veículos hipotética, uma vez que não tínhamos essas informações disponíveis. Cada execução partiu de uma semente diferente de números aleatórios. O tempo de CPU de cada teste foi de 60 minutos.

	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5
Função Objetivo	122.631	122.873	118.728	121.046	122.551

Tabela 5.2.1: Valores das soluções finais.

A tabela 5.2.2 reúne as informações relativas à melhor solução obtida (Teste 3).

	Valor
Número de tripulações	218
Número de jornadas	5886
Tempo total trabalhado	42428:40
Tempo médio de trabalho	194:37
Nº de trocas de tipo de pegada	8
Falta de tempo entre jornadas	0:00
Nº de trocas de período de trabalho	88
Diferença máxima do tempo médio de trabalho	86:25
Função objetivo total	118.728

Tabela 5.2.2: Informações da melhor solução.

A tabela 5.2.3 mostra uma análise da média dos valores das funções objetivo obtidos em cada teste, bem como o desvio apresentado em relação à melhor solução conhecida.

Melhor Função Objetivo Conhecida (*)	14.541
Média das Funções Objetivo	121.565
Desvio (%)	736%

Tabela 5.2.3: Análise de desempenho do método heurístico.

$$\textbf{Desvio} = \left(\frac{FOMedia - FOMelhor}{FOMelhor} \right) \times 100$$

* Relatório técnico 02/2003 de autoria de Geraldo Regis Mauri, pela aplicação da técnica *Simulated Annealing* a esse problema.

Parâmetros e penalizações adotadas nos testes

Parâmetro do PPT	**Valor**
Diferença máxima do tempo médio trabalhado	300 minutos (5:00)
Horário limite para troca de período de trabalho	480 minutos (8:00)

Tabela 5.2.4: Parâmetros do PPT.

Peso	**Valor**
Para a diferença do tempo médio trabalhado	1
Para o número de jornadas diferentes	13
Para a falta de tempo entre jornadas	20
Para o número de trocas de tipo de pegadas	20
Para o número de trocas de período de trabalho	13

Tabela 5.2.5: Pesos.

6 Conclusões e Propostas de trabalhos futuros

6.1 Análise dos resultados e conclusões

6.1.1 Programação Diária

A tabela 6.1.1.1 resume as informações relativas à melhor solução encontrada pelo algoritmo VNS (vide seção 5.1), bem como aquelas adotadas pela empresa investigada. Podemos verificar que a melhor solução encontrada pelo algoritmo apresenta uma diminuição significativa da ociosidade e de horas extras em relação à adotada na programação de dias úteis da empresa. O algoritmo conseguiu, ainda, eliminar completamente o número de horas excedentes à jornada de trabalho.

Em todos os testes o algoritmo conseguiu atender os requisitos essenciais, ou seja, obteve, em cada caso, soluções viáveis.

Pela análise da tabela 5.1.3 verificamos que o método híbrido GRASP-VNS se mostrou robusto, pois partindo-se de diferentes soluções iniciais chega-se a soluções finais que diferem da melhor solução conhecida de apenas 3,4%.

Portanto, os resultados obtidos validam a utilização do Método de Pesquisa em Vizinhança Variável (VNS) para a resolução do Problema da Programação Diária da Tripulação.

Requisitos	Empresa	Melhor solução
Horas extras	116:00	97:57
Horas excedentes	12:06	0:00
Ociosidade	188:22	138:38
Duplas Pegadas	21	20
Número de tripulações	219	219

Tabela 6.1.1.1: Comparação dos valores obtidos com os dados reais.

6.1.2 Programação Mensal

Em todos os testes o algoritmo conseguiu atender os requisitos essenciais, ou seja, obteve, em cada caso, soluções viáveis. Entretanto, analisando a tabela 6.1.3 verificamos que o método VNS não teve um desempenho satisfatório comparado com o algoritmo *Simulated Annealing*, devido ao alto desvio apresentado pelas soluções finais obtidas.

Esse desempenho insatisfatório deveu-se, em parte, à baixa qualidade das soluções iniciais, as quais foram geradas aleatoriamente. Essa performance pode ser melhorada utilizando-se uma fase de construção GRASP para a solução inicial e pesquisando-se outras estruturas de vizinhança.

6.2 Propostas de trabalhos futuros

- Implementar novas estruturas de vizinhança para a Programação Diária, como por exemplo, com movimentos 1 ou 2-optimais envolvendo tarefas de um mesmo turno de trabalho.

- Implementar outros requisitos que não foram abordados tanto na Programação Diária, quanto na Mensal, como por exemplo, a fixação de uma tarefa a um determinado tripulante.
- Considerar dinâmico o percentual da vizinhança a ser analisada a cada iteração do métodoVND, isto é, no início do processo explorar uma vizinhança mais restrita e gradativamente, aumentar esse percentual à medida que não houver soluções de melhora.
- Desenvolver técnicas híbridas envolvendo o método VNS, isto é, associar o método VNS à uma outra metaheurística, como por exemplo, como fase de refinamento de uma solução advinda da aplicação do método *Simulated Annealing*. Dessa forma, conjugaremos os aspectos mais positivos de cada metaheurística com o objetivo de encontrar soluções finais de melhor qualidade.

7 Referências Bibliográficas

(a) BOUZADA, C. F (2002) Análise das despesas administrativas no custo do transporte coletivo por ônibus no município de Belo Horizonte. Dissertação de mestrado, Escola de Governo, Fundação João Pinheiro, Belo Horizonte.

(b) CLEMENT E WREN (1995) Greedy genetic algorithms, optimizing mutantis and bus driver scheduling. In: Daduna, J. R.; Branco, I.; Paixão, J. M. P. (eds.), p. 213-235.

(c) DADUNA, J. R.; BRANCO, I.; PAIXÃO, J. M. P. (eds.) (1995) Computer-Aided Transit Scheduling. Springer-Verlag, Berlin.

(d) DESROCHERS, M.; ROUSSEAU, J. M. (eds.) (1992) Computer-Aided Transit Scheduling. Springer-Verlag, Berlin.

(e) ELIAS, S. E. G. (1964) The use of digital computers in the economic scheduling for both man and machine in public transport. Technical Report 49, Kansas State University Bulletin, Kansas, EUA

(f) FEO, T.A. AND RESENDE, M.G.C. (1995) "Greedy randomized adaptive search procedures", *Journal of Global Optimization*, 6:109-133.

(g) GLOVER, F. (1989) Tabu search: Part I. ORSA Journal on Computing, 1, p. 190-206.

(h) GLOVER, F. (1990) Tabu search: Part II. ORSA Journal on Computing, 2, p. 04-32.

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

(j) KIRKPATRICK, S.; D. C. GELLAT E M. P. VECCHI. (1983) Optimization by Simulated Annealing. Science, p. 671–680.

(k) KWAN, A. S. K.; KWAN, R. K.; WREN, A. (1999) Driver scheduling using genetic algorithms with embedded combinatorial traits. In: Wilson, N. H. M. (ed.), Springer-Verlag, Berlin, p. 81-102.

(l) M. PRAIS AND C.C. RIBEIRO. Parameter Variation in GRASP Implementations. In *Proceedings of the Third Metaheuristics International Conference,* pages 375-380, Angra dos Reis, Brazil, 1999

(m) M. PRAIS AND C.C. RIBEIRO. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing, 1998.*

(n) M. PRAIS AND C.C. RIBEIRO. Variação de Parâmetros em Procedimentos GRASP. *Investigación Operativa*, 1999.

(o) MANINGTON, P. D.; WREN, A. (1975) Experiences with a bus scheduling algorithm which saves vehicles. In: Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services, Chicago.

(p) MLADENOVIC, N. AND HANSEN, P. (1997) "A Variable Neighborhood Search", *Computers and Operations Research*, 24: 1097-1100.

(q) MLADENOVIC, N. AND HANSEN, P. (1999) "Variable Neighborhood Search: Methods and Recent Applications", *In* Third Metaheuristics International Conference, Angra dos Reis, Brazil, pp.275-280.

(r) SHEN, Y.; KWAN, R. S. K. (2001) Tabu Search for driver scheduling. Berlin, In Voß, S; Daduna, J. (eds.) p. 121-135.

(s) VOß, S; DADUNA, J. (eds.) (2001) Computer-Aided Scheduling of Public Transport, Springer-Verlag, Berlin.

(t) WILSON, N. H. M. (ed.) (1999) Computer-Aided Transit Scheduling, Springer-Verlag, Berlin.

(u) WREN, A.; ROUSSEAU, J. M. (1995) Bus driver scheduling – An overview. In Daduna, J. R.; Branco, I.; Paixão, J. M. P. (eds.), p. 173-187.

(v) WREN, A.; WREN, D. O. (1995) A genetic algorithm for public transport driver scheduling. *Computer and Operations Research* 22, v.1, p.101-110.

Anexo A. Código Fonte do Sistema

Anexo B. Saída do Sistema