UNIVERSIDADE FEDERAL DO CEARÁ

PROGRAMA DE MESTRADO EM ENGENHARIA DE TRANSPORTES

UTILIZAÇÃO DA METAHEURÍSTICA *SIMULATED ANNEALING* NO PROBLEMA DE ALOCAÇÃO DE PESSOAL EM EMPRESAS DE TRANSPORTE COLETIVO POR ÔNIBUS

Heider Augusto da Silva Gomes

Dissertação submetida ao Programa de Mestrado em Engenharia de Transportes da Universidade Federal do Ceará, como parte dos requisitos para a obtenção do título de Mestre em Ciências (M.Sc.) em Engenharia de Transportes.

ORIENTADOR: Prof. Dr. Júlio Francisco Barros Neto

Fortaleza
2003

FICHA CATALOGRÁFICA

GOMES, HEIDER AGUSTO DA SILVA

Utilização da Metaheurística *Simulated Annealing* no Problema de Alocação de Pessoal em Empresas de Transporte Coletivo por Ônibus. Fortaleza, 2003.

137 fl., Dissertação (Mestrado em Engenharia de Transportes) – Programa de Mestrado em Engenharia de Transportes, Centro de Tecnologia, Universidade Federal do Ceará, Fortaleza, 2003.

1. Transportes (Dissertação) 2. Alocação de Pessoal

3. *Simulated Annealing* 4. Transporte Coletivo por Ônibus

CDD 388

REFERÊNCIA BIBLIOGRÁFICA

GOMES, H. A. S. (2003) Utilização da Metaheurística *Simulated Annealing* no Problema de Alocação de Pessoal em Empresas de Transporte Coletivo por Ônibus. Dissertação de Mestrado, Programa de Mestrado em Engenharia de Transportes, Universidade Federal do Ceará, Fortaleza, CE, 137 fl.

CESSÃO DE DIREITOS

NOME DO AUTOR: Heider Augusto da Silva Gomes

TÍTULO DA DISSERTAÇÃO DE MESTRADO: Utilização da Metaheurística *Simulated Annealing* no Problema de Alocação de Pessoal em Empresas de Transporte Coletivo por Ônibus.

Mestre / 2003

É concedida a Universidade Federal do Ceará permissão para reproduzir cópias desta dissertação de mestrado e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desta dissertação de mestrado pode ser reproduzida sem a autorização por escrito do autor.

_____

Heider Augusto da Silva Gomes

Rua do Melro, 8091, 1ª Etapa, Cidade Satélite

CEP 59068-010 - Natal-RN - Brasil

UTILIZAÇÃO DA METAHEURÍSTICA *SIMULATED ANNEALING* NO PROBLEMA DE ALOCAÇÃO DE PESSOAL EM EMPRESAS DE TRANSPORTE COLETIVO POR ÔNIBUS

Heider Augusto da Silva Gomes

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE MESTRADO EM ENGENHARIA DE TRANSPORTES DA UNIVERSIDADE FEDERAL DO CEARÁ COMO PARTE DOS REQUISITOS NECESSÁRIOS À OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE TRANSPORTES.

Aprovada por:

_____

Prof. Júlio Francisco Barros Neto, D.Sc.
(Orientador)

_____

Prof. Mário Ângelo Nunes de Azevedo Filho, M.Sc.
(Examinador Interno)

_____

Profª. Maria Elisabeth Pinheiro Moreira, D.Sc.
(Examinador Interno)

_____

Prof. Roberto Diéguez Galvão, Ph. D.
(Examinador Externo)

FORTALEZA, CE – BRASIL
SETEMBRO DE 2003

DEDICATÓRIA

À minha AVÓ, Eutália da Silva Gomes (*in memorian*), pelo apoio e carinho.

Aos meus pais, Marli Ferreira da Silva Gomes e Joel Américo Gomes, e às minhas irmãs, Patrícia, Priscila e Pollyana, que sempre acreditaram no meu potencial.

Ao meu tio, Juvan Augusto Gomes, que sempre investiu na minha formação.

À Ligia Maia Silva, pelo amor e carinho.

Aos meus verdadeiros amigos, pelo carinho e companheirismo.

DEDICO.

AGRADECIMENTOS

A conclusão desse trabalho só foi possível graças à colaboração das seguintes pessoas, a quem dedico meus sinceros agradecimentos:

Ao professor Júlio Francisco Barros Neto pelo apoio, paciência e incentivo para a conclusão deste trabalho.

Aos professores Maria Elisabeth Pinheiro Moreira e Mário Ângelo Nunes de Azevedo Filho, pelo apoio e amizade, fatores importantes para a realização desse trabalho.

Ao professor Carlos Felipe Grangeiro Loureiro, pelo grande incentivo e amizade.

Ao professor Enilson Santos pela enorme ajuda e pelo imenso incentivo.

Em especial, aos meus amigos Alane Barros, Everton Correia, Hamifrancy Brito, Lucimar Santiago, Luciana Mota, Jéferson Meneses pela imensa ajuda e pela grande amizade que foi construída durante todo esse tempo e que sempre será guardada em meu coração.

Aos amigos Álvaro BoaVista, Camila Henrique, Marcos Timbó, Waldemiro, Thiago, Inês, Eduardo Praça, Expedito e demais amigos pelo companheirismo e incentivo.

À Ligia Maia Silva e família pelo carinho, apoio e companheirismo.

Ao Departamento de Engenharia de Transportes, em nome do Prof. Sérgio Benevides, pelo apoio prestado.

À empresa Rotaexpressa Transportes de Passageiros pelo apoio e transparência no fornecimento das informações necessárias para a aplicação do programa.

A toda a minha família, em especial aos meus tios Juvan, Jandira, João, Anselmo e Fanco, a quem agradeço de todo coração pelo grande apoio e carinho.

À Ivone Sales Aleixo pelo imenso carinho, atenção e receptividade.

A Deus por ter me concedido força e perseverança para concluir este trabalho em meio às diversas dificuldades enfrentadas.

Resumo da Dissertação submetida ao PETRAN/UFC como parte dos requisitos para a obtenção do título de Mestre em Ciências (M.Sc.) em Engenharia de Transportes.

UTILIZAÇÃO DA METAHEURÍSTICA *SIMULATED ANNEALING* NO PROBLEMA DE ALOCAÇÃO DE PESSOAL EM EMPRESAS DE TRANSPORTE COLETIVO POR ÔNIBUS

Heider Augusto da Silva Gomes

Setembro/2003

Orientador: Júlio Francisco Barros Neto

A programação das escalas de tripulação (motorista e cobrador) é uma importante etapa no processo de planejamento da operação de transportes coletivos. Esta mão-de-obra de operação representa uma parcela importante nos custos totais de uma empresa operadora, com efeitos diretos na tarifa cobrada ao usuário. Estes custos, o contexto econômico do país e os novos modelos de regulamentação do transporte coletivo têm produzido ações voltadas ao melhor dimensionamento de escalas de veículos e de tripulações. Essa pesquisa tem a finalidade de propor uma nova ferramenta computacional capaz de facilitar o processo da programação de escala de pessoal em empresas de transporte público por ônibus. A concepção metodológica deste trabalho é baseada no uso da metaheurística *Simulated Annealing* com a qual se pretende reduzir o tempo para a obtenção de soluções, possibilitando ao programador, com base na sua experiência, formular, analisar e comparar diferentes alternativas de programação e, ao final, selecionar a mais adequada. O modelo computacional proposto apresentou resultados bastante satisfatórios que representaram uma redução nos custos atuais, para algumas linhas testadas. Porém, verificou-se também que ainda são necessários alguns ajustes a fim de que o programa computacional se torne mais robusto e ainda mais eficiente. Isso se deve ao fato de que o modelo proposto não considera algumas peculiaridades que são adotadas pelas empresas, mas que podem ser realizadas, se necessário, a partir de alterações manuais na solução final fornecida pelo programa.

Abstract of Thesis submitted to PETRAN/UFC as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.) in Transportation Engineering

UTILIZATION OF METAHEURISTIC SIMULATED ANNEALING IN THE CREW SCHEDULING PROBLEM IN PUBLIC TRANSPORTATION COMPANIES FOR BUS

Heider Augusto da Silva Gomes

September/2003

Advisor: Júlio Francisco Barros Neto

Crew scheduling is an important stage of the operational planning of transit. The expenses with drivers and collectors represent a considerable part of the bus companies costs, with direct effects on the values of the fares. Those costs, the economic situation of the country and the new regulation models are generating actions to improve vehicle and crew scheduling procedures. The purpose of the present research is to develop a new software tool to be used for bus crew scheduling. The methodology to be used will consider the application of metaheuristic Simulated Annealing, with which it is intended to produce a set of solutions, allowing the decision maker to select one based on his/her experience that will give a better overcoming. The software tool has presented quite satisfactory results for some tested lines implying in a reduction in the costs adopted by the company operator of that type of transport system. However, it was verified that they are still necessary some agreement. This way, the software tool becomes more robust and more efficient. That is due to the fact that the software tool doesn't consider some peculiarities that they are adopted by the companies, but that can be accomplished, if necessary, starting from manual alterations in the final solution supplied by the program.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

1.1. CONTEXTUALIZAÇÃO

Nas últimas décadas, várias cidades brasileiras vêm passando por um acelerado crescimento urbano que tem como conseqüência um aumento no número de viagens por diversos motivos, ou seja, na necessidade que a população tem de se deslocar. Segundo dados do IBGE, através do censo demográfico 2000, a população total do Brasil é de aproximadamente 170 milhões de pessoas. Além disso, a população urbana cresceu de 12.880.182 de habitantes em 1940 para 137.953.959 em 2000, atingindo um índice de 81,24% em relação à população total.

PEREIRA (1985) lembra que já na segunda metade da década de 80 a demanda de transportes urbanos, por meios motorizados, nas grandes cidades do País era estimada em 56 milhões de viagens/dia quando a população era aproximadamente 119 milhões de habitantes. E a maioria das viagens por transporte público era, na época, realizada por ônibus, responsáveis por mais de 60% do total de viagens diárias em muitas das cidades brasileiras.

Nas áreas urbanas e metropolitanas do Brasil, o ônibus constitui o modo predominante de transporte público para a parcela não-motorizada da população. Embora nas últimas décadas, a aquisição de automóvel venha crescendo de forma acelerada, e mais recentemente ainda surgiram outras opções de modalidade (serviços de vans, metrôs etc), a grande parte da população urbana brasileira ainda depende fortemente do ônibus para realizar suas viagens (SANTOS e ORRICO FILHO, 2000).

A operação do Sistema de Transporte Coletivo por Ônibus (STCO) caracteriza-se pelo inter-relacionamento entre os três grandes elementos envolvidos: o usuário do sistema, a empresa operadora e a comunidade, que pode ser considerada representada pelo Poder Público.

Segundo PEREIRA (1985), o Transporte Coletivo é o serviço público regular e contínuo de transporte de passageiros em veículos que percorrem linhas estabelecidas entre pontos perfeitamente delimitados, segundo itinerários e horários previamente estabelecidos e pagamento individual de passagens, fixadas pelo órgão concedente do serviço de transporte. Além disso, o autor afirma que a produtividade da mão-de-obra é um dos indicadores que medem o desempenho operacional das empresas e, portanto, é uma das variáveis que são utilizadas no cálculo do custo operacional.

Nas cidades brasileiras, as empresas responsáveis pelo STCO são na sua grande maioria privadas, nas quais o maior objetivo é a lucratividade dos seus investimentos. Sendo assim, os objetivos sociais muitas vezes ficam esquecidos, visto que cada empresa apresenta metas individuais que nem sempre convergem com o interesse social.

Segundo FERRAZ e TORRES (2001), as empresas prestadoras de serviço de transporte coletivo urbano têm, em geral, uma organização que segue os princípios de Taylor, resultando em um modelo piramidal cuja estrutura é baseada em uma hierarquia de autoridade bem definida. A distribuição das atividades de comando leva a criação de departamentos e seções (divisões) para racionalização do trabalho (ver Figura 1.1.).

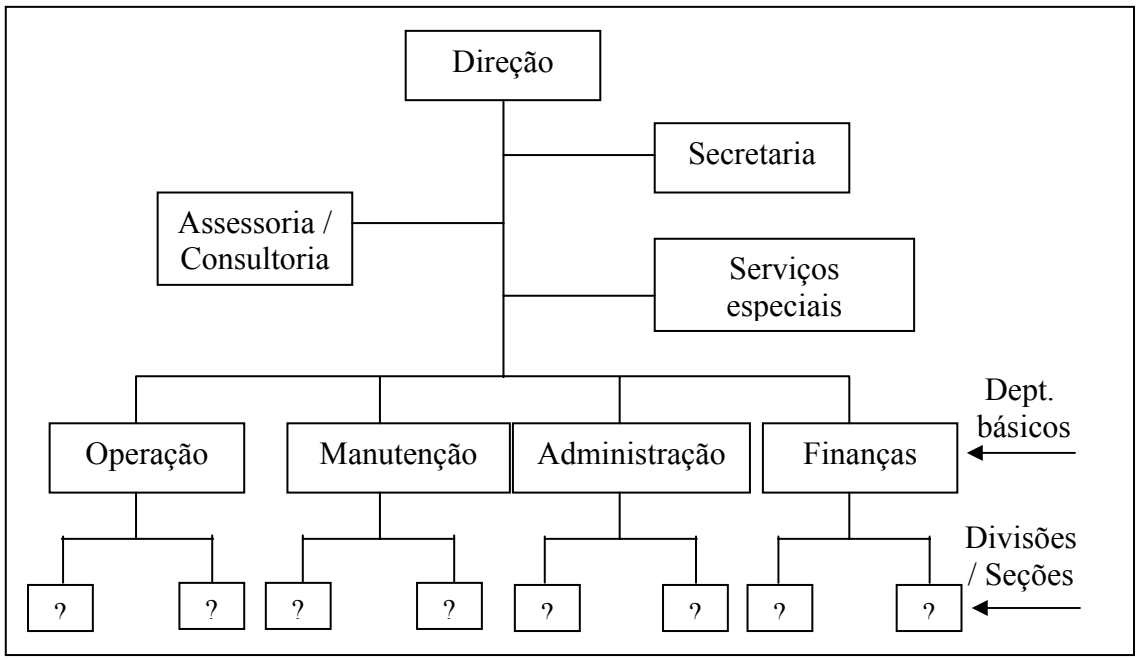

Figura 1.1: Organograma típico das empresas de ônibus urbano (FERRAZ e TORRES, 2001).

Como se pode perceber na Figura 1.1, as empresas de transporte coletivo apresentam quatro departamentos básicos que podem possuir, ou não, divisões dependendo do porte da empresa. Cada divisão ou seção é estabelecida de acordo com a forma de operação da empresa.

Segundo FERRAZ e TORRES (2001), o Departamento de Finanças destas empresas engloba as seções (divisões) de tesouraria, contabilidade, compras, pagadoria e controle financeiro. O Departamento de Administração abrange as seções de pessoal (controla faltas de empregados, horas trabalhadas, elabora folhas de pagamentos, contratação e demissão etc) e a de serviços gerais (responsável pelo patrimônio, serviços de limpeza e manutenção, escala de porteiros, telefonistas etc, e executa serviços de correspondência). Já o Departamento de Manutenção tem por função colocar a frota de veículos a disposição da operação.

Quanto ao Departamento de Operação, FERRAZ e TORRES (2001) afirmam que o mesmo tem a responsabilidade por realizar a produção do transporte de passageiros. É constituído, em geral, pelas seguintes divisões (seções): técnico-administrativa e tráfego.

Conforme FERRAZ e TORREZ (2001) a Seção Técnico-Administrativa é responsável pelas seguintes atividades:

a) definição da programação operacional de cada linha (intervalo entre atendimentos nos diversos períodos dos diferentes dias típicos: dia útil, sábado, domingo e feriado);
b) determinação do número e do tipo de veículo a ser utilizado em cada linha;
c) elaboração das tabelas de horários;
d) elaboração da escala de pessoal;
e) preparação dos modelos de informes a serem utilizados por fiscais, motoristas e cobradores para relatar acidentes e incidentes;
f) controle da ocorrência de acidentes e incidentes durante a operação;
g) medição dos serviços realizados;

h) controle estatístico do transporte;
i) comunicação ao Departamento Pessoal sobre faltas e incidentes com funcionários etc.

E a seção de Tráfego engloba o trabalho dos fiscais e dos operadores (motoristas e cobradores).

Os motoristas e cobradores é que efetivamente fazem o serviço de transporte dos passageiros. A tarefa dos fiscais é proceder a supervisão e o controle da operação, no que diz respeito ao cumprimento da programação operacional e ao comportamento dos operadores. É de responsabilidade dos fiscais tomar as providências necessárias no caso de incidentes e acidentes durante a operação e a emissão dos relatórios correspondentes (FERRAZ E TORRES, 2001).

Essa mão-de-obra de operação representa uma parcela importante nos custos totais de uma empresa prestadora desse tipo de serviço, com efeitos diretos na tarifa cobrada ao usuário. A Tabela 1.1 apresenta o crescimento da participação da mão-de-obra na composição dos custos totais de uma empresa em transporte público por ônibus, entre os anos de 1994 e 1998 (NTU, 1998).

Tabela 1.1: Participação da Mão-de-Obra nos Custos Totais (NTU, 1998).

Ano	**1994**	**1995**	**1996**	**1997**	**1998**
Mão-de-Obra	38%	44%	51%	53%	52%

A Figura 1.2, mostrada a seguir, apresenta o comportamento do salário médio dos motoristas entre os meses de junho de 1994 e dezembro de 1998. Segundo NTU (1998) para os salários de motoristas, os indicadores foram baseados nos dados de todas as capitais brasileiras, coletados pela NTU ao longo dos doze meses do ano. Esses valores foram expressos em reais constantes de dezembro de 1998, com base no IGP-DI da FGV.

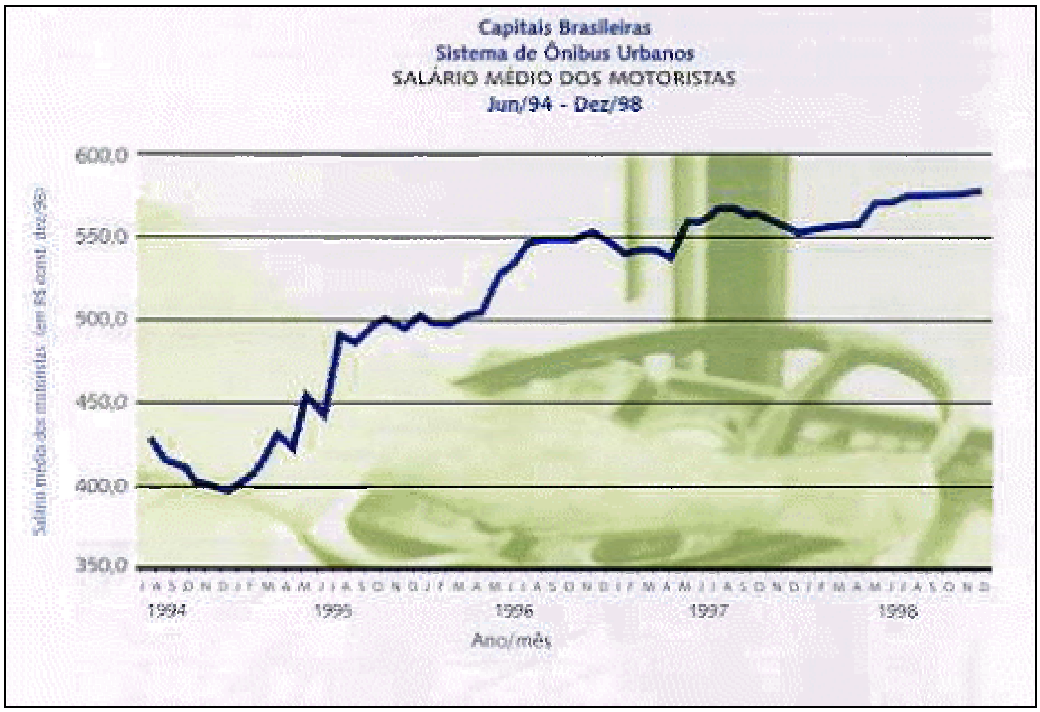

Figura 1.2: Comportamento do Salário Médio dos Motoristas (NTU, 1998).

Com isso pode-se verificar a alta representatividade do custo com pessoal em uma empresa operadora do sistema de transporte coletivo por ônibus. Dados mais atuais são apresentados no capítulo 2 quando da revisão desse sistema de transporte no município de Fortaleza.

Segundo PEREIRA (1985), custo de pessoal de operação representa uma parcela dos custos dependentes e outra dos independentes, de acordo com a metodologia adotada pelo DTC/RJ, sendo considerados três grupos funcionais:

a) motorista;
b) cobrador;
c) outras categorias (despachantes, fiscais, inspetores etc).

Conforme PEREIRA (1985), em relação aos dois primeiros grupos, utiliza-se, nos estudos tarifários, um coeficiente denominado Fator de Utilização (funcionário por veículo), enquanto que em relação ao terceiro grupo, devido à dificuldade de se obter salários padronizados, utiliza-se a agregação dos custos incorridos e a quilometragem rodada, sendo, portanto, um item dos custos independentes.

O custo do item pessoal de operação para o grupo dos motoristas e cobradores é determinado pelo produto entre os fatores de utilização e os respectivos salários constantes dos acordos sindicais, acrescidos do percentual devido aos encargos sociais.

Sendo assim, CUNHA (1992) afirma que a parcela representada pelos salários e encargos sociais da mão-de-obra, o contexto econômico do país e os novos modelos de regulamentação do transporte coletivo têm influenciado em ações voltadas ao dimensionamento de escalas de veículos e de tripulações (motorista e cobrador).

Com isso, é natural que haja uma preocupação, por parte dos responsáveis pela gerência do serviço de transporte público, bem como pelos donos das empresas operadoras, no sentido de obter o melhor aproveitamento, tanto da frota, como da mão-de-obra existente.

Nas últimas décadas estudos vêm sendo desenvolvidos com a finalidade de criar programas computacionais capazes de resolver este tipo de problema de forma mais rápida e eficiente. O uso de procedimentos computacionais possibilita a obtenção das escalas de pessoal, em um menor tempo, permitindo ao programador, com base na sua experiência, formular, analisar e comparar diferentes alternativas de programação e selecionar a mais adequada.

1.2. PROBLEMA DE PESQUISA

O serviço de transporte público por ônibus apresenta características bastante peculiares e complexas no que diz respeito à programação de sua operação, a qual está intimamente ligada à demanda dos usuários do transporte público em uma determinada região. Essa demanda, que varia entre regiões e ao longo do dia, de acordo com os horários de pico e de entre-pico, é que determina as viagens necessárias para a operação do serviço.

A programação de tripulação é uma importante etapa no processo final de planejamento da operação de transportes públicos. A partir da programação dos veículos é que se tem subsídios para se realizar a programação da escala de pessoal necessária

para operar os veículos. O objetivo do processo é minimizar os custos de operação das empresas responsáveis pela operação do serviço, reduzindo, o máximo possível, o número de veículos e o número de jornadas de trabalho necessário para cobrir a programação de veículos.

Segundo AZEVEDO *et. al.* (1994), o problema da alocação de tripulações (*Crew Scheduling Problem*) consiste no estabelecimento de jornadas de trabalho para as tripulações incluindo a definição do número de trechos de trabalho, o início e a duração das jornadas, o uso ou não de horas-extras, etc.

O que se tem conhecimento é que, na maioria dos casos, essa programação é feita manualmente e intuitivamente por funcionários da própria empresa ou por técnicos dos órgãos gestores, responsáveis pela fiscalização desse tipo de serviço. Por serem procedimentos trabalhosos e demorados, as programações das escalas de pessoal são pouco alteradas. Além disso, não se estudam escalas alternativas, o que poderia levar a escolha de uma programação mais eficiente. Com isso, a empresa não tem informações suficientes para garantir que a programação executada é a melhor e a mais eficiente.

Portanto, o procedimento atualmente utilizado para se obter a escala de pessoal em uma empresa de transporte coletivo por ônibus não garante, na maioria das vezes, soluções eficientes e que venham a atingir uma diminuição de custo.

1.3. OBJETIVOS

1.3.1. Objetivo Geral

O objetivo geral desta pesquisa é o de elaborar um procedimento computacional, baseado na metaheurística *Simulated Annealing*, que gere soluções viáveis para o problema de escala de pessoal em empresas de transporte coletivo por ônibus, de forma a minimizar os custos de operação do sistema e o tempo de obtenção da solução.

1.3.2. Objetivos Específicos

Os objetivos específicos são:

a) Analisar as metodologias atualmente utilizadas para a alocação de tripulação;
b) Identificar os procedimentos heurísticos mais utilizados e as suas potencialidades para a solução do problema em estudo;
c) Conceber e implementar o procedimento computacional para definição das escalas;
d) Realizar um estudo de caso para validar o procedimento proposto.

1.4. CONSIDERAÇÕES FINAIS

Este capítulo teve como finalidade apresentar o problema de pesquisa, suas justificativas e os objetivos a serem atingidos.

Além deste tópico, o escopo desse trabalho é composto por uma Revisão Bibliográfica sobre a Programação de Tripulação, que será apresentada no capítulo 2. Uma abordagem sobre Heurísticas e Metaheurísticas será apresentada no capítulo 3. O capítulo seguinte apresenta a metodologia desenvolvida, o modelo computacional, e a aplicação desse modelo a algumas linhas de uma empresa de transporte coletivo por ônibus do município de Fortaleza-CE. E por último, serão apresentadas as conclusões e recomendações.

CAPÍTULO 2

PROGRAMAÇÃO DE TRIPULAÇÃO EM EMPRESA DE TRANSPORTE COLETIVO POR ÔNIBUS

Este capítulo tem a finalidade de apresentar uma revisão bibliográfica sobre o problema de programação de tripulação em empresas de transporte coletivo por ônibus. São apresentados os dados necessários para elaboração de uma escala de pessoal (motorista e cobrador) em uma empresa de transporte coletivo por ônibus, as leis trabalhistas que regulam a operação do serviço, e a metodologia usada pelas empresas. Além disso, são mostradas as potencialidades e limitações de algumas ferramentas computacionais já desenvolvidas para a resolução desse tipo de problema.

2.1. ALOCAÇÃO DE TRIPULAÇÃO EM EMPRESAS DE TRANSPORTE COLETIVO POR ÔNIBUS

Segundo CUNHA (1992), o sistema de transporte público é responsável por uma grande parcela das viagens realizadas todos os dias nas cidades urbanas, sendo o ônibus o modal predominante na operação desse sistema.

Segundo WREN (1996a) *apud* AZEVEDO FILHO (1997), o planejamento da operação de transportes públicos é composto das seguintes etapas:

a) planejar a rede;
b) desenhar as rotas, incluindo os tempos de viagem;
c) decidir os tempos de partida (Quadros de Horários);
d) alocar os ônibus para as jornadas;
e) integrar as programações sobre toda a rede, incluindo as jornadas especiais;
f) construir as programações da tripulação para cobrir todo o trabalho dos ônibus;
g) construir listas para incluir todas as jornadas da tripulação sobre um período de semanas;

h) produzir uma documentação; e,
i) fornecer informações para alimentação de um banco de dados gerencial.

Porém, segundo EBTU (1988) é importante ressaltar que a operação do sistema não é realizada pela mesma equipe que executa o planejamento e a programação do Sistema de Transporte Público de Passageiros (STPP), ou mais especificamente no caso da programação das linhas de ônibus de uma determinada empresa. Por isso é necessário que a equipe envolvida na operação (motoristas, fiscais, chefes de tráfego etc) seja informada de forma simples e direta sobre a especificação da operação a ser executada. Os parâmetros da operação podem ser divididos em três grupos: parâmetros físicos, operacionais, e de âmbito legal.

Os parâmetros físicos se referem às informações que sofrem poucas alterações durante a operação. Dentre eles se destacam: o itinerário (de forma detalhada e considerando cada um dos sentidos da operação), a definição precisa dos pontos extremos da linha (servem de base para o controle da operação), as características do veículo (microônibus, ônibus convencional, tipo PADRON, articulado etc) e a operação nos pontos intermediários.

Os parâmetros operacionais são definidos na programação da linha e sofrem uma maior variação em períodos curtos de tempo (possivelmente meses) devido à adequação da oferta à demanda. Estes parâmetros são os períodos típicos, as tabelas de horários, a otimização da frota em relação à equipe de operação, e a operação nos terminais (por exemplo: tempos máximos e mínimos que os veículos podem ficar parados).

Como o transporte urbano é um serviço delegado pelo Poder Executivo, toda a programação tem que ser regulamentada por uma legislação e formalizada junto aos operadores através de alguns parâmetros legais como, por exemplo: Ordem de Serviço de Operação (OSO) e Ordem de Regulamentação da Operação. A OSO é o documento que formaliza a operação de uma linha de transporte público por ônibus. Na OSO devem constar todas as informações sobre os parâmetros físicos e operacionais definidos na programação do sistema, além de informações sobre procedimentos especiais a serem seguidos pelos operadores (retornos operacionais, horas-extras,

extensões e derivações de linha, etc). Já na Ordem de Regulamentação devem constar os parâmetros gerais do sistema (não vinculados somente a uma linha) e suas condições de operacionalização.

Portanto, finalizada a etapa de planejamento da rede, designação do quadro de horários e determinação dos parâmetros citados anteriormente é realizada a programação de veículos, alocando os veículos para um conjunto de viagens. Segundo WREN e KWAN (1998), a programação de veículos pode ser definida como a alocação de veículos para jornadas previamente definidas. Isto requer a junção de jornadas individuais em blocos que incluem todo o trabalho feito por um veículo entre a partida e o retorno à garagem. O principal objetivo na programação de veículos é minimizar o número de ônibus, enquanto cobre todas as viagens. Concluída essa etapa, é realizada a programação da tripulação (ou escala de motoristas e cobradores), alocando assim um motorista e um cobrador para cada veículo.

Segundo WREN e ROUSSEAU (1993), o processo de programação de escalas de motoristas é a construção de um conjunto de *Jornadas* legais (incluindo horas-extras, quando permitido) que juntas cobrem todos os *Blocos* em uma determinada programação de veículos, seja para a operação como um todo, ou apenas para uma parte reservada da operação. *Blocos* podem ser considerados como sendo um conjunto de unidades de trabalho que iniciam e terminam em *oportunidades de rendição* (ou seja, em um ponto, determinado no tempo e no espaço, onde há a possibilidade de mudança de motorista e cobrador).

Segundo AZEVEDO FILHO (1997), *Jornada* é o trabalho alocado para ser cumprido por uma tripulação em um dia. Uma *Jornada* é formada por uma série de *Pedaços de Trabalho* (porção de trabalho entre dois pontos de rendição) de um ou mais blocos. Uma série de *Pedaços de Trabalho* consecutivos no mesmo ônibus é chamada de *Período de Trabalho* e uma série de *Períodos de Trabalho* consecutivos sem um lanche (ou intervalo) é chamada de *Trecho de Trabalho*. Estes trechos de trabalho podem ser formados por um único *Período de Trabalho* ou por dois ou mais *Períodos de Trabalho*.

Ainda de acordo com AZEVEDO FILHO (1997) as jornadas podem ser divididas em dois tipos: *Jornadas Divididas* e *Jornadas Diretas*. A característica das *Jornadas Divididas* é que há uma longa parada entre dois trechos de trabalho. Estas jornadas são usadas para fornecer tripulação para cobrir períodos de pico quando há mais ônibus em operação. Elas também são úteis para cobrir períodos em que há outros motoristas fazendo a refeição. Jornadas deste tipo são indesejáveis entre a tripulação. As *Jornadas Diretas* são compostas de um ou dois *Trechos de Trabalho*. Dependendo do período do dia coberto pelo turno, as *Jornadas Diretas* são classificadas em diferentes categorias: matutinas (*Early Shift*), vespertinas (*Late Shift*), medianas (*Middle Shift*) e diurnas (*Day Shift*).

Segundo SMITH (1986) as jornadas matutinas (*Early Shifts)* são aquelas que iniciam por tomar um ônibus, também classificado como *matutino*, na garagem e normalmente trabalham nesse ônibus até o fim do pico da manhã. O ônibus classificado como *matutino* é aquele que deixa a garagem antes do prazo máximo de saída para veículos que cumprem jornadas matutinas.

Os motoristas das jornadas vespertinas (*Late Shifts*) levam o ônibus para a garagem após a última viagem, já tarde da noite. As jornadas diurnas (*Day Shifts*) são aquelas na qual a tripulação toma um ônibus de uma outra tripulação que teve uma jornada matutina ou inicia uma jornada da garagem durante a manhã. As jornadas medianas (*Middle Shift*) ou finalizam na garagem após o pico da noite ou entregam o veículo para a tripulação que cumpre uma jornada vespertina (AZEVEDO FILHO, 1997).

Diferentes unidades sucessivas de um bloco podem fazer parte de uma jornada, ou ainda uma jornada pode cobrir parte de diferentes blocos em diferentes ônibus, embora WREN e ROUSSEAU (1993) afirmem que a eficiência é um pouco perdida quando os motoristas mudam, desnecessariamente, entre blocos de ônibus diferentes. A Figura 2.1 exemplifica um caso de uma linha de ônibus com 13 veículos, cada um com os seus blocos de trabalhos e seus possíveis pontos de rendição.

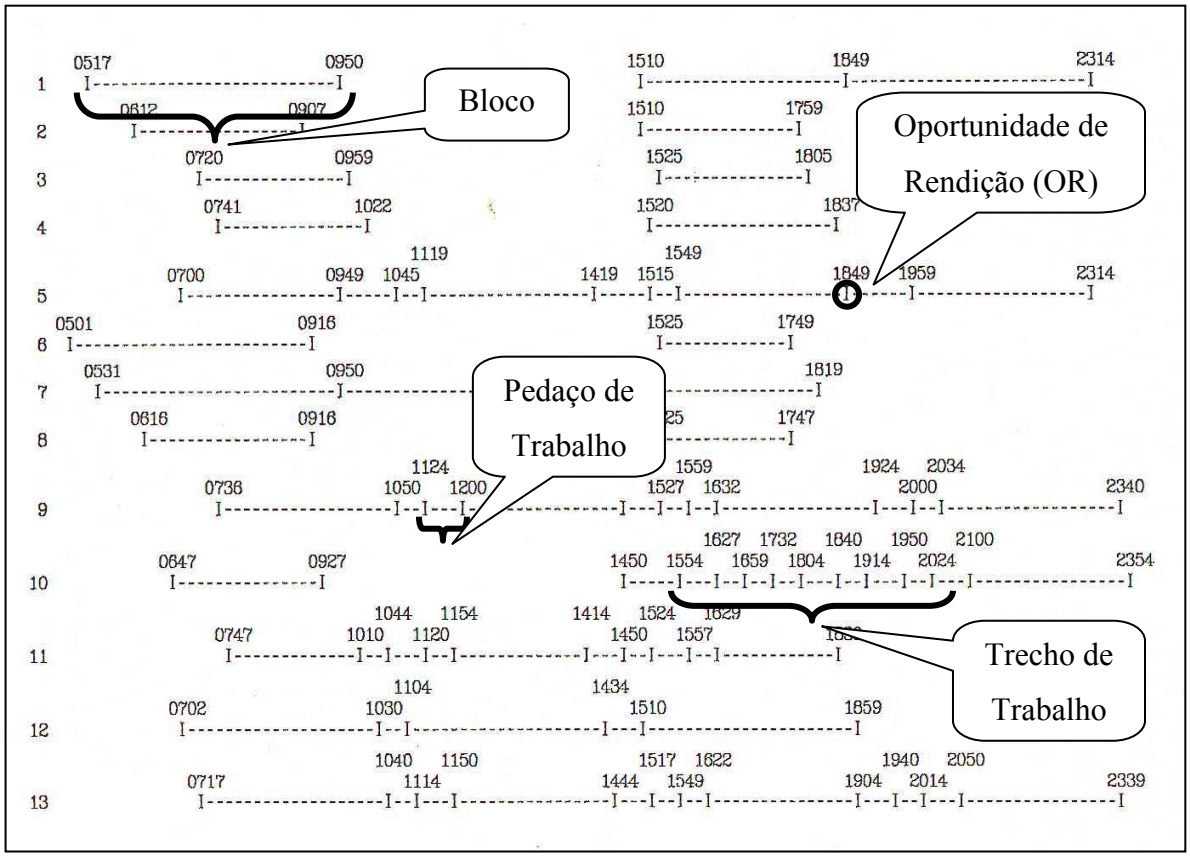

Figura 2.1: Gráfico de programação de ônibus e os pontos de rendição (AZEVEDO FILHO, 1997).

Portanto, a programação da tripulação consiste na elaboração de um conjunto de jornadas que juntas cobrem a programação de veículos. Com isso, o motorista é nomeado para cada jornada. Os principais objetivos considerados quando da programação de tripulação são:

a) ter um motorista alocado para cada veículo todo o tempo que o veículo estiver em uso;
b) as jornadas devem ser designadas de acordo com as leis trabalhistas. Por isso, devem ser levados em consideração:
c) tempo máximo entre início e fim da jornada;
d) tempo máximo trabalhado (tempo dirigindo o veículo);
e) tempo máximo antes do intervalo; e,
f) comprimento mínimo e máximo do intervalo.
g) o número de jornadas deve ser minimizado; e,
h) o custo total deve ser minimizado.

Outro fator que interfere na programação da tripulação de uma empresa de transporte coletivo por ônibus são as leis trabalhistas. Algumas empresas adotam, além das leis trabalhistas, acordos com o sindicato dos motoristas e cobradores e regras específicas com o objetivo de facilitar a sua programação, levando-se em conta o seu modo de operação. Alguns desses acordos podem ser os chamados "acordos de cavalheiro", nos quais não há nenhum documento escrito que prove a sua existência. Outros, porém, são acordos leves que podem ser quebrados em algum caso especial.

A natureza dessas regras tem um grande efeito na programação, tornando-a mais difícil de ser realizada. Em visita a algumas empresas de transporte coletivo por ônibus da cidade de Fortaleza-CE e através da Convenção Coletiva de Trabalho 2002/2003 realizada pelo Sindicato das Empresas de Transporte de Passageiros do Estado do Ceará – SINDIÔNIBUS e pelo Sindicato dos Trabalhadores em Transportes Rodoviários no Estado do Ceará, pôde-se conhecer as leis trabalhistas que regularizam a operação, são elas:

a) a jornada diária deve ser de no máximo 7 horas e 20 minutos, totalizando 44 horas semanais;
b) o intervalo para descanso e refeição é de no mínimo 15 minutos (porém este valor está sofrendo uma alteração e as empresas ainda estão em fase de adaptação ao novo valor que é de 1 hora). Mesmo assim algumas empresas afirmam que serão usados valores entre 30 e 60 minutos;
c) o operário (tanto o motorista quanto o cobrador) não pode ter um intervalo maior do que 4 horas entre um trecho de trabalho e outro;
d) só são permitidas 2 horas-extras (além das 7h e 20 mim) por funcionário;
e) a cada 7 dias, o funcionário tem direito a folgar 1 dia;
f) entre 2 jornadas de trabalho haverá um período mínimo de 11 horas consecutivas para descanso.

Além das leis trabalhistas, as empresas adotam alguns sistemas de trabalho de forma a facilitar o controle e buscar uma melhor eficiência na operação, como por exemplo:

a) a empresa sempre procura manter o motorista na mesma linha e veículo, pois as experiências mostram que o motorista se sente o dono do veículo, passa a cuidar melhor do mesmo, a conhecer melhor o desempenho (mecânico) do veículo e saber quando os problemas passaram a existir. Além disso, a empresa descobre com maior facilidade qual motorista que, possivelmente, danificou o veículo em uma determinada viagem. Algumas empresas também estabelecem que todo dia, por exemplo, o motorista A sempre irá render o motorista B, de forma que apenas esses dois motoristas utilizem o mesmo ônibus. Além disso, o motorista fica conhecendo melhor o itinerário da viagem a ser cumprido facilitando o seu trabalho diário;
b) no caso do cobrador, algumas empresas tentam sempre manter a mesma dupla (motorista e cobrador), enquanto outras realizam mudanças nas duplas de acordo com um período estabelecido. Já que a profissão de cobrador trata-se de um cargo de confiança para a empresa, algumas empresas verificaram que a permanência de uma mesma dupla por muito tempo pode ocasionar uma evasão de renda devido à amizade que é firmada entre o motorista e o cobrador.

Com isso, verifica-se que os dados necessários para se efetuar a programação de tripulação (motoristas e cobradores) são a programação de veículo com os pontos de rendição previamente determinados e o conjunto de leis trabalhistas que regulam a operação desse tipo de serviço.

2.1.1. A Experiência do Município de Fortaleza-CE

Segundo dados do IBGE (2000), Fortaleza apresenta uma população de 2.141.402 habitantes. Os empregos estão concentrados principalmente na área central e o sistema principal de transportes é o sistema de transportes por ônibus. Segundo ETTUSA (2001) cerca de 63% das viagens diárias, em Fortaleza, são feitas por ônibus. O sistema de ônibus é constituído de 224 linhas, operadas por 22 empresas com uma frota de 1704 veículos.

Segundo AZEVEDO FILHO *et al.* (1994) até 1992, o sistema de ônibus em Fortaleza era operado de maneira convencional com um conjunto de linhas, cada linha sendo operada por uma ou mais empresas, sem qualquer facilidade de integração. Em termos de planejamento operacional, as empresas operadoras recebiam do órgão de gerência o intervalo de operação que deveria ser utilizado a cada período do dia (pico e entre-picos) e dia-tipo (dias úteis, sábados, domingos e outros). Então, cabia às empresas a tarefa de alocar veículos e tripulações (motoristas e cobradores). Essas empresas utilizavam um método manual realizado por pessoas com pouca preparação teórica, tendo apenas o conhecimento prático.

AZEVEDO FILHO *et al.* (1994) também observam que, em julho de 1992, começou a ser operado um novo sistema de transporte chamado de "Sistema Integrado de Transportes – SIT", onde foram construídos terminais de integração, localizados nos principais corredores de tráfego; várias linhas que antes iam até o centro da Cidade de Fortaleza, agora param nestes terminais; e passou a existir uma tarifa única para todas as linhas. Além disso, no dia 23 de dezembro de 1993, através da Lei Municipal Nº 7.481, foi criada a Empresa Técnica de Transporte Urbano S. A. – ETTUSA (ETTUSA, 2001).

A função da ETTUSA, como Órgão Gestor do Transporte, é a prestação de serviços a entidades públicas ou privadas na área de transporte e tráfego, tais como: assessoria de planejamento; elaboração e desenvolvimento de projetos; implantação e gerenciamento de sistemas; treinamento de profissionais; pesquisa e acompanhamento de dados; criação, manutenção e atualização de banco de dados; desenvolvimento e acompanhamento do controle da operação; acompanhamento, gerenciamento e implantação de obras e equipamentos de infra-estrutura; administração e coordenação de instalações e equipamentos do sistema; assessoria e elaboração de planilha de custos (ETTUSA, 2001).

Com o SIT, a ETTUSA é responsável pela elaboração de todos os quadros de horários de linhas de ônibus de Fortaleza. A ETTUSA apenas faz a alocação de veículos e envia para a empresa que terá que se adequar aos horários estabelecidos. Em visita a algumas empresas verificou-se que essas também submetem as programações de veículos para a ETTUSA avaliar e dar seu parecer, concordando ou não com o sugerido

pela própria empresa. Para a ETTUSA, além da alocação de veículos, sempre é necessário considerar a alocação das tripulações (motoristas e cobradores), para evitar problemas entre as empresas operadoras e o pessoal de operação. Ou seja, a alocação de veículos é feita com o pensamento prévio de facilitar a alocação das tripulações nas empresas operadoras do sistema.

Segundo AZEVEDO FILHO *et al.* (1994) no SIT as empresas operadoras são remuneradas com base na frota alocada (número e tipo de veículos) e na quilometragem percorrida. Existe uma compensação na qual o lucro das "linhas boas" (superavitárias) é usado para cobrir o prejuízo das "linhas ruins" (deficitárias). "Linha boa" é aquela na qual a receita é maior do que o custo. O sistema em Fortaleza não recebe subsídios provenientes de recursos públicos e a receita total, basicamente o dinheiro arrecadado dos usuários, deve ser igual ao custo total. Neste ponto os procedimentos de alocação de veículos e tripulações têm uma importância muito grande. É necessário que se dimensione o serviço considerando a demanda existente, mas, por outro lado, minimizando o uso de recursos. A Tabela 2.1 apresenta alguns valores de custos das empresas operadoras do sistema.

Tabela 2.1: Percentuais dos custos da tarifa de ônibus urbano - 2001

Custos	JAN	FEV	MAR	ABR	MAI	JUN	JUL	AGO	SET	OUT	NOV	DEZ	Média
CUSTO VARIÁVEL	28,03%	28,33%	28,12%	28,14%	27,30%	27,33%	28,71%	28,90%	28,66%	28,56%	28,68%	28,67%	28,29%
Combustível	18,70%	18,96%	18,77%	18,76%	18,21%	18,23%	19,17%	19,52%	19,28%	19,12%	19,33%	19,31%	18,95%
Óleos e Lubrificantes	1,08%	1,11%	1,11%	1,11%	1,07%	1,07%	1,43%	1,13%	1,12%	1,12%	1,15%	1,14%	1,14%
Rodagem	3,25%	3,28%	3,25%	3,24%	3,15%	3,15%	3,32%	3,47%	3,44%	3,45%	3,03%	3,04%	3,26%
Peças e acessórios	5,00%	4,99%	4,99%	5,04%	4,87%	4,88%	4,80%	4,78%	4,82%	4,87%	5,17%	5,17%	4,95%
CUSTO FIXO	60,82%	60,52%	60,73%	60,71%	61,55%	61,53%	60,14%	59,96%	60,20%	60,29%	60,18%	60,18%	58,93%
Remuneração	3,09%	3,07%	3,07%	3,05%	2,92%	2,87%	2,83%	2,80%	2,77%	2,89%	3,20%	3,19%	2,98%
Depreciação	5,92%	5,87%	5,86%	5,84%	5,61%	5,53%	5,43%	5,35%	5,32%	5,57%	6,15%	6,16%	5,72%
Pessoal Oper. E Man.	44,15%	43,91%	44,12%	44,11%	44,93%	45,02%	43,94%	43,88%	44,13%	43,86%	42,82%	42,83%	43,98%
Despesas Adm.	6,25%	6,21%	6,24%	6,24%	6,41%	6,43%	6,27%	6,26%	6,30%	6,26%	6,11%	6,11%	6,26%
IMPOSTOS E TAXAS	11,15%	11,15%	11,15%	11,15%	11,15%	11,14%	11,15%	11,14%	11,14%	11,15%	11,14%	11,15%	11,15%
CUSTO TOTAL	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%

Fonte: ETTUSA/DICUT (ETTUSA, 2001)

Pode-se verificar, a partir da Tabela 2.1, a alta representatividade dos custos fixos (em média 58,93% no ano de 2001) nos custos totais de uma empresa operadora desse tipo de sistema. Além disso, observa-se que os custos com Pessoal de Operação e Manutenção representaram em média 43,98% (no ano de 2001) dos custos totais de uma empresa. Com isso, reforça-se aqui a importância que deve ser dada à operação de alocação de veículos e tripulação dentro de uma empresa operadora de sistema de ônibus.

Segundo AZEVEDO FILHO *et al.* (1994) o método de alocação da frota usado no início era o método gráfico (gráfico de marcha), podendo um técnico gastar de 8 a 16 horas para fazer uma escala, dependendo da demanda da linha (número de veículos, viagens, *etc*). Este método mostrou-se impraticável e foi desenvolvida uma nova ferramenta usando um *software* de planilha eletrônica, caracterizando-se ainda como um método manual, porém o uso do computador fez com que o tempo gasto para a criação de uma escala reduzisse para o tempo de 1 a 3 horas.

No entanto, essa técnica não garante encontrar a melhor solução, nem fornece várias escalas para que se possa escolher a melhor. Além disso, existem muitos técnicos realizando essa tarefa, gastando muitas horas de trabalho. Com isso, é necessário adotar outras técnicas de forma a minimizar o tempo de obtenção de uma solução e que melhorem os resultados.

2.2. APLICAÇÕES E *SOFTWARE* DESENVOLVIDOS: LIMITAÇÕES E POTENCIALIDADES

A seguir serão abordadas algumas aplicações desenvolvidas para a solução do problema de alocação de pessoal em empresas de transporte coletivo por ônibus. Essas aplicações foram desenvolvidas com o uso de heurísticas e/ou procedimentos baseados em programação matemática.

Segundo SHEN e KWAN (2000) a programação de motoristas de transporte público usando computadores iniciou-se em 1960. Conforme WREN *et al.* (2000) o trabalho mais antigo em programação de motoristas de ônibus foi empreendido no

início da década de 60 por DEUTSCH em uma associação com a IBM na Universidade de Oxford, mas não foi publicado. Baseado em dados do transporte de Londres, DEUTSCH desenvolveu três técnicas de busca exaustiva, que falharam em produzir boas soluções até mesmo em pequenos problemas.

Desde então, muito progresso tem sido feito e muitas abordagens têm sido desenvolvidas. Para tentar solucionar este tipo de problema iniciou-se primeiramente por usar heurísticas (seqüência de passos bem definidos para solucionar um determinado tipo de problema), que são altamente confiantes em uma experiência específica. Já que o problema é complexo, as heurísticas não se adequam prontamente para diferentes operações de transporte e, freqüentemente, falham em escapar de um ótimo local.

Por volta de 1980 foi reconhecido que somente heurísticas não eram satisfatórias para o uso geral (WREN e ROUSSEAU, 1995 *apud* SHEN e KWAN, 2000). A abordagem puramente heurística foi abandonada em favor das abordagens de programação matemática auxiliada por heurísticas.

Segundo SHEN e KWAN (2000) problemas de programação de motoristas são conhecidos como problemas *NP-hard* (ou NP-difícil), e, em geral, não é possível obter uma solução ótima. Segundo TOSCANI (2000) a classe NP-difícil é a classe dos problemas *P* tais que para todo problema *P*' pertencente à classe *NP*, pode-se reduzir *P*' a *P*. O problema *P* não necessariamente pertence à classe *NP*. Classe *NP* é a classe dos problemas de decisão que podem ser resolvidos por algoritmos não-determinísticos em tempo polinomial.

Os métodos baseados em programação matemática são incapazes de resolver o problema de uma só vez, e a subdivisão do problema inevitavelmente leva a uma perda de otimalidade na solução recombinada. O problema encontrado no uso de métodos baseados em programação matemática ou técnicas de solução inteira é que problemas complexos demandam uma grande quantidade de tempo para serem processados e podem finalizar sua execução sem que uma solução inteira tenha sido encontrada.

Já que o uso de uma formulação matematicamente embasada pode não resolver problemas muito grandes de forma ótima. Esses problemas são divididos em subproblemas que poderão ser resolvidos através de métodos heurísticos. O mesmo se faz com alguns métodos heurísticos nos quais se usam métodos baseados em programação matemática para resolver subproblemas que surgem durante a construção ou refinamento da programação (SMITH, 1986).

Recentemente, as metaheurísticas têm sido altamente e satisfatoriamente usadas buscando soluções práticas próximas do ótimo para problemas NP-*hard*. A principal vantagem das metaheurísticas é que elas são geralmente muito eficientes em procurar boas soluções através de espaços de soluções muito grandes, e cada classe de metaheurística tem uma estrutura estratégica e metódica que é independente do domínio do problema (REEVES, 1993).

AZEVEDO FILHO (1997) afirma que mesmo com o extenso uso dos computadores há diferentes casos em que a programação de veículos e/ou tripulação é preparada manualmente. Procedimentos manuais freqüentemente demandam muitas horas de trabalho e a qualidade dos resultados depende da experiência do programador.

Outra dificuldade, algumas vezes encontrada com o uso de procedimentos automáticos, é a incapacidade de um programa de aplicação geral considerar todos os requisitos do problema, sendo necessária algumas adaptações ou até mesmo o desenvolvimento de novos programas. Entretanto, com a disponibilidade de computadores mais poderosos e o desenvolvimento de programas mais genéricos, a maioria dos problemas de programação pode ser resolvida com pouco ou nenhum ajuste (AZEVEDO FILHO, 1997).

De acordo com AZEVEDO FILHO (1997) as vantagens do uso de computadores para programação de veículos ou de tripulação são:

a) melhores soluções;
b) resultados mais rápidos;
c) múltiplas soluções;

d) programadores treinados.

Nos próximos sub-itens abaixo são apresentadas algumas aplicações de alocação de pessoal usando o método manual, heurísticas e técnicas baseadas em programação matemática. Aplicações utilizando metaheurísticas serão apresentadas no próximo capítulo quando da revisão do conceito de algumas metaheurísticas mais conhecidas e suas potencialidades.

2.2.1. Método Manual Aplicado no Município de Fortaleza-CE

Segundo AZEVEDO FILHO *et al.* (1994) o método manual consiste em:

a) gerar, após o estabelecimento dos intervalos adotados para cada período do dia, o quadro de horários da linha, usando um *software* de planilha eletrônica, neste caso o *Microsoft Excel* (versão para *Windows*);
b) alocar os veículos às viagens de acordo com os tempos de partida e chegada a cada terminal;
c) fornecer, nos períodos apropriados, um intervalo de mínimo estabelecido por lei entre a chegada ao terminal e a partida do mesmo, para que a tripulação possa realizar a refeição ou merenda;
d) procurar manter, na medida do possível, o critério de recolher primeiro os veículos que começaram mais cedo, evitando assim jornadas curtas (cobrindo só os picos);
e) tentar manter, na medida do possível, a duração do período de operação do veículo de forma que no máximo 2 tripulações usem o veículo, ou seja, aproximadamente em torno de 14:40 horas (duas jornadas de 7 horas e 20 minutos).

Com isso, a alocação de motoristas e cobradores é feita tentando usar ao máximo uma ou duas jornadas com um único trecho de trabalho por veículo e poucas jornadas de dois trechos para agregar tarefas curtas de períodos de pico com outras em períodos de entre-picos.

2.2.2. *Software* ALOCA

O ALOCA é um software desenvolvido através do curso de Engenharia Civil e o Departamento de Engenharia de Transportes da Universidade Federal do Ceará, para o Órgão de Gerência do Sistema de Transportes de Fortaleza. O software foi escrito em Pascal tentando simular o processo manual. Este *software* não está em uso, mas foram obtidos resultados bons e mais rápidos do que o método manual. Porém, algumas soluções geradas não são muito aceitáveis por apresentar jornadas de curta duração, por exemplo. Isso ocorre principalmente quando se tem uma parte significativa da frota operando somente nos picos (AZEVEDO FILHO *et al.*, 1994).

O processo de programação começa com a construção de uma lista de horários de partida de um terminal e respectivos horários de chegada no outro terminal. Os ônibus são alocados na ordem cronológica. O ALOCA faz apenas a alocação dos veículos, e a alocação das tripulações é realizada seguindo o método manual apresentado no sub-item 2.2.1.

2.2.3. Procedimentos Heurísticos

Alguns procedimentos heurísticos tentam simular o processo manual usado pelos programadores. Porém, quando ocorrem algumas mudanças no sistema de ônibus, uma remodelagem, ou até mesmo, uma reformulação de alguns programas pode ser necessária (AZEVEDO FILHO, 1997).

Sistemas tais como ELIAS (1964) *apud* SMITH (1986), TRACS (PARKER e SMITH, 1981 *apud* AZEVEDO FILHO, 1997), RUCUS (HILDYARD e WALLIS, 1981 *apud* AZEVEDO FILHO, 1997), RUCUS II (LUEDTKE, 1985 *apud* AZEVEDO FILHO, 1997), e outros mais (outros procedimentos heurísticos podem ser encontrados em SMITH, 1986) são exemplos de aplicações heurísticas para a resolução do problema de alocação de tripulação.

- **Sistema Elias**

Segundo WREN *et al.* (2000) ELIAS iniciou os trabalhos aproximadamente na mesma época que DEUTSCH, no início dos anos 60, e desenvolveu uma série de

heurísticas que produziam, rapidamente, um grande número de programações (escalas de pessoal), e a partir disso escolhia-se a melhor. Resultados razoáveis foram obtidos em algumas situações práticas, nos quais eram estabelecidas restrições leves. Subseqüentemente, ELIAS adicionou alguns refinamentos heurísticos não-publicados. Esses refinamentos foram subseqüentemente estendidos por WARD e DEIBEL (1972) *apud* WREN (2000) que obtiveram consideráveis sucessos, embora o trabalho tenha sido abandonado no início da década de 70.

De acordo com SMITH (1986) a programação de tripulação é dividida em três fases:

a) O quadro ou o "bloco" de viagens é dividido em jornadas diretas com o resto do quadro sendo composto por períodos de trabalho menores que 7 horas. Para blocos maiores que 8 horas, uma viagem direta (no caso de 8 horas) é cortada do início do bloco. Se não há tempos de rendição fornecendo uma viagem que resulte exatamente numa viagem direta de 8 horas, o programa escolhe viagens curtas ou longas, sejam elas baratas ou não. Se ainda houver blocos maiores que 8 horas, outra viagem direta é cortada do fim do bloco. Estes passos são repetidos, sempre colocando as viagens diretas de 8 horas para o fim do bloco.
b) Com isso, o programador tem em mãos uma lista de formas alternativas de quebrar cada bloco, e escolhe uma seleção de viagens diretas. O restante dos pedaços de trabalho é combinado em todas as formas possíveis pelo segundo programa para formar uma lista de viagens divididas.
c) Novamente, o programador terá que escolher um conjunto de viagens divididas desta lista. Se houver um número grande de pedaços de trabalho únicos, o programador poderia escolher algumas das viagens diretas da primeira fase, e esta seria dividida e combinada com os pedaços restantes por um terceiro programa.

- ***Software* TRACS**

De acordo com WREN *et al.* (2000) o TRACS foi designado para operações de ônibus urbano, embora ele também possa ser aplicado com sucesso em serviços interurbanos curtos e rurais (intermunicipais).

Segundo SMITH (1986) os trabalhos para o desenvolvimento desse sistema iniciaram na Universidade de Leeds na Inglaterra, em 1967 e a sua solução inicial é construída de forma mais cuidadosa do que o apresentado no sistema ELIAS, pois experiências mostraram que as técnicas de melhoramento podem não transformar uma programação inicial ruim em uma boa solução.

Segundo CURTIS (2000) uma das razões pelas quais isto pode ter sido verdade foi que o desenvolvimento deste sistema iniciou em 1967 e técnicas modernas de busca local ainda não eram disponíveis. Para CURTIS (2000) uma solução ruim seria um mínimo local pobre no espaço de busca e teria vários movimentos de não-melhoramento para chegar a um estágio em que ela pudesse ser melhorada significativamente. Assim, antes do TRACS fazer melhoramentos heurísticos, similares ao RUCUS (que será citado no próximo sub-item), ele primeiro concentra-se em produzir uma solução inicial tão boa quanto possível.

Na construção da solução inicial cada tipo de jornada é construído separadamente. As jornadas matutinas (ou *Early Shifts)* são construídas primeiramente, uma de cada vez, iniciando pelo ônibus que apresenta a oportunidade de rendição mais cedo para esse tipo de jornada e unindo esse 1º trecho de trabalho a outro trecho que exista antes ou após o já escolhido. Posteriormente, são formadas as jornadas vespertinas (*Late Shifts*) e as jornadas medianas (*Middle Shift*), de forma similar, considerando do fim para o início do dia. Os trabalhos que compreendem o pico da tarde e que são viáveis para serem alocados às jornadas divididas não são considerados nesse momento. A partir disso, os primeiros e segundos trechos de trabalho das jornadas divididas são emparelhados e alguns trabalhos que compreendem o pico da manhã são atribuídos às jornadas matutinas extras.

As rotinas de refinamento no TRACS são mais poderosas do que as de outros sistemas heurísticos. Um conjunto de rotinas tenta reduzir o número de jornadas e o número de pedaços não-alocados. Cada jornada é considerada, uma de cada vez, para verificar se todo o trabalho contido nela pode ser acomodado em outras jornadas. Os pedaços não-alocados são reduzidos no comprimento se possível, até mesmo se eles não podem ser eliminados completamente, ou são modificados por pedaços curtos de jornadas (SMITH, 1986).

Essas rotinas de refinamento consistem na construção de duas listas onde em cada lista está contida a metade das jornadas. A 1ª lista contém as metades das jornadas matutinas, divididas e medianas, com as segundas metades das jornadas tardias, e a outra lista contém as metades complementares. O emparelhamento dos membros de uma lista com os da outra lista é resolvido como um problema de alocação. Outra rotina consiste em considerar as oportunidades de rendição e movimentá-las para um ponto de rendição adjacente se o custo for reduzido. Além disso, uma outra rotina considera pares de jornadas para observar se o custo pode ser reduzido por modificar trechos de jornadas, ou por mover um trecho da jornada de uma para a outra. Todas essas rotinas podem ser usadas em qualquer ordem, e algumas mais de uma vez.

Segundo SMITH (1986) o sistema foi usado para compilar diferentes programações para um número de diferentes companhias de ônibus. O principal obstáculo para uma implementação satisfatória foi a dificuldade de adaptar o sistema para cada novo modo de programação. O sistema foi designado para ser modificado facilmente mantendo os requisitos primários, de forma que as jornadas formadas seriam válidas sob as novas regras. Entretanto, para se produzir boas programações, foram necessários modificações consideráveis no método de construção da solução inicial. Com isso, as companhias de ônibus relutaram em investir no trabalho necessário para se realizar tais modificações, pois poderia não garantir um sucesso final.

- ***Software* RUCUS**

Segundo SMITH (1986) RUCUS é um sistema de programação de ônibus e tripulação (o nome significa *RUn CUtting and Scheduling*) que foi desenvolvido pela

MITRE *Corporation* sobre o patrocínio da *U.S. URBAN MASS TRANSPORTATION ADMINISTRATION* no início dos anos 70.

O elemento principal do sistema de programação de tripulação é um programa chamado RUNS que primeiramente usa métodos heurísticos para criar, e depois melhora, uma programação inicial.

Segundo CURTIS (2000) o RUCUS primeiro cria jornadas de período único (ou seja, o motorista permanece dentro do mesmo ônibus sem parar para lanches) e jornadas de dois períodos, após este processo os pedaços de trabalho restantes que não são alocados para alguma jornada são tratados como períodos de hora-extra.

Para CURTIS (2000) isso limita o uso do sistema, sendo a razão para o desuso do RUCUS, pois muitas companhias não usam horas-extras ou quando usam, tentam ao máximo minimizá-las. Uma vez a solução inicial tenha sido criada, o sistema usa movimentos de busca local para melhorá-la. O sistema troca alguns pedaços de trabalho cobertos por uma jornada por pedaços de trabalho de outra jornada, ou move oportunidades de rendição para frente ou para trás. Há então um procedimento de reparo que cuida de fixar algumas jornadas que tenham se tornado inválidas após as mudanças. Mesmo assim, ainda se encontra um número alto de jornadas inválidas na programação final necessitando assim de uma intervenção manual.

Para SMITH (1986) o RUCUS não manipula jornadas de 3 trechos de trabalho e embora o programador possa modificar a programação produzida ao variar os valores dos parâmetros iniciais, o mecanismo para fazer isto é desajeitado, e pode ser difícil de predizer a forma com que os parâmetros afetam a programação, de forma que sem uma grande porção de experiência, o programador é reduzido a fazer mudanças aleatórias, ao invés de controlar o processo de programação.

Para FORES (1996) o sistema RUCUS se mostrou difícil de ser operado quando se consideravam muitos parâmetros e eram necessárias muitas execuções para que se alcançassem bons resultados.

Um estudo realizado, em 1975, pela *METROPOLITAN TRANSIT COMMISSION* (MTC), St. Paul, Minnesota, documentou economias específicas em alguns casos. A análise custo-benefício da MTC demonstrou as seguintes vantagens do RUCUS: elimina o programador de atividades repetitivas e que sejam propensas ao erro; produz jornadas eficientes e eficazes em relação ao custo; e fornece uma principal contribuição para a administração da informação do banco de dados do sistema (*U.S. DEPARTMENT OF TRANSPORTATION*, 1985).

Ainda de acordo com *U.S. DEPARTMENT OF TRANSPORTATION* (1985) os resultados produzidos pelo *software* RUCUS foram comparados com soluções de programação manual aplicadas nas cidades de Akron, Baltimore e San Diego. A alocação de motoristas produzida pelo RUCUS economizou aproximadamente 2% em horas pagas ao motorista em relação à programação manual.

- ***Software* RUCUS II**

O sistema RUCUS II é uma nova versão do sistema RUCUS. A versão preliminar do RUCUS II foi lançada em 1982 e foi descrita por LUEDTKE (1985) *apud* SMITH (1986).

As principais modificações foram em relação à especificação dos parâmetros de entrada. Segundo FORES (1996) o RUCUS II melhorou a entrada de especificação dos parâmetros e alterou levemente o método de produzir a solução inicial, mas ainda usou a mesma heurística que considera a troca de partes de jornadas para uma oportunidade de rendição alternativa.

SMITH (1986) afirma que o RUCUS II fornece ao programador um número de comandos para direcionar a formação da programação. Seis destes comandos formam certos tipos de jornadas automaticamente, e aproximadamente correspondem aos passos pelos quais o RUCUS I forma a programação inicial. Dois comandos permitem ao programador especificar uma jornada com 1 ou 2 trechos de trabalho a ser adicionada à programação, e outros dois comandos eliminam uma jornada existente.

Ainda de acordo com SMITH (1986) as heurísticas empregadas pelos comandos automáticos parecem ser as mesmas como em RUCUS I, com exceção de duas versões de cada um dos comandos que formam jornadas diretas matutinas e vespertinas com dois trechos de trabalho. Um par de comandos é mais sofisticado que o outro, pois eles tentam encaixar as jornadas formadas através da construção de cadeias de refeições.

Porém, FORES (1996) afirma que a busca heurística é ainda muito ineficiente e a versão melhorada do RUCUS II tem melhorado, balanceando e trocando heurísticas para usar um algoritmo de emparelhamento (*matching*) que minimize uma função custo. Esta versão conseguiu bons resultados, embora a mesma dependa da qualidade da programação inicial.

Segundo SMITH (1986) o RUCUS II atual não fornece ao programador um maior controle sobre a formação da programação, pois as tarefas formadas pelos comandos automáticos são adicionadas à programação imediatamente, e não são apresentadas para o programador para que ele possa aprová-las primeiro, embora elas possam ser excluídas uma por uma como o uso de comandos manuais.

2.2.4. Procedimentos Matemáticos

Segundo FORES (1996) abordagens de programação matemática trabalham na definição de que cada variável representa uma jornada, e as restrições representam pedaços de trabalho que necessitam ser cobertos. Como o número de jornadas válidas para algum problema é tipicamente muito grande, a maioria dos sistemas incorpora heurísticas para eliminar jornadas que são improváveis de serem usadas em boas programações, e assim o modelo reduzido pode ser resolvido muito mais rápido. A desvantagem do uso de abordagens com programação matemática é que o encontro do ótimo é limitado pelo conjunto de jornadas que foi gerado, e após algumas restrições já terem sido fixadas o programador tem pouco controle sobre o processo.

O problema de alocação de tripulação (motorista e cobrador) se configura como um problema de *set covering* ou *set partitioning* no qual um grande número de jornadas é formado, do qual uma programação de tripulação é selecionada para cobrir todo o

trabalho na programação de ônibus no mínimo uma vez (*set covering*) ou exatamente uma vez (*set partitioning*).

Sendo assim, o problema de *set covering* pode ser descrito como um problema de Programação Linear Inteira (PLI) da seguinte forma:

$$Min \sum_{j=1}^{n} c_j x_j \tag{2.1}$$

$$sujeito\ a: \sum_{j=1}^{n} a_{ij} x_j \geq 1 \qquad i = 1,2,.....,m \tag{2.2}$$

$$\text{e } x_j = 0 \text{ ou } 1 \qquad j = 1,2,.....,n \tag{2.3}$$

Como mencionado anteriormente, para o problema de programação de tripulação, as *m* restrições correspondem aos pedaços de trabalho na programação de ônibus, e as *n* variáveis correspondem às jornadas do conjunto gerado, de forma que $a_{ij} = 1$ se a *j-ésima* jornada cobre o *i-ésimo* pedaço de trabalho, caso contrário será igual a 0. c_j é o custo associado com a *j-ésima* jornada. A variável x_j será igual a 1 se a *j-ésima* jornada está na programação, caso contrário será igual a 0.

As restrições indicam que cada pedaço de trabalho pode ser coberto por no mínimo uma jornada. Se as desigualdades são substituídas por igualdades, ele se torna um problema *set partitioning*, e cada pedaço de trabalho deve então ser coberto por exatamente uma jornada. Segundo FRELING *et al.* (2000) a vantagem de se trabalhar com a formulação do *set covering* é a facilidade de resolvê-la em relação ao modelo de *set partitioning*. E algumas mudanças podem ser feitas na solução, após a resolução do problema de *set covering*, tornando a mesma uma solução de um problema de *set partitioning* através da eliminação da sobre-cobertura (pedaços de trabalho cobertos por mais de uma jornada). Outra principal vantagem citada por FRELING *et al.* (2000a) é o ganho em tempo computacional ao se usar o modelo de *set covering*.

Já que cada restrição corresponde a um pedaço de trabalho, definido como o intervalo entre duas oportunidades de rendição consecutivas na programação de ônibus, é de se esperar que em operações de ônibus urbano, com oportunidades de rendição bastante freqüentes, existam muitas restrições no PLI, ao menos que o conjunto de pedaços de trabalho seja reduzido ou que o problema seja decomposto em pequenas unidades.

Segundo WREN e ROUSSEAU (1993) os sistemas mais conhecidos são provavelmente os da Teleride-Sage, UMA, HASTUS, IMPACS e HOT. Outros trabalhos também citam o COMPACS (FORES, 1996), IMPACS (SMITH, 1986) EXPRESS (CURTIS, 2000) e TRACS II (SMITH, 1986).

A seguir serão relatadas algumas características dos sistemas HASTUS, IMPACS e TRACS II.

- ***Software* HASTUS**

Segundo FORES (1996) HASTUS é um pacote completo de programação desenvolvido originalmente em 1974 pelo Centro de Pesquisa em Transportes da Universidade de Montreal, e em colaboração com *GIRO* inc., Canadá. Além de seu uso em programar veículos e motoristas, são disponíveis também *software* para sistemas de informação aos passageiros, sistemas de operação de trânsito e ferramentas de planejamento avançado.

O sistema HASTUS contém um conjunto de programas tanto para a programação de tripulação quanto para a programação de veículo. Segundo CURTIS (2000) o componente de programação de tripulação do HASTUS é constituído de dois sistemas, HASTUS-micro e HASTUS-macro. O HASTUS-macro fornece uma solução inicial e o HASTUS-micro gera uma solução final. Porém, um novo componente já foi desenvolvido e é chamado de *Crew-Opt*. O *Crew-Opt* é um novo módulo de programação de motoristas implementado dentro do sistema HASTUS, que usa uma técnica de geração de colunas para resolver uma formulação de *set covering* do problema.

De acordo com SIQUEIRA (1999) as escalas de trabalho definidas pelo sistema HASTUS são construídas em três fases:

a) 1ª Fase: uma escala aproximada é construída usando-se um problema de programação linear, relaxando-se as restrições de integralidade e factibilidade com o objetivo de reduzir o tempo computacional. Assim, obtém-se as primeiras escalas que são chamadas de pedaços de trabalho;
b) 2ª Fase: consiste na combinação destas escalas utilizando-se do algoritmo de *matching*, atribuindo-se pesos às combinações factíveis de escalas de trabalho. A melhor solução resultante da combinação é aquela de peso máximo;
c) 3ª Fase: as divisões da escalas são reconsideradas a fim de que sejam realizadas algumas melhorias visando a redução do número de funcionários e a quantidade de horas-extras.

A descrição mais detalhada do sistema HASTUS pode ser encontrada em SMITH (1986), BLAIS *et al.* (1976) *apud* SIQUEIRA (1999), LESSARD *et al.* (1981) *apud* SIQUEIRA (1999) e ROUSSEAU *et al.* (1985) *apud* SIQUEIRA (1999).

- ***Software* IMPACS e TRACS II**

O sistema TRACS II (*Techniques for Running Automatic Crew Schedules*) é derivado do programa IMPACS (*Integer Mathematical Programming for Automatic Crew Scheduling*) desenvolvido na Universidade de Leeds. Segundo AZEVEDO FILHO (1997), a primeira versão do sistema IMPACS foi apresentada por PARKER e SMITH em 1981 com o primeiro protótipo do IMPACS sendo instalado na companhia *LONDON TRANSPORT* em 1983 e, em 1985, o programa completo foi aceito. Após isto, o *software* foi instalado para mais companhias.

Segundo FORES *et al.* (2000) o TRACS II usa uma abordagem de programação matemática que utiliza uma formulação de *set covering* ou *set partitioning* para assegurar que todo o trabalho do veículo é coberto, com uma função objetivo para diferenciar as soluções.

Segundo KWAN *et al.* (2000) o sistema TRACS II foi primeiro desenvolvido para a programação de motoristas de trens. Porém, de acordo com WREN e GUALDA (1997) *apud* AZEVEDO FILHO (1997), este *software* também produz bons resultados para programação de tripulação de ônibus.

Além disso, KWAN *et al.* (2000) afirmam que a habilidade do TRACS II para resolver problemas muito complexos, apresentados por operadores de trem, é de grande benefício para companhias de ônibus, cujas operações apresentam uma quantidade menor de restrições. Entretanto, elas apresentam aspectos de operação que são únicos para a indústria de ônibus.

Segundo FORES *et al.* (2000) o sistema TRACS II apresenta três estágios para a construção de uma solução:

a) geração de jornadas válidas;
b) redução do conjunto de jornadas válidas; e,
c) seleção de um sub-conjunto de jornadas que cubra todo o trabalho dos veículos e que minimize a função objetivo.

Com base nesses estágios, CURTIS (2000) apresenta o modelo mostrado na Figura 2.2 para o sistema TRACS II:

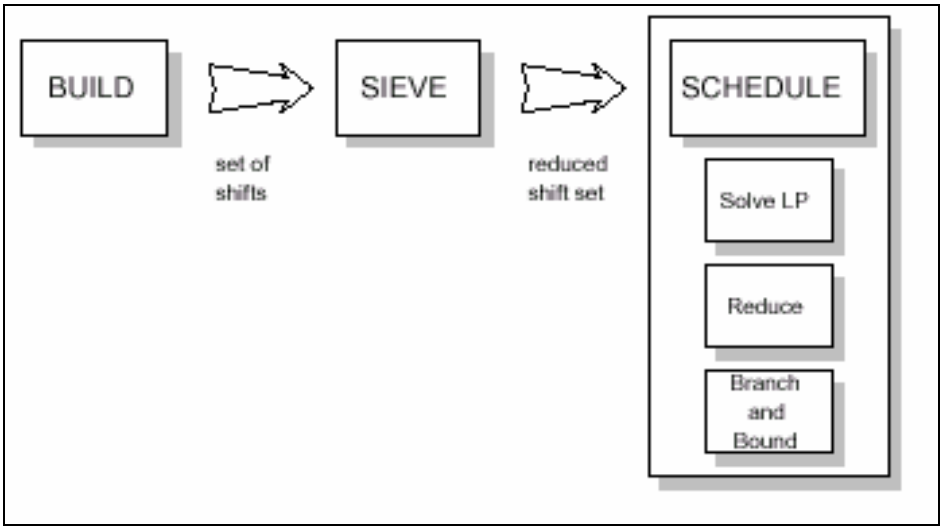

Figura 2.2: Modelo do TRACS II CURTIS (2000)**.**

Sendo assim, o programa BUILD gera um conjunto de jornadas válidas. FORES *et al.* (2000) afirmam que a geração de jornadas antecipadamente apresenta a vantagem de que conjuntos diferentes de jornadas podem ser formados usando diferentes combinações de restrições e podem até ser combinados antes da otimização. Além disso, pode-se identificar rapidamente problemas relacionados com dados ou parâmetros fornecidos pelo usuário através das evidências que surgem baseadas na formação, por exemplo, de altos ou baixos números de jornadas, ou restos de pedaços de trabalho descobertos.

Porém, CURTIS (2000) adverte que uma vez que só são geradas apenas jornadas válidas, ou "boas", a omissão de algumas jornadas importantes pode ser um fator negativo desse processo e prejudicar a eficiência da solução final. Segundo CURTIS (2000) a qualidade da jornada dependerá do quão bem ela combina com as outras jornadas, e esta jornada não poderá ser descartada até que o problema seja resolvido.

O tamanho desse conjunto de jornadas gerado pelo programa BUILD pode, em algumas vezes, ser muito grande. Sendo assim, o programa SIEVE é utilizado para reduzir esse conjunto. Segundo AZEVEDO FILHO (1997) o princípio para a redução considera que a jornada, cujos pedaços de trabalho são altamente cobertos por outras jornadas, tende a ser uma jornada de menor importância. Um sistema de *ranking* é usado considerando três atributos: um índice de custo eficácia, um número mínimo e um número médio de jornadas cobrindo os pedaços de trabalhos individuais daquela jornada escolhida para análise. Jornadas com o menor *rank* são eliminadas. O valor que estabelece se mais ou menos jornadas são eliminadas é determinado pelo usuário.

Após essas duas etapas, o sistema TRACS II irá realizar a seleção do sub-conjunto de jornadas usadas para formar a programação final. O programa SCHEDULE forma a programação mais econômica, sendo designado a produzir a melhor solução que cubra todo o trabalho dos veículos e que atenda todas as restrições. Conforme CURTIS (2000) o limite superior para o tamanho do problema que pode ser tratado pelo TRACS II é de 53297 jornadas potenciais, 976 trechos de trabalho e 195 jornadas na solução final.

Por fim, CURTIS (2000) lembra que há um problema com os modelos de *set partitioning/covering* que não é considerado pelo TRACS II. Trata-se das janelas de oportunidades de rendição. Freqüentemente os veículos chegam em um ponto de rendição e permanecem por vários minutos parados até que voltem à operação, variando consideravelmente esse tempo de parada. Sob alguns acordos de operação, uma mudança de motorista pode ser feita em algum instante entre a chegada e a saída do veículo. Entretanto, em um modelo *set partitioning/covering* é necessário um ponto específico para criar trechos de trabalho. O tamanho do problema aumentaria para uma proporção inaceitável se fosse necessário estabelecer uma oportunidade de rendição (OR) para todos os minutos em que o veículo está parado. Atualmente, a OR que é levada em conta é a que corresponde ao tempo de chegada do veículo no ponto de rendição. Infelizmente, devido aos acordos trabalhistas, isto pode significar que trechos de trabalho que poderiam teoricamente ser permitidos não são gerados. Isto se deve ao fato de que esses trechos poderiam afetar algumas questões tais como o tempo máximo antes de um intervalo.

2.2.5. Outras Aplicações

Nos últimos anos, muitas aplicações vêm sendo apresentadas com a finalidade de aperfeiçoar, ou até mesmo criar métodos de solução alternativos para alguns componentes ou etapas do sistema TRACS II, como pode ser visto em FORES *et al.* (1997), LAYFIELD *et al.* (1998), entre outros. Além disso, outros trabalhos atentam para a programação integrada de ônibus e tripulação (motorista e cobrador, no caso do Brasil) como é o caso de FRELING *et al.* (2000) e WREN e GUALDA (1997).

Por exemplo, em LAYFIELD *et al.* (1998) é criado um procedimento computacional de seleção de pontos de rendição que é utilizado como uma etapa de pré-programação antes da utilização do TRACS II. A partir desse procedimento como dado de entrada, o sistema TRACS II apenas irá selecionar um sub-conjunto de oportunidades de rendição que são viáveis para serem usadas. LAYFIELD *et al.* (1998) também afirmam que essa redução no número de oportunidades de rendição diminui, em até 75 %, o tempo gasto pelo TRACS II para produzir uma programação em alguns casos.

O modelo computacional proposto por este trabalho utiliza uma metodologia semelhante à abordada por LAYFIELD *et al.* (1998) no que se refere à seleção de oportunidades de rendição. Portanto, essa metodologia será abordada com mais detalhe no capítulo 4.

Segundo FORES *et al.* (1997) diferentes aperfeiçoamentos e métodos de solução alternativos têm sido incorporados ao componente de programação matemática do sistema TRACS II, incluindo uma técnica de geração de colunas que implicitamente considera muito mais jornadas válidas do que a abordagem de programação linear padrão. Todas essas modificações têm sido implementadas com a finalidade de permitir diferentes métodos de solucionar, quando necessário, esse tipo de problema e resolver grandes problemas em um único passo, como também produzir melhores resultados.

Outra metodologia que vêm sendo praticada recentemente é a programação integrada de veículos e tripulação (ou motoristas). FRELING *et al.* (2000) apresentam uma comparação entre a programação integrada e a programação seqüencial (é realizada a programação de veículos e posteriormente a de tripulação). É verificado em FRELING *et al.* (2000) que a programação integrada apresenta melhores resultados que a seqüencial. Em FRELING *et al.* (2000a) é realizado um estudo com três tipos de abordagens: i) abordagem seqüencial tradicional; ii) abordagem independente (as duas programações são resolvidas independentemente); e iii) abordagem integrada. Os autores afirmam que se não houver uma diferença muito grande entre os dois primeiros tipos de abordagem, não há a necessidade de integrar e o benefício pode ser medido comparando os custos das programações da tripulação para os dois casos.

Existem ainda aplicações que utilizam metaheurísticas para a resolução do problema de alocação de pessoal em empresas de transporte coletivo por ônibus. Porém, essas serão abordadas no capítulo 3 quando da revisão sobre metaheurísticas com ênfase para o *Simulated Annealing*.

Vale ressaltar que além de trabalhos para o modal ônibus, existem vários trabalhos desenvolvidos para a área de transporte ferroviário. Uma das aplicações em

transporte ferroviário bastante conhecida trata-se do trabalho realizado por CAPRARA *et al.* (1995) e CAPRARA *et al.* (1995a).

Segundo CAPRARA *et al.* (1995) em 1994, a companhia de trens italiana, *Ferrovie dello Stato SpA*, juntamente com a AIRO (*Italian Operational Research Society*), organizou duas competições entre os departamentos das universidades italianas para promover o desenvolvimento de heurísticas eficazes para os problemas de *Crew Scheduling* e *Rostering Problem* (trata da alocação de tripulação para um período bem mais abrangente do que um dia ou uma semana, considerando folgas semanais, férias, etc). A primeira competição, nomeada de FASTER (*Ferrovie Airo Set covering TendER*), requisitou o desenvolvimento de algoritmos para problemas de programação de tripulação de larga escala, envolvendo até 5.000 linhas e 1.000.000 colunas (CAPRARA *et al.* 1995a). A segunda competição, nomeada de FARO (*Ferrovie Airo Rostering Optimization*), requisitou algoritmos para solucionar o problema de *rostering* de forma a construir listas para a tripulação de trens, que é caracterizada por diferentes restrições operacionais. A função objetivo do problema foi do tipo hierárquica, no qual o objetivo mais importante foi minimizar o número de tripulações necessárias para executar as jornadas.

2.3. CONSIDERAÇÕES FINAIS

A partir da revisão bibliográfica sobre o problema de Alocação de Pessoal em empresas de transporte coletivo por ônibus e os métodos já desenvolvidos para a sua resolução pôde-se perceber a real importância que esse tipo de problema vem alcançando.

Porém, verificou-se que a maioria dessas aplicações não está sendo aplicada em grande escala pelas empresas que operam esse tipo de sistema. Acredita-se que isso se deve ao fato da grande complexidade do problema e da enorme dificuldade de implementar a heurística desenvolvida quando há uma modificação considerável na operação do sistema. Aliado a isso há ainda uma relutância, por parte de algumas empresas, em investir recursos para a aplicação e possível atualização desses procedimentos computacionais.

Em visita a algumas empresas do município de Fortaleza-CE percebeu-se a falta de credibilidade nos procedimentos, adotando assim uma postura de dúvida quanto ao sucesso de um determinado procedimento computacional.

Mesmo assim, nota-se em outros países a enorme utilização de vários modelos, como TRACS II, IMPACS, e a Programação Linear Inteira (PLI) na resolução desse tipo de problema. Além disso, verifica-se uma enorme quantidade de trabalhos comprometidos com a finalidade de aperfeiçoar e modificar aplicações já existentes, como forma de comparar resultados e metodologias.

CAPÍTULO 3

METAHEURÍSTICAS

3.1. INTRODUÇÃO

Segundo BLUM e ROLI (2001) o termo metaheurística deriva da composição de duas palavras gregas: heurística deriva do verbo *heurisken* que significa “encontrar”, enquanto o sufixo “meta” significa “além de, em um nível superior”.

As metaheurísticas têm sido alvo de muitos estudos nos últimos anos, inclusive para resolução do problema de programação de tripulação. Conforme VIANA (1998) “Heurística é qualquer método ou técnica criada, ou desenvolvida, para resolver um determinado tipo de problema. As Metaheurísticas são consideradas heurísticas de uso geral ou uma heurística das heurísticas”.

Segundo REEVES (1993) uma metaheurística é uma técnica que procura soluções boas (próximas da ótima) a um custo computacional razoável sem se tornar hábil para garantir a viabilidade ou o grau de otimalidade dessa solução, ou até em muitos casos a metaheurística pode dizer o quão próxima uma solução viável está da “solução ótima”. Isso para casos em que a solução ótima já seja conhecida para algumas instâncias do problema em análise. Outra forma de medir a qualidade da solução é compará-la com métodos que geram limites inferiores e/ou superiores para as instâncias dos problemas em estudo que não se conhece seus valores ótimos.

De forma mais detalhada OSMAN e LAPORTE (1996) *apud* BLUM e ROLI (2001) afirmam que a metaheurística é um processo de geração iterativa que guia uma heurística subordinada por combinar, inteligentemente, conceitos diferentes para investigar e explorar o espaço de busca, e utilizar estratégias de memorização para estruturar informações a fim de obter soluções eficientes e próximas do ótimo global.

Cabe aqui ressaltar que o termo “investigação” se refere à capacidade da metaheurística de saltar, a passos largos, de uma região para outra no espaço de busca. Já o termo “exploração” reflete a capacidade em explorar de forma mais intensa uma mesma região dentro do espaço de busca. Na verdade, o que existe são dois processos de busca, um externo (investigação) e outro interno (exploração). A investigação é comumente chamada de *diversificação*, enquanto a exploração é denominada de *intensificação*.

Segundo BLUM e ROLI (2001) as metaheurísticas apresentam as seguintes características:

a) guiam o processo de busca;
b) exploram, eficientemente, o espaço de busca a fim de encontrar soluções ótimas;
c) variam de procedimentos de busca local simples para processos de memorização complexos;
d) são aproximativas e não-determinísticas;
e) incorporam mecanismos para evitar com que a busca fique confinada em um determinado local do espaço de busca;
f) permitem, através do seu conceito básico, uma representação de nível abstrato;
g) não são específicas para um único tipo de problema; e,
h) fazem uso de conhecimento de domínio específico e/ou experiência de busca (memória) para influenciar a busca.

As metaheurísticas mais encontradas na literatura, e conseqüentemente as mais aplicadas são: *Simulated Annealing* (KIRKPATRICK, 1983), Busca Tabu (GLOVER, 1986), Algoritmos Genéticos (HOLLAND, 1975), *Ant Colony Optimization* - Otimização de Colônia de Formigas (DORIGO *et al.*, 1996 e DORIGO E STUTZLE, 2000), GRASP - *Greedy Randomized Adaptive Procedure* (FEO e RESENDE, 1995). Porém, existem outras como *Variable Neighborhood Search* - VNS (HANSEN e MLADENOVIC, 1997 *apud* GOLDBARG e LUNA, 2000), *Multi-Start* (FEO e RESENDE, 1989 *apud* GOLDBARG e LUNA, 2000), dentre outras.

3.2. METAHEURÍSTICAS MAIS UTILIZADAS

Nos próximos itens será feita uma revisão sobre algumas metaheurísticas mais encontradas na literatura, com ênfase para a denominada *Simulated Annealing* que foi a metaheurística escolhida para este trabalho.

3.2.1. Busca tabu

Busca Tabu é uma das metaheurísticas mais usadas em problema de otimização combinatória. Segundo HERTZ *et al.* (1992) a origem da Busca Tabu se deu na década de 70 e ela foi primeiramente apresentada em sua forma atual por GLOVER, 1986 *apud* HERTZ *et al.* (1992).

Segundo VIANA (1998) essa metaheurística apresenta três princípios: i) uso de uma estrutura de dados (lista) para "guardar" o histórico da evolução do processo de busca; ii) uso de um mecanismo de controle para fazer um balanceamento entre a aceitação, ou não, de uma nova configuração, com base nas informações registradas na "lista tabu" referentes às restrições e aspirações desejadas e iii) incorporação de procedimentos que alternam as estratégias de diversificação e intensificação.

A metaheurística Busca Tabu utiliza uma memória muito grande (lista tabu) onde são armazenados movimentos, ou atributos (componentes de uma solução, diferença entre duas soluções, etc), que serão proibidos de acontecerem por um tempo determinado. Esses movimentos podem ser em pontos específicos da solução ou na solução como um todo. Na Lista Tabu estão contidos os caminhos para as soluções visitadas recentemente e com isso proíbe movimentos que retornem a essa solução. Portanto, além de armazenar o valor da função objetivo de uma determinada solução, a Busca Tabu também armazena informações do itinerário com base nas últimas soluções visitadas.

Sendo assim, o espaço de vizinhança da solução corrente é restrito a soluções que não pertencem à lista tabu. A nova solução é então adicionada à lista tabu e uma das soluções contidas na lista é descartada. Devido a esta restrição dinâmica de permitir

soluções em uma vizinhança, a Busca Tabu pode ser considerada como uma técnica de vizinhança dinâmica (STUTZLE, 1999 *apud* BLUM e ROLI, 2001).

Segundo BLUM e ROLI (2001) o uso de uma lista tabu impede o retorno a soluções visitadas recentemente, portanto, ela previne ciclos indefinidos e força a busca a aceitar até movimentos que pioram a solução. O comprimento da lista tabu controla a memória do espaço de busca. Com lista pequena a busca se concentrará em pequenas áreas do espaço de busca. Caso contrário, uma lista grande força o processo de busca a explorar grandes regiões, porque ela proíbe revisitar um alto número de soluções. O comprimento da lista tabu pode ser variado durante a busca, levando a algoritmos mais robustos.

De acordo com BRAGA *et al.* (1994) é necessário que a Lista Tabu tenha um comprimento definido, pois poderá haver momentos em que seja necessário voltar para alguma solução, e a partir dela, buscar outras soluções. Além do comprimento da Lista Tabu, VIANA (1998) afirma que o tamanho da vizinhança é outro parâmetro importante para o uso da Busca Tabu. A escolha do tamanho da vizinhança regula o tempo de execução do algoritmo. Uma má escolha desse parâmetro pode comprometer a execução do algoritmo; assim este valor deve ser ajustado, juntamente com o número máximo de iterações, para se obter bons resultados num tempo aceitável.

As estruturas de memória no Busca Tabu apresentam quatro princípios:

a) Recenticidade: se a solução foi visitada há pouco tempo ou não.
b) Freqüência: quantas vezes a solução ou atributo foi visitado.
c) Qualidade: habilidade de diferenciar o mérito de uma solução durante a busca. Pode ser medida tanto em relação à solução total como em relação a partes da solução.
d) Influência: influência de uma determinada solução, ou determinado atributo, em relação às próximas soluções, ou aos próximos atributos.

Vale ressaltar que, ao proibir um número muito grande de soluções, o desempenho do algoritmo pode ficar prejudicado. Portanto, pode ser interessante, em

certas ocasiões, permitir que uma determinada solução, ou atributo saia da Lista Tabu. O critério para se remover uma solução ou atributo da Lista Tabu é chamado de *Critério de Aspiração*.

BLUM e ROLI (2001) citam que um dos critérios de aspiração mais comumente usado é a seleção de soluções que são melhores do que a solução corrente. Segundo REEVES (1993) o uso apropriado de tal critério pode ser muito importante para capacitar o método Busca Tabu para alcançar seus níveis de melhor performance.

Por último, vários critérios de parada podem ser adotados para o processo iterativo. O processo pode parar após um número *K* de iterações forem atingidos, sem que ocorra alguma melhora na solução. Outro critério de parada pode ser quando as soluções vizinhas a uma solução corrente são todas tabus.

Segundo GLOVER e LAGUNA (s.d) a metaheurística Busca Tabu pode ser aplicada em diversas áreas, como por exemplo: programação, locação e alocação, roteamento, telecomunicações, tecnologia, e outras mais.

Uma das aplicações, para o problema de programação de motoristas, é o trabalho reportado por SHEN e KWAN (2000) onde é desenvolvido um sistema automático de programação de motoristas chamado HACS. O sistema HACS parte de uma solução inicial que não necessariamente precisa ser uma solução viável. Para modificar essa solução inicial são designadas quatro estruturas de vizinhança de forma a diversificar a busca de vizinhança. SHEN e KWAN (2000) afirmam que o algoritmo é mais rápido do que aqueles baseados em programação matemática e que o HACS se mostra bastante satisfatório para tratar de problemas grandes gerando resultados comparáveis com os melhores resultados já conhecidos.

3.2.2. Algoritmos Genéticos

Algoritmos Genéticos (AGs) são metaheurísticas baseadas na teoria da seleção natural proposta por Charles Darwin, na qual, sob muitas gerações, as populações evoluem de acordo com a lei da seleção natural. Ou seja, o indivíduo mais forte é que

sobrevive. Sendo assim, os AGs incorporam o conceito de cruzamento (*crossover*), mutação, população, pais, filhos, clonagem e outros mais.

Por analogia, a população é o conjunto de soluções. Pode-se verificar aqui que os AGs não partem de uma única solução inicial, como é visto em algumas outras metaheurísticas, e sim de uma população de indivíduos gerados, aleatoriamente ou não, na qual cada indivíduo sofre uma avaliação (aplicação da função objetivo). A partir dessa avaliação somente os melhores indivíduos terão a chance de sofrerem manipulações genéticas como cruzamento (*crossover*), mutação, ou até mesmo a clonagem. A clonagem pode ser utilizada em casos onde seja necessário copiar um indivíduo de alta qualidade, fazendo com que o mesmo permaneça na população na geração seguinte.

Além dos conceitos de *crossover*, mutação, clonagem, população e indivíduos (alguns destes serão posteriormente detalhados) existem outros termos que também são abordados, de forma análoga, pelos AGs. Os termos usados nos sistemas naturais têm as seguintes correspondências: *cromossomos* (representação da solução ou indivíduo); *gens* (representa uma certa característica da solução); *alelo* (um particular estado de um gene); *lócus* (posição de um gene em um cromossomo); *genótipo* (estrutura) e *fenótipo* (“aparência”, forma, ou modelo dos indivíduos dominantes). Segundo CORRÊA (2000) o genótipo é a variável independente da função objetivo *f(x)*, e o fenótipo é a variável dependente ou valor da função objetivo.

VIANA (1998) também acrescenta o conceito de *schema* ou *máscara*, que é um modelo de representação dos cromossomos parecidos. Um *schema* é construído com a inclusão de um símbolo especial (*) no alfabeto de *gens*. Por exemplo, para cromossomos representados por gens binários (0 ou 1) o *schema* (*10*1) representa quatro configurações: (0100**1**), (0101**1**), (1100**1**) e (1101**1**). A Figura 3.1, como mostra CORRÊA (2000), apresenta de forma mais detalhada esses conceitos citados anteriormente.

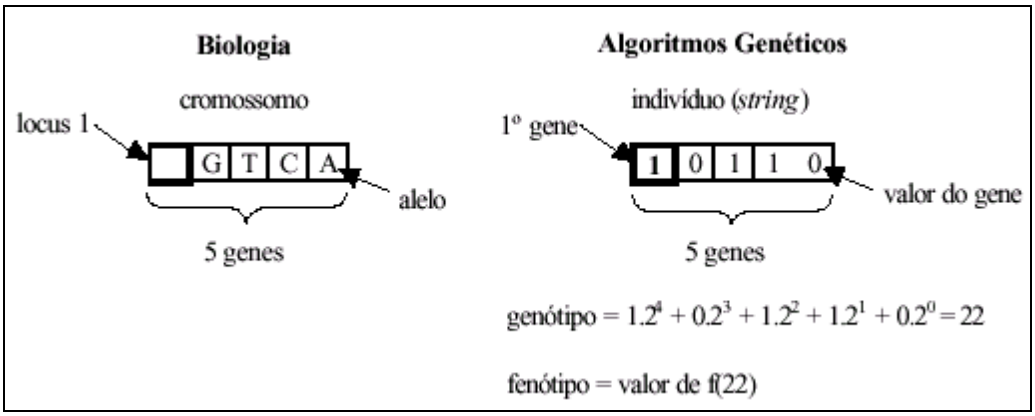

Figura 3.1: Representação de um cromossomo na Biologia e em AGs.

Segundo BEASLEY *et al.* (2001) os princípios básicos dos AGs foram primeiro estabelecidos rigorosamente por HOLLAND (1975). Porém, REEVES (1993) ressalta que as primeiras aplicações foram no campo da Inteligência Artificial. Algumas abordagens foram feitas em otimização de funções, mas só recentemente é que aplicações para problemas de interesse para a comunidade de Pesquisa Operacional têm sido mais extensamente reportado.

Conforme VIANA (1998) os AGs apresentam as seguintes características:

a) usam técnicas de randomização;
b) são robustos (abrangentes);
c) trabalham unicamente com o valor da função objetivo;
d) utilizam somente regras probabilísticas (não determinísticas); e
e) são de uso geral (metaheurísticas).

De acordo com BARCELOS (2000) o AG pode ser descrito em seis passos:

a) inicie uma população de tamanho *N*, com cromossomos gerados aleatoriamente ou não;
b) aplique a função de adequação (ou função objetivo) a cada cromossomo dessa população;
c) crie novos cromossomos através de combinação entre os cromossomos selecionados dessa população. Execute recombinação e mutação nestes cromossomos;

d) elimine membros da antiga população de forma a obter espaço para introduzir esses novos cromossomos, mantendo a população com o mesmo tamanho *N*;
e) aplique a função de adequação a esses novos cromossomos e insira-os na população;
f) se a solução ideal for encontrada, ou se o tempo (ou número de gerações) se esgotou, retorne o melhor cromossomo encontrado. Caso contrário, volte ao passo (c).

Para um maior entendimento dos processos realizados pelo AG faz-se necessário uma maior explanação sobre os operadores genéticos (*crossover* e mutação) e sobre a configuração de alguns parâmetros importantes para uma boa performance dos AGs.

- **Cruzamento (*Crossover*)**

Segundo VIANA (1998) o *crossover* é a operação que possibilita a recombinação das estruturas genéticas da população. Permite uma diversificação no espaço de configurações ao gerar estruturas diferentes das atuais. O operador "*crossover*" escolhe aleatoriamente dois cromossomos pais e troca partes de seu padrão genético.

De acordo com BARROS NETO (1997) o processo de *crossover* ou recombinação envolve cortes que serão efetuados nos cromossomos pais que, então, serão trocados segundo uma regra pré-definida. Os cortes podem ser fixos para todos os cromossomos ou aleatórios.

Existem, basicamente, dois tipos de *crossover*: *crossover* em um ou mais pontos do cromossomo e *crossover* uniforme. Além disso, o operador *crossover* é aplicado aos indivíduos da população com uma probabilidade constante, geralmente entre 0,5 e 0,8. A Figura 3.2 ilustra o *crossover* uniforme, no qual cada gene do descendente é escolhido aleatoriamente dos correspondentes genes dos pais

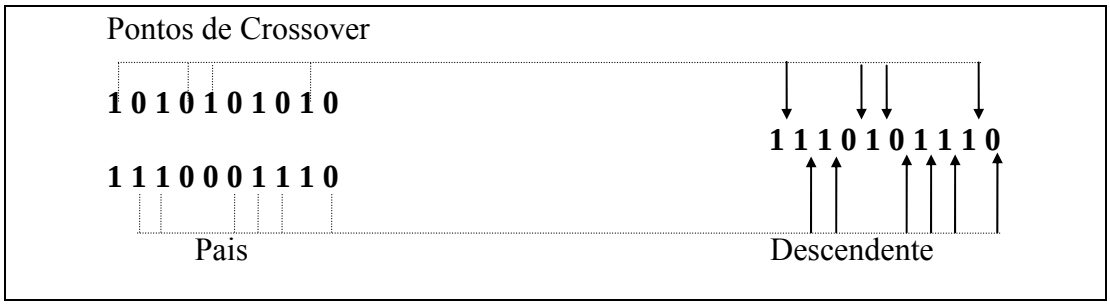

Figura 3.2: *Crossover* uniforme (BARROS NETO, 1997).

BUSSETI (2001) afirma que, entre os vários tipos de *crossover*, o do tipo uniforme parece ser mais robusto. Semelhantemente, SYSWERDA (1989) *apud* BEASLEY *et al.* (1993) argumenta em favor do *crossover* uniforme. Quando dois cromossomos pais são semelhantes, os segmentos modificados pelo *crossover* em dois pontos são prováveis de serem idênticos, fazendo com que os descendentes sejam idênticos aos pais. Com o *crossover* uniforme há uma menor probabilidade disso acontecer. *Crossover* uniforme tem a vantagem de que a ordem dos genes é totalmente irrelevante. Isto significa que operadores de reordenação são desnecessários, excluindo assim a preocupação sobre o posicionamento dos genes.

- **Mutação**

VIANA (1998) afirma que, "esta operação corresponde a uma pequena perturbação numa configuração que tem por objetivo tentar regenerar algum indivíduo que, por 'culpa' da lei da probabilidade, foi eliminado de forma inesperada".

De acordo com BRAZ JUNIOR (2000) o processo de mutação é equivalente a busca aleatória. Seleciona-se uma posição no cromossomo e muda-se o valor do gene aleatoriamente para um outro alelo possível.

A mutação também acontece seguindo uma probabilidade. Porém, essa probabilidade é menor do que a especificada para o *crossover* visto que, por analogia ao processo de evolução natural, a mutação tem um caráter negativo em relação às espécies. É sempre vista como um fator anormal da natureza. Essa probabilidade é de no máximo 0,2. A Figura 3.3 a seguir ilustra uma operação de mutação.

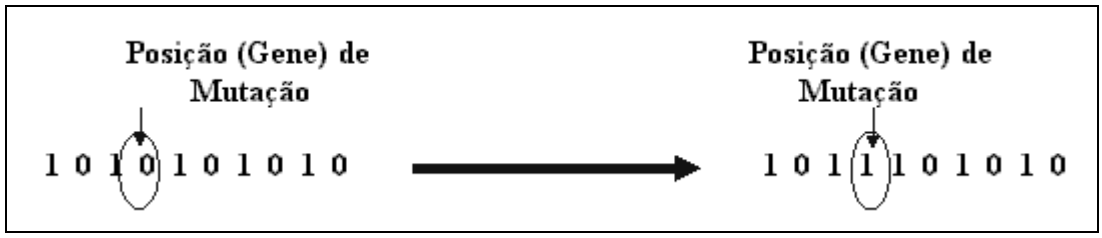

Figura 3.3: Mutação (BARROS NETO, 1997).

Além dos operadores genéticos (mutação e *crossover*), a boa escolha de outros parâmetros tem bastante importância para que se consiga uma boa performance do AG. São eles: tamanho da população, representação do cromossomo (ou codificação), critério de parada, e a determinação da função objetivo (ou função de *fitness*).

Algumas aplicações de AGs no problema de programação de tripulação podem ser encontradas em WREN *et al.* (2000), KWAN *et al.* (1997), entre outros.

3.2.3. *Simulated Annealing*

Segundo SILVA *et al.* (2002) *Simulated Annealing* foi proposta originalmente por KIRKPATRICK *et al.* (1983). Trata-se de uma técnica de busca local probabilística, que se baseia em uma analogia com a termodinâmica, ao simular o resfriamento lento da matéria, após ser aquecida (METROPOLIS *et al*., 1953).

Conforme BUSSETI (2001a) *Simulated Annealing* (SA) é uma técnica de busca aleatória que investiga uma analogia entre a forma em que um metal esfria e congela em uma estrutura cristalina de energia mínima (o processo de recozimento), e a busca para um mínimo em um sistema mais geral.

A analogia com o processo termodinâmico é bastante direta. A função objetivo substitui a energia, os diversos estados da matéria substituem as possíveis soluções, os estados metaestáveis da matéria seriam os ótimos locais, e a estrutura cristalina (considerando um cristal sendo levado a sua temperatura de fusão) sendo o ótimo global. O parâmetro mais importante, e que regula o processo do SA, é a chamada temperatura. O valor da temperatura inicial e o fator de decrescimento dessa

temperatura (resfriamento) são fatores muito importantes para uma boa performance do *Simulated Annealing*.

NORONHA (2001) ilustra na Figura 3.4 a formulação geral do SA. Este algoritmo se decompõe em duas grandes buscas sobrepostas. A busca externa controla o término do processo e é baseada na noção de estado resfriado. A busca interna contém o processo de otimização. A variável s^* representa a melhor solução encontrada durante a execução do algoritmo.

```
Algoritmo Metropolis
-Início
1. Estabelecer uma solução realizável s^0;
2. s ← s^0;
3. Escolher uma temperatura inicial T > 0;
4. Enquanto o sistema não esteja resfriado Fazer
        Efetuar L iterações de:
        Escolher aleatoriamente um vizinho v ∈ v(s)
        Δ = objetivo(v) – objetivo(s)
        Se Δ < 0 então
            s ← v;
        Se objetivo(v) < objetivo(s*) então
            s* ← v;
        Senão
        s ← v com probabilidade e^(−Δ/T);
        Fim da busca interna;
    Reduzir a temperatura;
  Fim do Enquanto
-Fim
```

Figura 3.4: Formulação geral do *Simulated Annealing*.

Segundo BLUM e ROLI (2001) o algoritmo começa por gerar uma solução inicial (construída aleatoriamente ou heuristicamente) e por inicializar a temperatura. Então ele repete o processo de busca até alcançar algum critério de parada. Diferentes critérios de parada podem ser adotados: tempo máximo de processamento, número máximo de iterações, quando é encontrada uma solução *s* com *f(s)* menor que um valor predefinido, ou um número máximo de iterações sem alcançar um melhoramento.

O SA também aceita movimentos que piorem a solução. Esses movimentos não são freqüentes e acontecem segundo uma probabilidade, que é apresentada por

REEVES (1993) e mostrada na Equação 3.1. Onde δE é a diferença entre os valores da função objetivo, k é o número da iteração corrente e t é a temperatura atual.

$$p(\delta E) = \exp\left(-\delta E / kt\right) \tag{3.1}$$

Portanto, é gerado um número aleatoriamente e se o mesmo for menor do que a probabilidade $p(\delta E)$, a solução é aceita. Com isso, é possível piorar a solução para que se possa alcançar outras regiões dentro do espaço de vizinhança e assim chegar a uma solução de boa qualidade (mais próxima possível da solução ótima global). Fica claro, através da Equação 3.1, que essa probabilidade de aceitação depende muito da temperatura e da diferença entre os valores da função objetivo das duas soluções que estão sendo comparadas.

Assim quanto maior a temperatura, maior a probabilidade de aceitação de uma solução pior do que a solução corrente. Já para uma temperatura fixa, quanto maior a diferença entre os valores da função objetivo, menor a probabilidade de aceitar movimentos que degradam a solução.

Outro fator importante nessa metaheurística é o processo de resfriamento. Segundo BLUM e ROLI (2001) programações de resfriamento que garantem a convergência para um ótimo global infelizmente são impraticáveis, pois necessitam de um tempo infinito para se atingir o ótimo global. Entretanto, programações de resfriamento mais rápidas são adotadas. Uma das mais usadas, segue uma lei geométrica: $T_{k+1} = \alpha\ T_k$, no qual k se refere à iteração e $\alpha \in\ ]0, 1[$, que corresponde a um decréscimo exponencial da temperatura. Geralmente o valor de α oscila entre 0,90 e 0,99.

BLUM e ROLI (2001) também acrescentam que a regra de resfriamento pode variar durante a busca, com o objetivo de ajustar o balanço entre diversificação e intensificação. Por exemplo, no início da busca, a temperatura T pode ser constante ou decrescer linearmente, favorecendo assim mais a diversificação. Então, T pode seguir uma regra, como a geométrica, para convergir para um mínimo local no fim da busca, ou seja, intensificado mais do que diversificando.

Existe também uma técnica, comumente usada, no processo do SA chamado de *Re-Annealing*. Esse processo nada mais é do que promover um reaquecimento no processo. Ou seja, o sistema é novamente aquecido. Essa técnica pode ser utilizada em casos quando é verificado que a solução está permanecendo por muitas iterações sem ser melhorada. Portanto, é promovido um reaquecimento no sistema com a finalidade de escapar de um possível ótimo local, para que se possa atingir outras regiões e assim voltar a melhorar a solução.

Com isso, a metaheurística *Simulated Annealing* apresenta as vantagens e desvantagens descritas a seguir. As vantagens se referem à mesma apresentar uma implementação simples, pois só visita uma única solução a cada iteração bastando calcular o valor da função objetivo da solução vizinha gerada. Além disso, *Simulated Annealing* pode lidar com modelos altamente não-lineares, dados caóticos e muitas restrições. Ela é uma técnica robusta em geral (BUSSETI, 2001a).

BUSSETI (2001a) afirma que, “Sua principal vantagem sobre os outros métodos é a sua habilidade e flexibilidade para se aproximar do ótimo global. O algoritmo é bastante versátil desde que ele não dependa de alguma propriedade restritiva do modelo”.

BUSSETI (2001a) também afirma que os métodos SA apresentam a vantagem de serem facilmente “ajustados”, pois para qualquer sistema estocástico ou não-linear razoavelmente difícil, um dado algoritmo de otimização pode ser ajustado para aumentar sua performance, requerendo, subseqüentemente, tempo e esforço para se tornar familiar com um dado código. A habilidade para ajustar um dado algoritmo, para usar em mais de um problema, deve ser considerada uma característica importante de um algoritmo.

Porém, o SA apresenta algumas desvantagens:

a) apesar de convergir para a solução ótima, a velocidade de redução de temperatura exigida implica em visitar um número exponencial de soluções;

b) a princípio é necessário um processo lento de resfriamento e isso resulta em tempos elevados de processamento;
c) é pouco inteligente, pois só usa a variação do valor da função objetivo como informação do problema;

No sub-item a seguir é apresentada uma experiência nacional da aplicação da metaheurística *Simulated Annealing* no problema de alocação de pessoal em empresas de transporte coletivo por ônibus.

- **Aplicações com o uso do SA**

Uma das aplicações mais recentes é a de SILVA *et al.* (2002). Essa aplicação foi utilizada para o transporte público da cidade de Belo Horizonte-MG. O processo segue os seguintes passos:

a) *Passo 1*: gera as tarefas tomando como base a programação de veículos;
b) *Passo 2*: obtém a solução inicial estimando-se um número de tripulações (muito maior do que o necessário);
c) *Passo 3*: minimiza a função objetivo com o uso do SA;
d) *Passo 4*: se necessário, aumenta o número de tripulações. Caso a melhor solução ainda contenha inviabilidades;
e) *Passo 5*: o programa pára quando as inviabilidades forem eliminadas. Caso a solução não seja satisfatória, o sistema é reaquecido e volta ao Passo (3).

Em SILVA *et al.* (2002) a função objetivo apresenta três metas:

a) Eliminação das inviabilidades: procura eliminar completamente todas as inviabilidades da programação de tripulação (sobreposição de tarefas, horas-extras, folga acumulada deficiente e troca de pontos proibida).
b) Redução de custos: tem por finalidade minimizar os custos operacionais da programação corrente. São considerados os seguintes custos: duplas pegadas, n-uplas pegadas, horas-extras, troca de pontos permitida, troca de veículos, e número de tripulantes.

c) Maximização da utilização da mão-de-obra: tem a finalidade de aproveitar, ao máximo, o tempo de cada tripulação, minimizando a ociosidade.

Conforme SILVA *et al.* (2002) o algoritmo trabalha com o movimento *Inserir Tarefa*, que consiste em atribuir uma tarefa de uma tripulação *i* a uma tripulação *j*. São sorteadas duas tripulações de forma aleatória, *i* e *j*, com a condição que a tripulação *i* tenha pelo menos uma tarefa, e então se escolhe, também de forma aleatória, uma tarefa da tripulação *i* sobre a qual se faz o movimento.

SILVA *et al.* (2002) afirmam que "apesar das dificuldades inerentes à abordagem do problema de programação de tripulação, os resultados obtidos mostraram uma significativa redução nos custos referentes à mão-de-obra operacional da programação diária".

3.2.4. *Ant colony optimization* (ACO)

A otimização através de colônia de formigas é uma metaheurística proposta por DORIGO (1992) *apud* BLUM e ROLI (2001) e posteriormente por DORIGO *et al.* (1996 e 1999) *apud* BLUM e ROLI (2001).

Segundo DORIGO e DI CARO (1999) algoritmos desse tipo têm sido inspirados pelas experiências realizadas por GOSS *et al.* (1989) usando uma colônia de formigas reais. A experiência consistiu em construir dois caminhos de comprimentos diferentes entre uma colônia de formigas e uma fonte de comida. Os caminhos foram arranjados de modo que as formigas pudessem escolher ambos os caminhos, tanto no sentido de ida como no de volta, conforme a Figura 3.5.

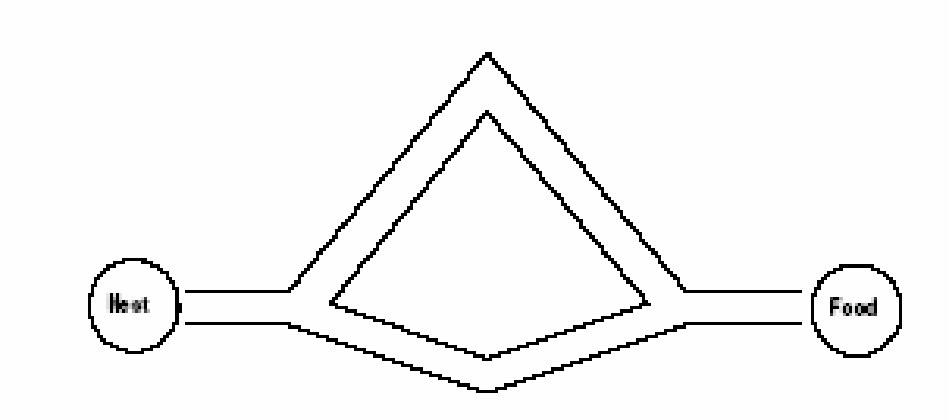

Figura 3.5: Aparelho utilizado por GLOSS (DORIGO e DI CARO, 1999).

Sendo assim, observou-se que após uma fase transitória (poucos minutos após o início do experimento) a maioria das formigas usa o caminho mais curto. Observou-se também que a probabilidade da colônia escolher o caminho mais curto aumenta diretamente proporcional a diferença entre os dois caminhos. O aparecimento desse comportamento, segundo DORIGO e DI CARO (1999), pode ser explicado em termos de autocatálise (*feedback* positivo) e comprimento diferencial de caminho, e ele é possível de ser feito através de uma forma indireta de comunicação mediada por modificações locais do meio ambiente.

As formigas, enquanto caminham da colônia para a comida e vice-versa, depositam no chão uma substância química chamada de *pheromone*. Quando as formigas chegam em um ponto de decisão, como a intersecção entre os dois caminhos, elas fazem uma escolha probabilística baseada na quantidade dessa substância que é cheirada pelas formigas. Portanto, à medida que a formiga escolhe um determinado caminho, a quantidade de *pheromone* naquele caminho vai aumentando, aumentando assim a probabilidade desse mesmo caminho ser escolhido futuramente.

Por analogia, segundo MANIEZZO e CARBONARO (1999) uma formiga é definida como um agente computacional simples, que constrói, iterativamente, uma solução para o problema a ser resolvido. Soluções parciais do problema são vistas como *estado*; cada formiga se move de um *estado n* para outro *estado p*, correspondendo a uma solução parcial mais completa. Em cada passo *j*, cada formiga *k* armazena um

conjunto de expansões viáveis para seu estado corrente, e se move para um dessas expansões de acordo com uma probabilidade.

Conforme MANIEZZO e CARBONARO (1999) a probabilidade da formiga mover-se de um estado *n* para um *p* depende da combinação de dois valores:

a) a atratividade do movimento, que é computado por algumas heurísticas indicando, a priori, o desejo de se fazer aquele movimento;
b) a importância do caminho (o valor da trilha) a ser seguido, indicando quão proficiente ele foi no passado para realizar aquele movimento particular. Entretanto, ele representa uma indicação, a posteriori, do desejo daquele movimento.

As trilhas são atualizadas a cada iteração, aumentando o nível daquelas que facilitam os movimentos para partes de soluções "boas", e diminuindo o nível das outras trilhas.

Uma aplicação simples dessa metaheurística é apresentada, de forma mais detalhada, por DORIGO e DI CARO (1999) na resolução de um problema de caminhos mínimos. Basicamente, a principal tarefa de cada formiga artificial é encontrar o menor caminho entre um par de nós em um grafo. Para cada arco (*i,j*) do grafo é associada uma variável chamada de *pheromone* artificial. A quantidade de *pheromone* em cada arco é proporcional à utilidade, como estimado pelas formigas, de usar aquele arco para construir boas soluções. Dessa forma, em cada nó, ou nos arcos adjacentes, são armazenadas informações necessárias para que a formiga selecione o próximo nó a visitar, construindo assim a solução.

A metaheurística *Ant Colony Optimization* (ACO) também é aplicada em problemas de alocação quadrática, caixeiro viajante, roteamento de veículos, coloração de grafos, alocação de tripulação, etc.

Uma das aplicações desse tipo de metaheurística na área de alocação de motoristas é o trabalho reportado por FORSYTH e WREN (1997). É desenvolvido um

sistema de formigas artificiais em que cada formiga segue uma trilha através de uma rede. Cada trilha representa uma programação completa de motoristas de ônibus. Programações iniciais são determinadas pela retirada de turnos gerados por outros sistemas disponíveis na Universidade de Leeds. O sistema proposto utiliza o processo BUILD do TRACS II para gerar um grande número de jornadas viáveis que será reduzido pelo sistema. Efetivamente, o sistema proposto pelos autores substitui o componente SCHEDULE (existente no TRACS II) por dinamizar sua habilidade em reduzir o número total de jornadas.

3.2.5. *Greedy Randomized Adaptive Search Procedures* (GRASP)

Segundo RESENDE e RIBEIRO (2001) a metaheurística GRASP é um processo iterativo em que cada iteração consiste de duas fases: construção e busca local. A fase de construção produz uma solução viável, cuja vizinhança é investigada até um mínimo local ter sido encontrado durante a fase de busca local. A melhor solução global é mantida como resultado.

Na fase de construção uma solução viável é iterativamente construída, um elemento de cada vez. A fase de construção do GRASP básico é similar à heurística semi-gulosa proposta independentemente por HART e SHOGAN (1987) *apud* RESENDE (1998). Em cada iteração construtiva, a escolha do próximo elemento a ser adicionado é determinada pelo ordenamento de todos elementos candidatos (ou seja, aqueles que podem ser adicionados à solução) em uma lista C de candidatos.

Conforme RESENDE e RIBEIRO (2001) as soluções geradas pela fase de construção não são necessariamente ótimas, até mesmo em relação a simples vizinhanças. A fase de busca local usualmente melhora a solução construída. O algoritmo de busca local trabalha em uma forma iterativa por substituir sucessivamente a solução corrente por uma solução melhor na vizinhança da solução corrente. O processo termina quando nenhuma solução melhor é encontrada na vizinhança.

RESENDE (1998) afirma que a metaheurística é adaptativa porque os benefícios associados com todos os elementos são atualizados em cada iteração da fase de

construção para refletir as modificações influenciadas pela escolha do elemento anterior. Além disso, uma característica especialmente atraente do GRASP é a facilidade com que ele pode ser implementado, visto que poucos parâmetros necessitam ser fixados e ajustados.

A metaheurística GRASP é aplicada na solução de diversos problemas, como por exemplo: sistemas de manufatura flexível, programação de linhas aéreas, produção e planejamento programado auxiliado por computador, problemas de localização, coloração em grafos, sistemas de planejamento de transmissão de potência, roteamento de veículos, etc, (NORONHA *et al.*, 2001).

3.3. CONSIDERAÇÕES FINAIS

Embora exista diferença entre as diversas metodologias e filosofias empregadas em cada metaheurística (alguma são baseadas em população, como AGs e *Ant Colonies*, e outras em métodos de trajetória, como SA, Busca Tabu, GRASP, etc), os mecanismos de explorar um espaço de busca são baseados em intensificação e diversificação, (BLUM e ROLI, 2001).

No *Simulated Annealing* a intensificação e a diversificação são controladas pela temperatura. No início, como há uma probabilidade maior de aceitar soluções que piorem a função objetivo, a diversificação é maior do que a intensificação. À medida que o sistema vai resfriando ocorre o inverso, a intensificação passa a ser maior do que a diversificação.

No caso da Busca Tabu, a Lista Tabu é quem controla os processos de intensificação e diversificação. O balanço entre esses dois processos é determinado pelo tamanho da Lista Tabu. Listas grandes resultam em uma menor intensificação e uma maior diversificação, visto que existirá um maior número de soluções ou atributos que não poderão ser visitados por um certo número de iterações. Uma maior intensificação pode ser conseguida através do critério de aspiração. Além do tamanho da Lista Tabu, a freqüência em que uma determinada solução é visitada, ou um determinado movimento é realizado, pode determinar o balanço entre diversificação e intensificação. Uma alta

freqüência em soluções consideradas de “elite” (boa qualidade) resultaria em uma intensificação. Caso contrário tem-se uma diversificação.

Os operadores de seleção de indivíduos são os controladores dos mecanismos de intensificação e diversificação, no caso dos Algoritmos Genéticos. A diversificação é alcançada através dos operadores de recombinação e cruzamento. O balanço entre diversificação e intensificação muda de alta diversificação e baixa intensificação para baixa diversificação e alta intensificação, à medida que a diversidade da população vai decrescendo.

Por fim, verificou-se que as metaheurísticas não são facilmente influenciadas pelo tipo de problema. Como são heurísticas mais gerais, qualquer modificação nas características do problema não irá afetar tanto o processo de otimização e, portanto não será necessário efetuar mudanças consideráveis no modelo. Nas aplicações que utilizam heurísticas ou programação linear inteira isso não é verificado.

CAPÍTULO 4

PROCEDIMENTO COMPUTACIONAL DESENVOLVIDO E ESTUDO DE CASO

4.1. INTRODUÇÃO

Para a realização do programa proposto neste trabalho, denominado de ATTON (Alocação de Tripulação em Transporte coletivo por Ônibus) foi utilizada a linguagem de programação Pascal através da ferramenta *Delphi* 6.0. Mesmo tendo conhecimento de uma versão mais atual (*Delphi* 7.0), optou-se pela versão 6.0 em face da não disponibilidade da versão mais utilizada.

A seguir serão apresentadas, de forma detalhada, todas as etapas criadas para a realização do procedimento computacional. São elas: Dados de Entrada, Restrições, Solução Inicial, Processo *Simulated Annealing*, Solução Final e Apresentação dos Resultados. Além disso, ao final desse capítulo será apresentada uma aplicação para uma linha de ônibus de uma empresa do município de Fortaleza-CE. Dessa forma, será possível melhor avaliar o procedimento computacional proposto.

4.2. DADOS DE ENTRADA E RESTRIÇÕES

Inicialmente, é importante caracterizar a operação do sistema de transporte coletivo por ônibus no município de Fortaleza. Isso se deve ao fato do programa ATTON incorporar algumas características relacionadas ao modo de operação por parte das empresas responsáveis por esse tipo de transporte.

Como mencionado no capítulo 2, a programação de veículos da linha de ônibus é realizada pela ETTUSA que envia essa programação a empresa para que a mesma seja cumprida. A fiscalização é realizada através de postos de controle onde são especificados horários em que o veículo terá que passar. Todas essas informações são registradas e enviadas para as empresas através de um documento chamado de Ordem de Serviço Operacional – OSO, como mostrado no Anexo 1.

Na OSO são especificados os horários de entrada e de saída do ônibus na linha, os horários de lanche (já especificados pela ETTUSA), informações sobre a empresa, a linha, o veículo e a operação (por exemplo, viagens programadas). Os horários se apresentam divididos de acordo com os postos de controle. Cada linha apresenta os seus respectivos postos de controle, que são locais existentes dentro do itinerário. Geralmente estes postos são os pontos extremos do itinerário da linha. No caso do município de Fortaleza, os terminais de integração também funcionam como postos de controle.

Quanto à operação, as empresas adotam a sistemática de tentar manter o motorista e o cobrador sempre no mesmo veículo durante toda a sua jornada, evitando assim a troca de veículos, opção essa já adotada em vários trabalhos realizados nessa área. Os operários (motorista e cobrador) também manifestam a sua preferência por este tipo de operação já que os mesmos não estarão sujeitos a passar o dia todo vinculado à empresa. Ou seja, o motorista e o cobrador trabalham no 1º trecho da jornada de trabalho, têm um intervalo de acordo com o mínimo especificado por lei e depois retornam para finalizar as suas jornadas, e com isso voltam para casa ou até mesmo exercem outras funções a fim de aumentar a sua renda, como por exemplo, a de taxista (como é bastante visto na prática).

Sendo assim, o ATTON se propõe a tentar resolver o problema sobre dois aspectos. Primeiramente, proibindo a troca de veículos semelhantemente a forma que é adotada pelas empresas do município de Fortaleza. E por outro lado, permitindo que o motorista e o cobrador possam mudar de veículos a fim de que a sua jornada de trabalho possa, ou não, ser finalizada em outro ônibus. Além disso, o ATTON permite que o programador escolha se vai considerar, ou não, os horários de lanche especificados pela ETTUSA. Caso o programador não selecione os horários de lanche especificados pela ETTUSA, o programa selecionará os horários de lanche, através do seu processo de otimização.

Porém, a opção de proibir a troca de veículos e horários de lanche livre (ou seja, não considerando os especificados pela ETTUSA) ainda necessita de outros dados de entrada para tal feito. Mesmo assim, o ATTON realiza a programação e acredita-se que

a mesma possa, pelo menos, servir de base para que o programador realize modificações, caso julgue necessário, na programação de veículos modificando assim os horários de lanche pré-especificados.

Portanto, assumindo essas características, o ATTON considera a programação de veículo como o primeiro dado de entrada. O programa não lê essa programação diretamente da OSO. Para isso, é construído um arquivo texto contendo todas as informações importantes contidas na OSO que serão usadas pelo ATTON. São elas: número da linha, número do ônibus, número do bloco, horários de entrada e saída do bloco, número do posto de controle, e horários de cada posto de controle, separadamente. Além disso, os horários de lanche são especificados nesse arquivo informando a linha, o ônibus, o bloco, o posto de controle e por fim o horário. A Figura 4.1, mostrada abaixo, ilustra o arquivo texto que contém a programação de veículos.

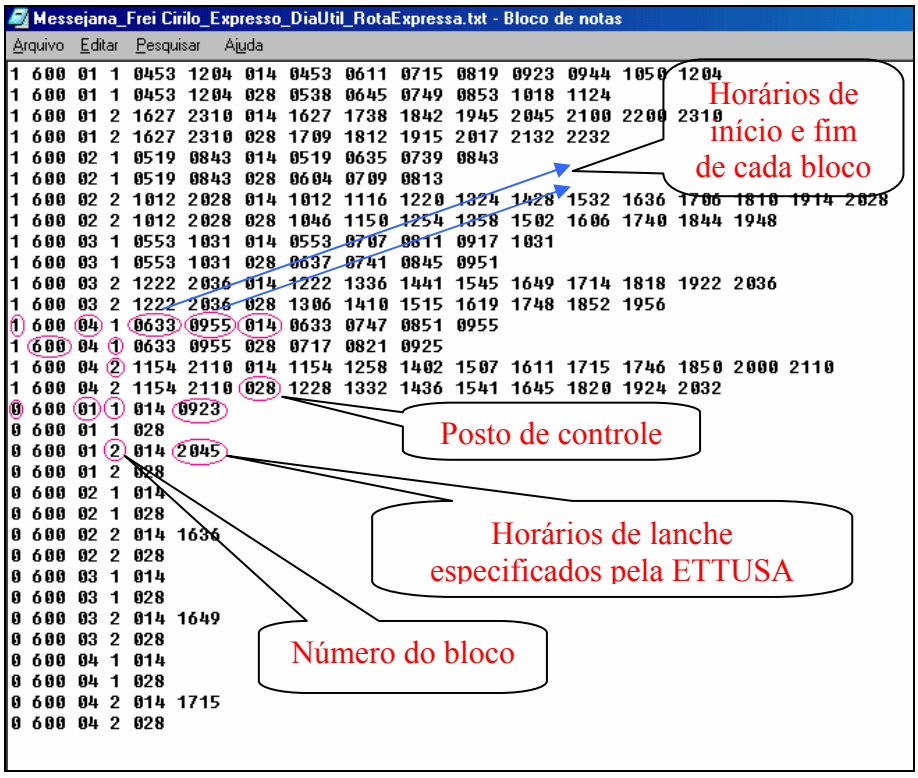

Figura 4.1: Arquivo texto que contém a programação de veículos.

Os números 1 e 0 que aparecem na primeira coluna, antes do número da linha 600 (segunda coluna) são apenas código para que o ATTON identifique horários normais da operação e horários de lanche especificados pela ETTUSA, respectivamente.

Portanto, para a Figura 4.1 o número da linha seria 600, e para o ônibus de número 04 (terceira coluna), por exemplo, têm-se dois blocos (o número do bloco corresponde à quarta coluna) e os horários de cada posto de controle (a partir da oitava coluna logo após o número do posto), posto 014 e posto 028. No ônibus 01 tem-se o horário de lanche (a partir da linha que inicia com o número 0) apenas no posto 014 para o primeiro bloco (09:23 h) e para o segundo bloco (20:45 h). O programa também contém dois arquivos de banco de dados, *linhas.dbf* e *postos.dbf*, onde são armazenadas as características de cada linha e posto. O arquivo *linhas.dbf* apresenta o número da linha, o nome da linha e os postos de controle da linha. Já no arquivo *postos.dbf* estão contidos o nome do posto e seu número.

Existem linhas onde a operação não inicia na garagem, apresentando assim o que se chama de kilometragem morta. Essa kilometragem se refere à extensão percorrida pelo ônibus da garagem até o ponto inicial da linha, ou então e do ponto final da linha até a garagem, quando do recolhimento do ônibus. Para algumas empresas visitadas no município de Fortaleza o tempo gasto para percorrer essa kilometragem não é o mesmo para todos os ônibus da mesma linha. Isso se deve ao fato de alguns ônibus, da mesma linha, iniciarem a operação em pontos diferentes. Por exemplo, a linha 052 – Grande Circular 02 (uma das linhas analisadas) apresenta ônibus que iniciam a operação em um determinado posto de controle enquanto outros ônibus iniciam a sua operação em outro posto de controle. Com isso, o motorista e o cobrador possuem um horário de saída e de chegada diferente do horário inicial e final da linha, como especificado na OSO.

Sendo assim, optou-se por introduzir, quando for o caso, esse novo horário de saída e de chegada diretamente na programação de veículos, e não deixar para que o programador informasse esse tempo ao programa visto que passaria a existir um tempo de diferença entre a saída (ou chegada) da garagem e o início (ou fim) da operação para cada ônibus.

Outros dados de entrada solicitados pelo ATTON são os comprimentos máximo e mínimo do trecho de trabalho. Ou seja, o comprimento máximo e mínimo em que o motorista e cobrador terão que trabalhar até que se possa realizar o intervalo. No caso em que os horários de lanche forem os especificados pela ETTUSA, se faz necessário

que o programador realize um estudo prévio na linha para que adote um valor de comprimento máximo para o trecho de trabalho, de forma a evitar que o programa quebre a programação de veículos em muitos pedaços.

O ATTON também solicita ao programador dados referentes ao comprimento máximo e mínimo para o intervalo, como também o comprimento máximo especificado por lei para o intervalo (no caso de Fortaleza, 4 horas ou 240 min). O comprimento mínimo fornecido pelo programador terá que ser superior ao especificado pela lei (30 min, no caso de Fortaleza). Porém, os testes consideraram um comprimento mínimo de 15 minutos para o intervalo, pois foi verificado, em visita as empresas, que esse valor ainda é praticado pelas mesmas.

Fornecidos esses dados inicialmente, o ATTON também solicita ao programador que informe quais os postos de controle em que poderá ser feita à rendição e se quer que o programa considere, ou não, os horários de lanche especificados pela ETTUSA. O programa informa os postos de controle através dos arquivos *linhas.dbf* e *postos.dbf*, e os horários de lanche através do arquivo texto contendo a programação de veículos, considerando apenas a parte em que existe o número 0 antes do número da linha. As Figuras 4.2 e 4.3 ilustram o já comentado anteriormente.

Como pode ser verificado nas Figuras 4.2 e 4.3, através dessas duas telas o programador fornece os valores referentes aos comprimentos máximo e mínimo dos trechos de trabalho, do intervalo, valor máximo permitido para o intervalo (240 minutos), escolhe a forma de operação (proibindo a troca de veículos ou não) e determina em quais postos poderão ser realizados o intervalo e a rendição.

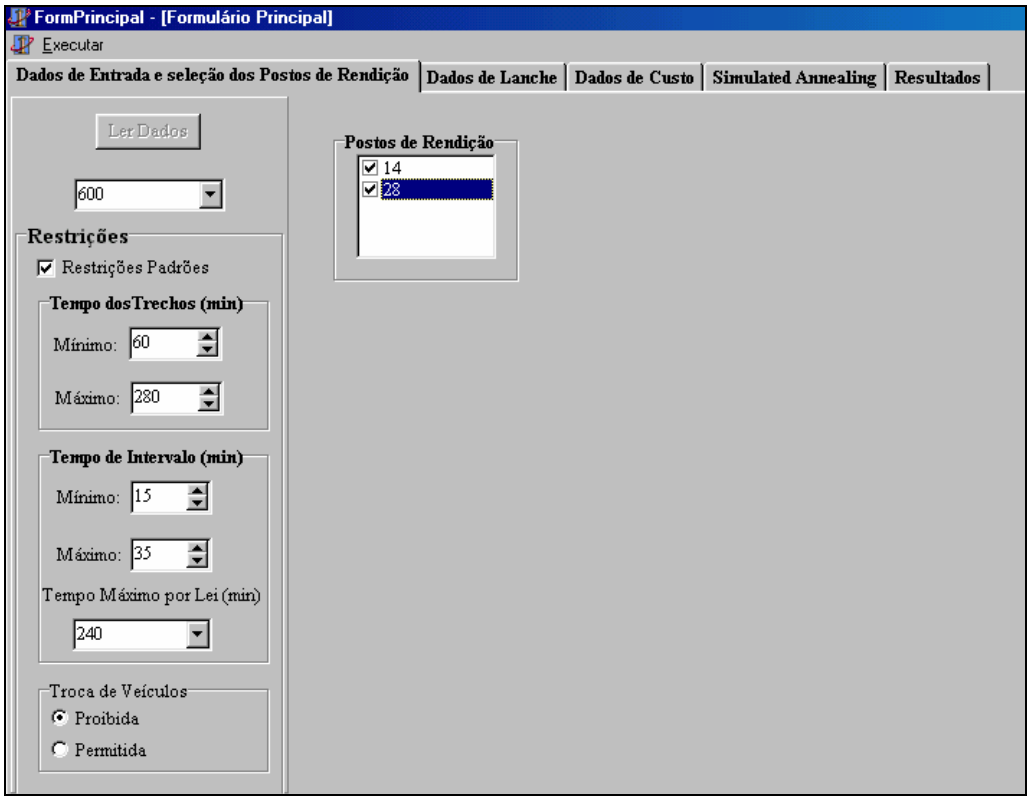

Figura 4.2: Tela de especificação dos dados de entrada e postos para rendição.

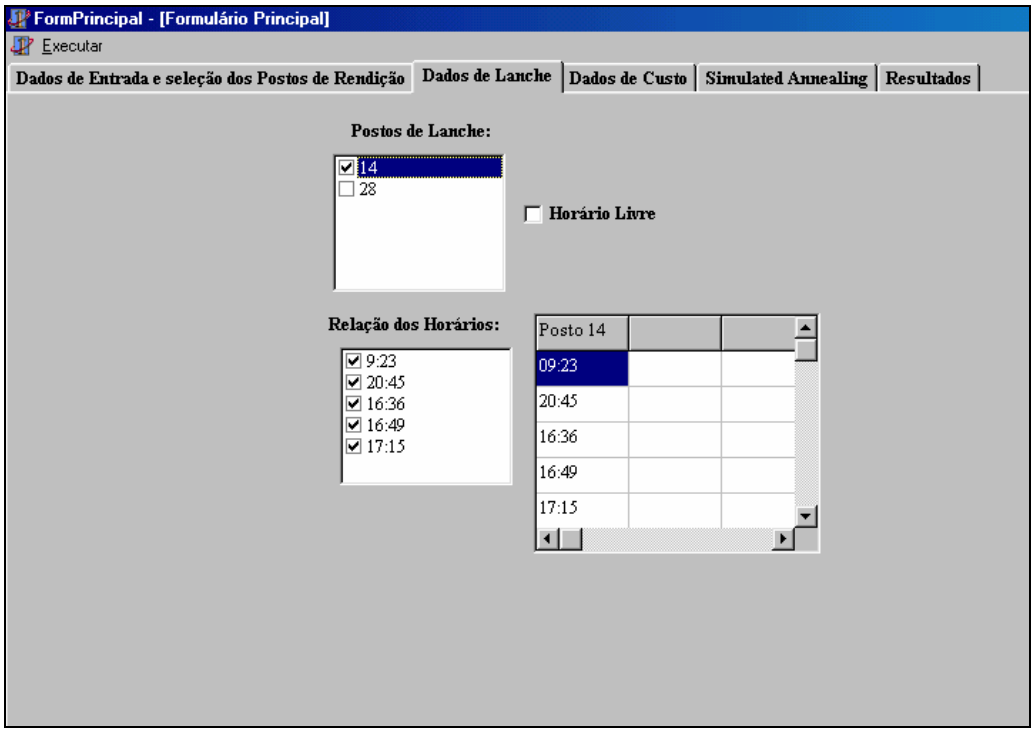

Figura 4.3: Tela para especificação dos postos escolhidos para a realização do lanche.

Os dados relativos ao custo também são informados ao ATTON nessa fase. Através de outra tela (Figura 4.4) o programa solicita que sejam informados dados

referentes a salários de motorista e cobrador, percentagem acrescida ao valor da hora trabalhada no caso de horas-extras e adicional noturno (caso em que o operário trabalha no período das 22:00 h até as 05:00 h), tempo ideal de trabalho (7:20 hs ou 440 minutos), tempo máximo de horas-extras (120 minutos ou 2 horas), e os horários inicial e final do período noturno para que seja calculado o adicional noturno.

Além disso, ainda de forma acadêmica, o ATTON solicita ao programador que informe o nível de importância dado para as horas-extras, jornadas de apenas um único trecho, ociosidade e sobre-cobertura de trechos de trabalho (caso em que o mesmo trecho de trabalho é coberto por mais de uma tripulação). Os valores do nível de importância para cada característica indesejável, citada anteriormente, varia de 0 a 100, e não necessariamente têm que somar 100. Esses dados serão utilizados no processo de otimização vinculado ao *Simulated Annealing* de forma que o programador estará priorizando que característica receberá uma maior atenção no processo de otimização. A ociosidade se refere ao fato de que a empresa paga o valor correspondente às 7:20 horas de trabalho, mesmo no caso em que o motorista e o cobrador trabalhem em uma jornada de tamanho inferior às 7:20 hs. Portanto, a ociosidade mede a má aplicação de recursos por parte da empresa. Mais adiante esses dados serão explicados de forma mais detalhada. A Figura 4.4 apresenta a tela onde são informados os dados referentes a custo.

Por fim, o programador terá que informar os dados que serão utilizados pela metaheurística *Simulated Annealing*. São eles: número de iterações externas, número de iterações internas e o coeficiente de decréscimo da temperatura. Além disso, o ATTON permite ao programador habilitar, ou não, uma condição de parada do processo *annealing* informando o número de iterações em que, caso a solução permaneça inalterada, o programa finalize o processo de otimização e informe a melhor solução encontrada. Caso essa condição de parada não seja habilitada pelo programador, o ATTON considerará o número de iterações externas como parâmetro. A Figura 4.5 ilustra a tela que será mostrada ao programador para que o mesmo informe os valores necessários para a metaheurística *Simulated Annealing*.

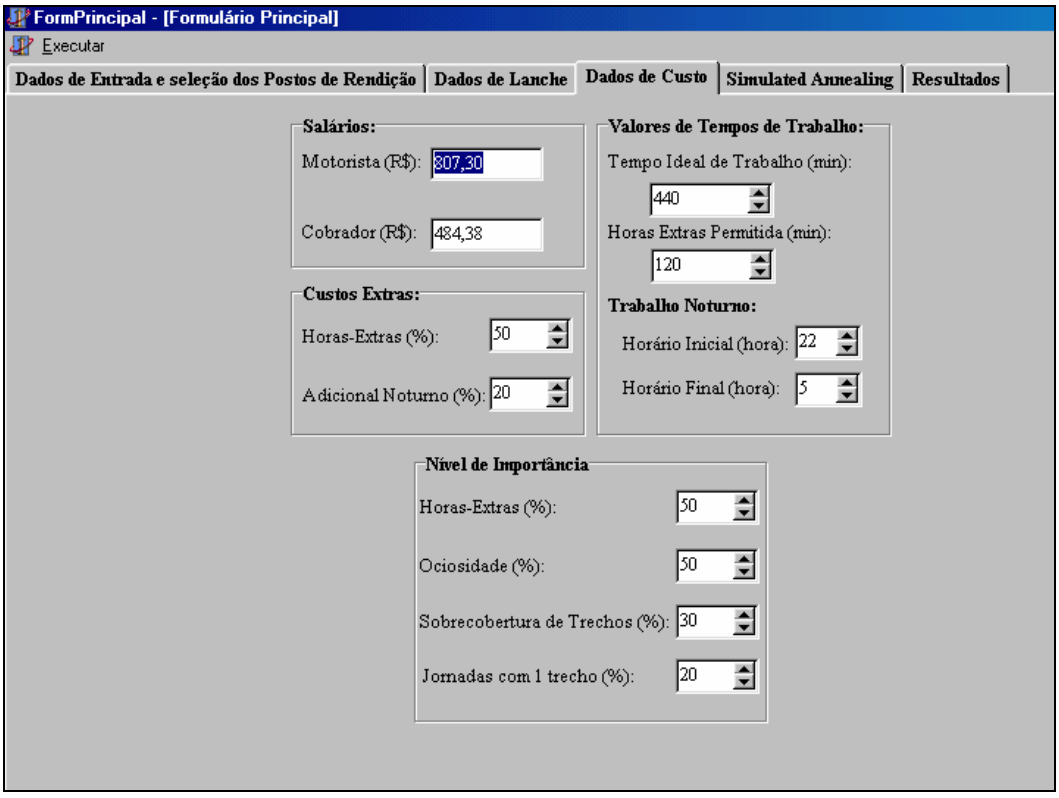

Figura 4.4: Tela para a especificação dos valores de custo.

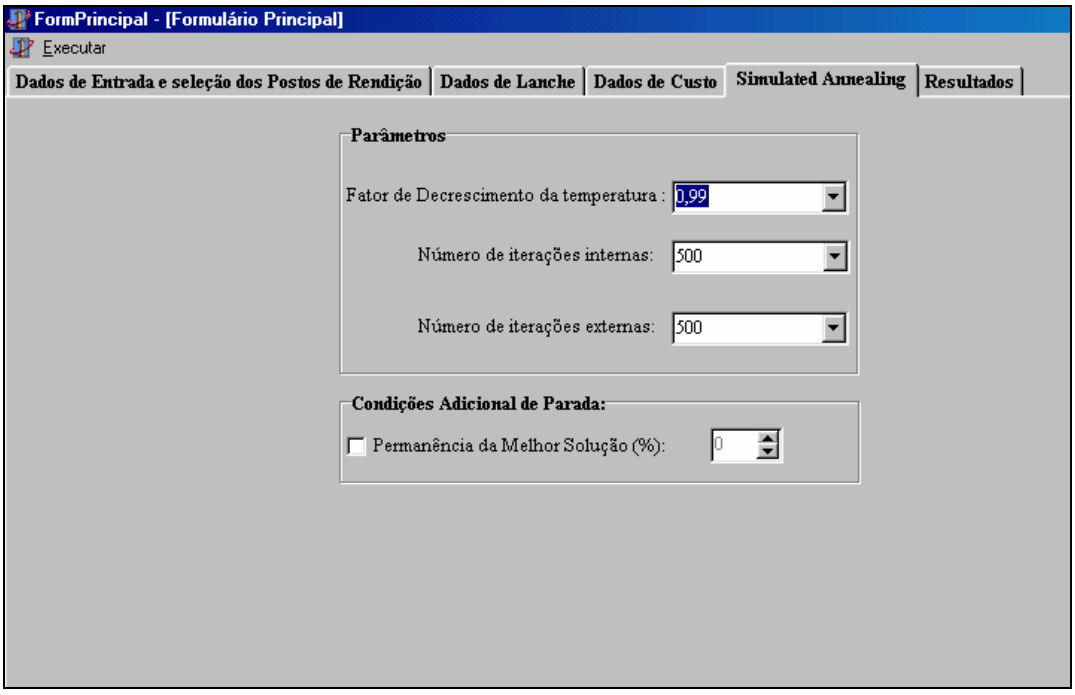

Figura 4.5: Tela para a especificação dos parâmetros do *Simulated Annealing*.

Finalizada a entrada de dados, o programador poderá selecionar o item “Solução Inicial” no menu “Executar” da barra de ferramentas e então obter os resultados da solução inicial.

4.3. SOLUÇÃO INICIAL

Antes da criação da solução inicial, o ATTON utiliza uma metodologia semelhante à apresentada por SMITH (1986) para a seleção de oportunidades de rendição viáveis. Isso se deve ao fato de que o problema se tornaria muito mais complexo se fossem consideradas todas as oportunidades de rendição.

Portanto, as oportunidades de rendição viáveis (OR viáveis) são escolhidas de acordo com os valores de comprimento máximo e mínimo dos trechos de trabalho, determinados pelo programador. O programa então inicia a seleção de oportunidades de rendição que satisfaçam essas restrições, tanto no sentido crescente como no sentido decrescente dos horários. A utilização desses dois sentidos é necessária para que sejam estabelecidas faixas de oportunidades de rendição viáveis. Assim o bloco de cada veículo ficará reduzido apenas às oportunidades de rendição contidas na faixa de OR viáveis. A Figura 4.6 ilustra esses passos para a construção da faixa de OR viáveis.

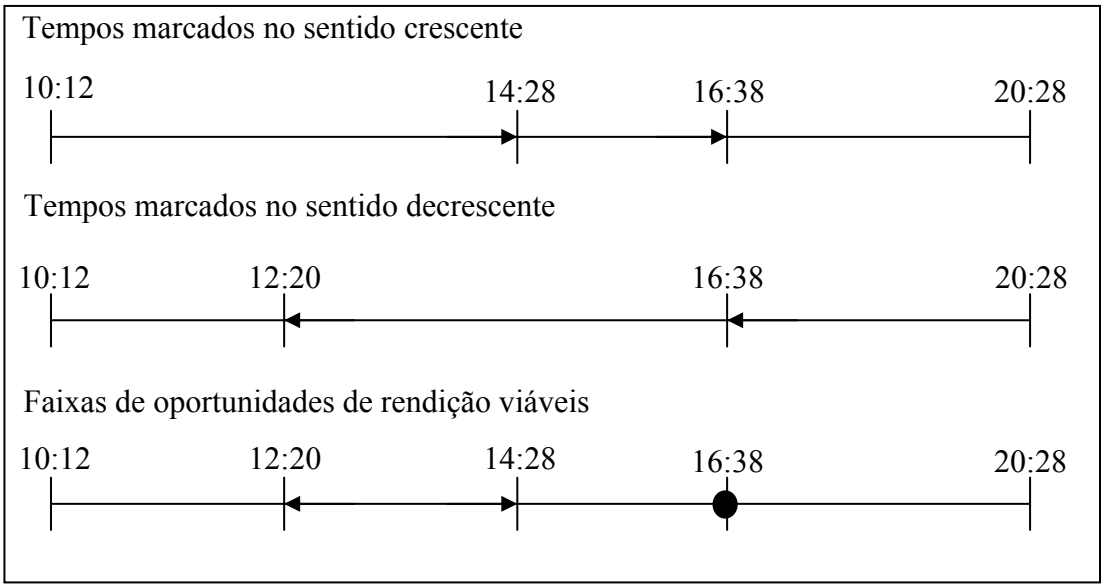

Figura 4.6: Exemplificação da determinação da faixa de OR viáveis.

Para a Figura 4.6 utilizou-se o segundo bloco do ônibus 2 da linha 600 (uma das linhas escolhidas para a realização do Estudo de Caso – item 4.5.1) e os comprimentos máximo e mínimo dos trechos de trabalho de 280 e 60 minutos, respectivamente.

Portanto, a faixa criada pelo ATTON compreende as oportunidades de rendição existentes entre o horário de 12:20 h e 14:28 h, como também o horário de 16:38 h.

Caso os horários de lanche especificados pela ETTUSA estejam sendo considerados, o programa terá que incluí-los dentro da faixa de OR viáveis (como é o caso do horário de 16:38 h escolhido e exemplificado na Figura 4.6). Após as faixas de OR viáveis terem sido criadas, o ATTON exibe um gráfico informando ao programador quais oportunidades de rendição foram escolhidas, conforme a Figura 4.7. Essa informação é importante no sentido em que pode possibilitar ao programador retornar para a criação de outras OR viáveis caso não prefira a apresentada pelo ATTON.

No gráfico apresentado pela Figura 4.7 os pontos em cor preta representam as oportunidades de rendição escolhidas pelo ATTON, e os pontos em cor vermelha representam o restante das oportunidades de rendição que constituem o bloco de cada veículo da linha, e que são determinadas pela ETTUSA, através da OSO.

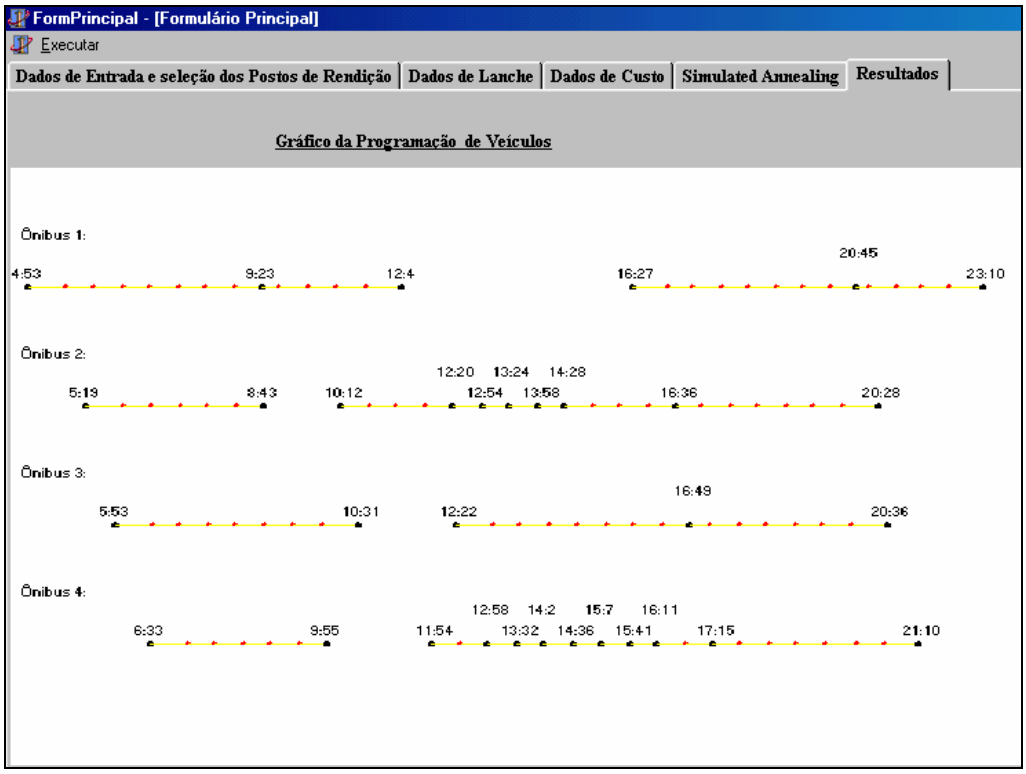

Figura 4.7: Gráfico que apresenta as faixas de OR viáveis determinadas pelo programa.

Após a criação das faixas de OR viáveis o programa parte para a construção da solução inicial. A solução inicial é construída de duas formas diferentes, de acordo com a opção escolhida pelo programador em relação à troca de veículos por parte do motorista e cobrador. Sendo assim, será apresentada a seguir a metodologia adotada para a construção da solução inicial para os dois casos.

4.3.1. Solução Inicial - Troca de Veículos Proibida

Para esse tipo de operação a solução é criada considerando um ônibus de cada vez de acordo com a ordem em que o ônibus aparece na programação de veículos. Para cada ônibus são determinados os trechos de trabalho por escolher aleatoriamente uma oportunidade de rendição contida nas faixas de OR viáveis. Paralelo a determinação dos trechos de trabalho, o programa constrói as jornadas de trabalho para o ônibus em análise.

Portanto, o programa parte do horário inicial *i* do trecho *N* (começando pelo horário de entrada do ônibus em análise) e percorre o vetor que contém as faixas de OR viáveis para esse ônibus. O horário final *j* do trecho de trabalho *N* será escolhido aleatoriamente dentro da faixa de OR viáveis que se encontra logo após o horário *i*. Porém, antes de escolher o horário final do trecho, o programa primeiro verifica qual tipo de horário de lanche está sendo considerado. Caso os horários de lanche sejam os especificados pela ETTUSA, o programa verificará se o horário, logo após o horário *i*, é um lanche, ou se não existe nenhum horário de lanche entre o horário *i* e o horário escolhido aleatoriamente. Essa verificação garante que o ATTON não venha a desconsiderar os horários de lanche caso eles sejam os especificados pela ETTUSA.

Sendo assim, à medida que o programa percorre o vetor de OR viáveis, os trechos de trabalho vão sendo escolhidos e automaticamente vão sendo alocados às jornadas de trabalho.

No caso em que está se considerando os horários de lanche especificados pela ETTUSA, a própria OSO estabelece os horários de saída após o lanche. Portanto, quando o horário final do 1º trecho de trabalho é um horário de lanche, o programa

automaticamente seleciona o horário de saída para ser o início do 2º trecho de trabalho para a jornada que está sendo construída. Vale ressaltar que é importante que o programador faça um estudo prévio na programação de veículos para que os valores de comprimento máximo e mínimo do intervalo sejam adequados a fim de compreender esses horários de saída. No caso em que os horários de lanche são escolhidos livremente pelo programa, isso não se faz necessário.

Para os ônibus que apresentam mais de um bloco de viagens, o ATTON também considera a possibilidade da jornada apresentar o 1º trecho em um bloco e o 2º trecho de trabalho em outro bloco. Para isto, basta que a diferença entre o horário final do 1º bloco e o horário inicial do 2º bloco não seja maior do que valor máximo especificado por lei para o intervalo.

Assim, pode-se verificar que para esse tipo de operação o ATTON não irá apresentar, na solução inicial, uma característica que é chamada de sobre-cobertura. Ou seja, um mesmo trecho de trabalho sendo coberto por mais de uma tripulação. Isso porque o programa percorre um ônibus de cada vez no sentido crescente dos horários e na ordem em que os veículos aparecem na programação de veículos. Portanto, quando o programa está considerando um horário inicial *i* para um trecho de trabalho *N* que apresenta uma diferença entre o horário *i* e o horário final de operação daquele ônibus menor do que o comprimento máximo estabelecido para o trecho de trabalho, o programa seleciona o horário final de operação daquele ônibus como o horário final do trecho de trabalho *N*, e parte para outro ônibus ou acaba a construção da solução inicial, caso esteja analisando o último veículo.

Após a solução inicial ter sido construída, o programa informa os resultados de custo e de tempo para toda a programação e também para cada jornada. Além disso, o programa mostra um gráfico, semelhante ao gráfico de escolha das OR viáveis (Figura 4.7), apresentando as jornadas criadas.

4.3.2. Solução Inicial - Troca de Veículos Permitida

Diferente do tipo de operação anterior, nesse tipo de operação o ATTON primeiramente estabelece os vários trechos de trabalho em cada ônibus da mesma forma que é feito para o caso anterior. Após o vetor, contendo os trechos de trabalho para cada ônibus, ter sido construído, nomeia-se cada trecho de trabalho como uma tarefa e posteriormente forma as jornadas de trabalho por combinar as tarefas entre si, permitindo a sobre-cobertura de tarefas.

Portanto, cada trecho de trabalho é relacionado a uma tarefa. Cada tarefa contém informações sobre o número do ônibus, o número do bloco, o número do posto de controle do horário inicial do trecho de trabalho, o horário inicial e final do trecho, a quantidade de vezes que aquele trecho está sendo usado na solução e a informação indicando se a tarefa está sobre-coberta ou não.

Após a criação das tarefas o ATTON irá tentar combinar cada tarefa com outra existente. Como o programa percorre cada tarefa, a solução inicial apresenta uma quantidade razoável de sobre-cobertura e algumas jornadas iguais. O programa elimina as jornadas repetidas antes de otimizar a solução inicial.

Para combinar as tarefas, o ATTON utiliza uma metodologia semelhante à apresentada por LAYFIELD *et al*. (1998) que trabalha com as chamadas cadeias de *mealbreak* (parada para lanche ou, mais especificamente, intervalo). Segundo LAYFIELD *et al*. (1998) uma boa programação de motoristas e cobradores deve conter boas oportunidades de rendição, que conseqüentemente estariam presentes nas cadeias de *mealbreak* estabelecidas.

De acordo com LAYFIELD *et al*. (1998) uma característica freqüentemente encontrada em boas programações de motoristas, em que o método de seleção de pontos de rendição é modelado, é que cadeias de parada para refeição (*mealbreak*) têm um efeito benéfico em reduzir o número de trechos de trabalho necessários em uma programação. Um *mealbreak* pode ser definido como o caso onde no mínimo dois turnos de motoristas são unidos de tal forma que um motorista termine um trecho para

iniciar um *mealbreak* em uma oportunidade de rendição onde outro motorista terminou um *mealbreak* (e inicie o trabalho novamente), e, então, substitua o primeiro motorista. A Figura 4.8 a seguir mostra o exemplo de um *mealbreak*.

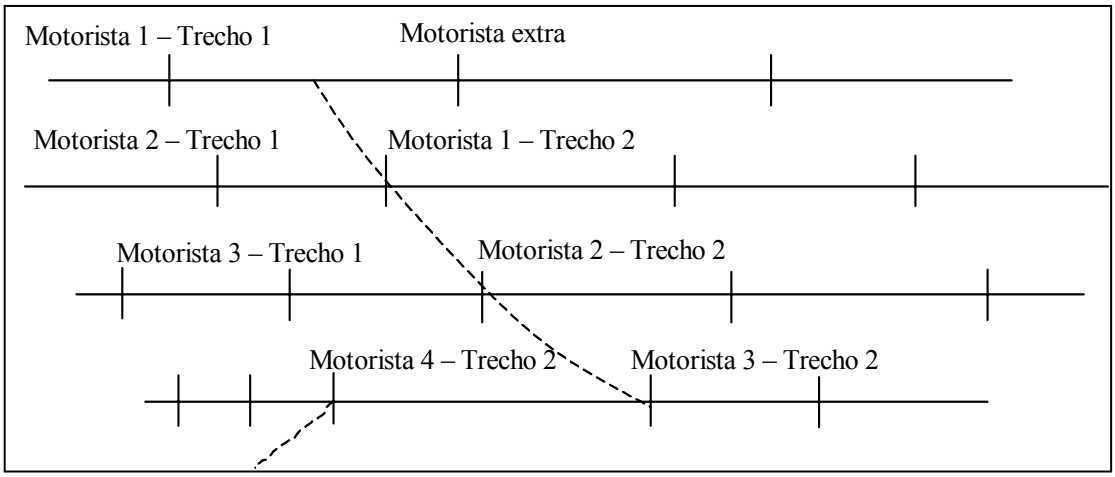

Figura 4.8: Cadeias de *mealbreak* (LAYFIELD *et al*., 1998).

Sendo assim, após o programa ter criado as faixas de OR viáveis, são estabelecidos dois vetores denominados *Prox* e *Ant*. Semelhantemente ao trabalho reportado por LAYFIELD *et al*. (1998), o vetor *Prox* contém, para uma determinada OR viável *i*, todos as outras oportunidades de rendição viáveis que iniciam um trecho de trabalho logo após a OR viável *i* (respeitando os valores de máximo e mínimo, especificados inicialmente, para o intervalo). Analogamente, o vetor *Ant* contém todas as OR viáveis que finalizam um trecho de trabalho antes da OR viável *i* (também respeitando os valores de intervalo). A Figura 4.9 ilustra um exemplo a fim de elucidar melhor o explicado acima.

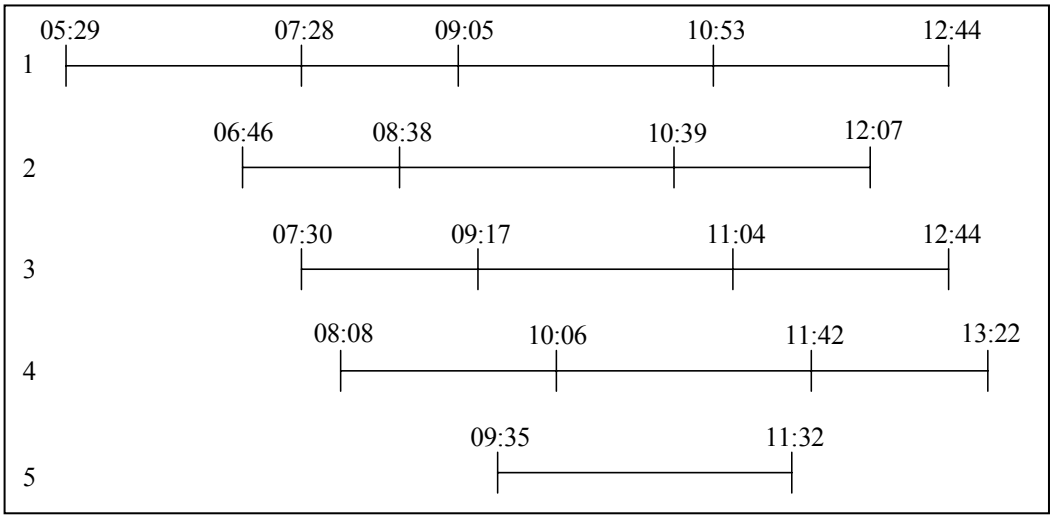

Figura 4.9: Gráfico da programação de veículos (LAYFIELD *et al*., 1998).

Por exemplo, na Figura 4.9 se for considerado um valor mínimo para o intervalo de 30 minutos e um valor máximo de 90 minutos, os valores das variáveis *Prox* e *Ant* para a OR viável que corresponde ao horário 08:38 h no ônibus 2 seriam os apresentados na equação 4.1 e 4.2. O primeiro número corresponde ao ônibus e o segundo ao horário (ônibus,hh:mm).

$$\Pr ox_{(2,0838)} = \{(3,09:17),(5,09:35),(4,10:06)\} \qquad (4.1)$$

$$Ant_{(2,0838)} = \{(1,07:28)\} \qquad (4.2)$$

Vale ressaltar que existem alguns horários (OR viáveis) específicos que não possuem valores alocados para o vetor *Prox* ou *Ant*. Por exemplo, uma OR viável *i* que corresponda a um horário inicial de operação do ônibus, ou a um horário inicial de um bloco, terá o vetor *Prox* nulo, visto que a OR viável *i* não finaliza nenhum trecho. Semelhantemente, uma OR viável *i* que corresponda a um horário final de operação do ônibus, ou horário final de um bloco, terá seu vetor *Ant* nulo, pois não inicia nenhum trecho de trabalho para que possa ser feita a combinação.

Além disso, ao formar os vetores *Prox* e *Ant*, o ATTON leva em consideração a restrição relacionada ao posto de controle. Essa restrição existe para que o motorista e o cobrador troquem de veículo no mesmo posto em que pararam para realizar o intervalo.

Ou seja, a dupla (motorista e cobrador) que realizar a sua parada, para o intervalo, no posto de controle *P*, terá que pegar outro ônibus (ou o mesmo se possível) no mesmo posto de controle *P*. Assim, evita-se que a dupla tenha que se deslocar para outro posto, o que tornaria a operação mais onerosa e indesejável por parte dos operários (motorista e cobrador).

Sendo assim, caso alguma OR viável *i* não apresente nenhum horário dentro dos vetores *Prox* e *Ant* que satisfaçam à restrição referente aos postos de controle, o ATTON irá construir outros trechos de trabalho, até essa restrição ter sido atendida. Por exemplo, se a OR viável *i* pertencer ao posto de controle *P* e não houver nenhuma OR viável que inicie ou termine um trecho de trabalho, dentro dos vetores *Prox* e *Ant*, que também seja do posto *P*, o programa irá determinar outros trechos de trabalho utilizando-se de outras OR viáveis.

Porém, se essa restrição não for atendida somente para um dos vetores *Prox* ou *Ant*, o programa continua o processo normalmente, sem retornar para a formação de outros trechos de trabalho. No momento está sendo avaliado o custo computacional relacionado a essa característica. Portanto, está sendo contabilizada, cada vez que o ATTON é executado, a quantidade de vezes em que o programa retorna para determinar os trechos de trabalho. Os resultados dessa análise serão apresentados no capítulo referente à conclusão. Vale ressaltar que esse processo só acontece para o caso em que a troca de veículos é permitida e que o programa só continua depois que a restrição for atendida de forma geral para todos os horários de início e fim dos trechos de trabalho.

A partir disso, selecionada a tarefa (ou trecho), o ATTON verifica se existem horários dentro do vetor *Prox* e *Ant* para os respectivos horários finais e iniciais da tarefa em análise. Se existir horários que satisfaçam à restrição relacionada ao posto de controle dentro dos dois vetores, o programa escolhe o que apresentar o menor intervalo. Se só houver o horário em apenas um dos vetores, o programa realiza a combinação normalmente. E com isso, a jornada de trabalho está formada. Caso não haja como combinar a tarefa em análise, a jornada será composta por um único trecho de trabalho.

Semelhantemente ao caso anterior, o ATTON apresenta uma tela contendo todos os resultados da solução inicial criada a partir da opção de troca de veículos permitida. Porém, para esse tipo de operação o ATTON também informa quantas vezes o vetor que contém os trechos de trabalho foi formado. Essa última informação tem um caráter apenas acadêmico já que se pretende investigar o custo computacional por se adotar essa prática de reformação de trechos de trabalho, caso a restrição relacionada aos postos de controle não seja atendida.

4.4. PROCESSO DE OTIMIZAÇÃO UTILIZANDO O *SIMULATED ANNEALING*

Semelhantemente à construção da solução inicial, o ATTON também adota dois processos distintos de otimização de acordo com a opção que permite, ou não, a troca de veículos. O processo *Simulated Annealing* consiste em realizar modificações na solução corrente *s* com o objetivo de encontrar outra solução vizinha *s*', e verificar se a solução encontrada *s*' é melhor (em termo de custo), ou não, do que a solução corrente *s*. Se *s*' for melhor, aceita *s*' como a solução corrente. Senão, aceita *s*' de acordo com uma probabilidade que dependerá da temperatura e da diferença entre os valores da função objetivo referente a cada solução.

Portanto, a seguir serão explicadas, de forma mais detalhada, os dois processos de otimização utilizando-se do *Simulated Annealing*. Porém, primeiramente será feita uma abordagem sobre a função objetivo a ser minimizada, posteriormente será apresentada a estrutura do processo de otimização, considerando os dois tipos de operação (troca de veículos proibida e permitida), e ao final uma explanação sobre as condições de parada para o processo do *Simulated Annealing*.

4.4.1. Função Objetivo

O ATTON tenta minimizar três aspectos importantes no problema de alocação de pessoal (motorista e cobrador) em empresas de transporte coletivo por ônibus. São eles: número de jornadas (conseqüentemente o número de motoristas e cobradores estará sendo minimizado), horas-extras e ociosidade. A ociosidade reflete o quanto a empresa pagou em horas trabalhadas sem que necessariamente essas horas tivessem

realmente sido trabalhadas. Ou seja, caso o motorista e o cobrador trabalhem em uma jornada de tamanho inferior ao ideal (7:20 horas), a empresa tem o dever de pagar o valor monetário correspondente às 7:20 hs completas.

Além disso, existem outros aspectos importantes que o ATTON tenta minimizar. Esses aspectos podem ser considerados como características indesejáveis na alocação de motorista e cobrador.

a) Sobre-cobertura de trechos de trabalho: é importante que a solução final não apresente nenhum trecho que esteja sendo coberto por mais de uma tripulação.
b) Jornadas com apenas um único trecho de trabalho: o programa tenta minimizar jornadas que apresentem apenas um único trecho de trabalho. O programa só constrói jornadas de no máximo dois trechos de trabalho.

Para cada aspecto indesejável atribui-se um peso. Tanto para os aspectos indesejáveis citados acima, como também para horas-extras e ociosidade. Portanto, a função objetivo pode ser caracterizada pela adição das equações 4.3, 4.4 e 4.5.

a) Minimizar as jornadas de trabalho e as horas-extras: minimizando as jornadas de trabalho, conseqüentemente o programa estará minimizando o número de tripulações (motorista e cobrador);

$$f(s) = \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} + \alpha \sum_{j=1}^{n} CHE_j \qquad (4.3)$$

em que,

s : Solução corrente;

$f(s)$: Função custo da solução corrente;

n : Número de jornadas existentes na solução;

m : Número de tipos de custo da hora trabalhada (diurno e noturno);

C_{ij} : Custo do tipo *i* da jornada *j*;

α : Peso atribuído ao fator horas-extras; e

CHE_j: Custo relativo à hora-extra da jornada *j*.

b) Minimizar a ociosidade: traduz-se por uma maximização da utilização da mão-de-obra. Ou seja, o ATTON tentar minimizar o surgimento de jornadas que apresentem um comprimento muito aquém do comprimento ideal (7:20 hs) a fim de evitar que a empresa pague por um serviço que não está sendo executado.

$$g(s) = \beta \sum_{j=1}^{n} CO_j \tag{4.4}$$

em que,

$g(s)$: Função custo da solução corrente;

β: Peso atribuído ao fator ociosidade; e

CO_j: Custo relativo à ociosidade da jornada *j*.

c) Minimizar aspectos indesejáveis na programação: estes aspectos são os relacionados à sobre-cobertura de trechos de trabalho e as jornadas de apenas um único trecho de trabalho.

$$h(s) = \gamma \sum_{j=1}^{n} CS_j + \delta \sum_{j=1}^{n} CUT_j . x_j \tag{4.5}$$

em que,

$h(s)$: Função custo da solução corrente;

γ: Peso atribuído à ocorrência de sobre-cobertura de trechos de trabalho;

CS_j: Custo associado à sobre-cobertura de trechos de trabalho;

x_j: Se igual a 1, a jornada apresenta apenas um único trecho de trabalho. Se igual a 0, a jornada apresenta mais de 1 trecho de trabalho;

δ: Peso atribuído à ocorrência de jornadas de apenas um único trecho; e

CUT_j: Custo associado às jornadas com 1 trecho de trabalho.

Portanto, a função objetivo global pode ser simplificada pela equação 4.6.

$$FG(s) = f(s) + g(s) + h(s) \tag{4.6}$$

De forma mais detalhada, o custo C_{ij} (custo do tipo *i* da jornada *j*) é calculado normalmente considerando a hora trabalhada pela tripulação multiplicada pelo seu valor monetário, de acordo com o período em que a jornada está sendo cumprida (seja durante o dia ou entre o período noturno, das 22 h às 05 h). Isso se deve à diferença no cálculo do custo da jornada. O custo CHE_j (custo relativo à hora-extra da jornada *j*) é calculado separadamente para cada jornada, caso a mesma apresente hora-extra. Se uma parte da hora-extra estiver dentro do período noturno, considera-se a hora-extra integral como se a mesma estivesse completamente dentro do período noturno.

Se a jornada apresentar ociosidade, o ATTON calcula a diferença entre o comprimento real da jornada e as 7:20 hs (comprimento ideal). A diferença (em tempo) será transformada em valores monetários e considerada como custo relativo à ociosidade da jornada (CO_j).

Quanto aos aspectos indesejáveis, para cada jornada é verificado se cada trecho de trabalho apresenta uma sobre-cobertura. Sendo assim, o ATTON calcula o valor da hora trabalhada apenas para o trecho que apresentar sobre-cobertura e calcula assim o custo CS_j. Já o custo CUT_j é calculado apenas para as jornadas que tiverem um único trecho de trabalho. O programa contabiliza as horas trabalhadas por esta jornada e as transforma em valor monetário.

Vale salientar que o ATTON trabalha com duas variáveis que armazenam o custo da solução. Uma variável armazena o valor real da solução e outra variável armazena esse mesmo valor real adicionado aos outros custos atribuídos ao processo de otimização. Ou seja, o ATTON apresenta, como resultado final, o custo total das jornadas de trabalho criadas, da mesma forma que é calculado pelas empresas que operam esse tipo de sistema. Porém, para efeito de otimização, o ATTON trabalha com

um valor de custo bem superior ao apresentado. Esse acréscimo se deve à introdução de outros tipos de custo, como já mencionados anteriormente através das equações 4.3, 4.4 e 4.5, considerando os pesos de cada aspecto indesejável.

O valor de custo que é utilizado pelo processo de otimização a partir da solução inicial é considerado como o valor para a temperatura inicial no processo realizado pela metaheurística *Simulated Annealing*.

4.4.2. Processo de Otimização - Troca de Veículos Proibida

A busca de vizinhança consiste em escolher, aleatoriamente, uma jornada de trabalho *n* e realizar modificações a fim de encontrar outra solução *s*'. Para esse caso optou-se por apenas um tipo de busca de vizinhança. A jornada escolhida aleatoriamente sofre modificações em seus horários com o objetivo de se obter outra configuração para a solução.

Após a escolha da jornada de trabalho *n*, o programa verifica quais horários, que constituem os trechos de trabalho da jornada escolhida, podem ser alterados. Os horários que não podem ser modificados são os horários iniciais e finais dos blocos de viagem de cada ônibus (incluem-se também os horários de entrada e saída dos veículos). Além desses horários, caso os horários de lanche estejam sendo considerados, os mesmos também não poderão ser modificados. Caso os dois horários (inicial e final da jornada) possam ser modificados, o programa realizará três tipos de modificação e escolherá a que originar a solução de menor custo. As opções são:

a) modifica apenas o horário inicial da jornada;
b) modifica apenas o horário final da jornada;
c) modifica tanto o horário inicial como o horário final da jornada.

Para uma possível jornada *n*, de apenas um único trecho, o programa tenta eliminá-la adicionando-a a alguma jornada anterior ou posterior a jornada *n* escolhida. Isso só será possível se as restrições de comprimento de trecho de trabalho e de comprimento máximo de uma jornada (7:20 hs + 2:00 hs (horas-extras)) não forem violadas. Caso a jornada de apenas um único trecho de trabalho apresente um

comprimento maior do que o máximo especificado pelo programador, o programa divide a jornada *n* em uma jornada de dois trechos de trabalho. A existência de jornadas com apenas um único trecho de trabalho é possível devido à possibilidade do ATTON transformar uma jornada de dois trechos de trabalho em uma jornada com apenas um trecho de trabalho, se for necessário.

Portanto, escolhida a jornada *n*, o programa verifica se a mesma é uma jornada que apresenta ociosidade ou hora-extra, e também verifica os horários iniciais e finais de cada trecho de trabalho. Se a jornada *n* apresentar ociosidade (tamanho menor do que 7:20 hs) o programa irá selecionar outro horário dentro do vetor que contém as OR(s) viáveis de forma a aumentar a jornada *n*. Automaticamente o programa modifica a jornada antecedente à jornada *n*, caso o horário escolhido para ser modificado seja o inicial, ou modifica a jornada posterior à jornada *n*, caso o horário escolhido seja o horário final da jornada.

Analogamente, se a jornada *n* apresentar horas-extras o programa tentará diminuir escolhendo outras OR(s) viáveis. Também da mesma forma, o programa modifica as jornadas adjacentes à jornada *n*, de acordo com o horário escolhido para ser modificado.

Essa modificação automática das jornadas adjacentes à jornada *n* escolhida é realizada com o objetivo de se evitar a sobre-cobertura de pedaços de trabalho. Vale ressaltar que pedaços de trabalho são porções de trabalho menores do que os trechos de trabalho (como já explicado em capítulos anteriores). Na verdade os pedaços de trabalho se constituem como porções de trabalho compreendidas entre duas oportunidades de rendição consecutivas. As jornadas adjacentes à jornada *n* são as jornadas pertencentes ao mesmo ônibus no qual a jornada *n* está contida.

Caso a sobre-cobertura fosse admitida para esse tipo de operação (troca de veículos proibida), a eliminação da mesma se tornaria mais difícil, pela forma em que é construída a solução para esse tipo de operação. Diferentemente da outra opção (troca de veículos permitida) o programa não cria trechos de trabalho primeiramente para

depois combiná-los. As jornadas de trabalho são determinadas simultaneamente à medida que os trechos de trabalho são construídos. E, portanto, o ATTON não teria como considerar esses possíveis pedaços de trabalho sobre-cobertos.

O ATTON também tenta diminuir, ou eliminar se possível, as horas-extras em jornadas que se encontram dentro do período noturno (22 às 05 hs). O objetivo é diminuir o custo global da solução visto que a jornada se torna mais cara nesses tipos de caso. Isso se deve ao fato de que o valor da hora-extra é acrescido de 50% e o valor da hora trabalhada dentro do período noturno é acrescido de 20%. Assim, a hora trabalhada seria multiplicada por 1,5 e por 1,2, o que aumentaria, para a empresa, as despesas com pessoal. Em visitas a algumas empresas do município de Fortaleza-CE verificou-se que as mesmas tentam elaborar programações de forma que as horas-extras aconteçam apenas em jornadas de trabalho operadas durante o dia.

Todas essas modificações nos horários inicial e final da jornada de trabalho escolhida são realizadas sem que as restrições de tamanho máximo da jornada de trabalho sejam violadas (7:20 hs + 2:00 hs). A restrição de comprimento máximo do trecho de trabalho pode ser violada a fim de que o espaço de vizinhança da solução corrente seja aumentado. A solução final poderá apresentar jornadas de trabalho com trechos de trabalho de comprimento um pouco superior ao especificado pelo programador.

Finalmente, o ATTON calcula o custo da nova solução obtida e verifica se essa nova solução será escolhida ou não.

4.4.3. Processo de Otimização - Troca de Veículos Permitida

Essa opção apresenta dois tipos de busca de vizinhança. A primeira opção escolhe aleatoriamente uma jornada e tenta encontrar outro trecho de trabalho, através dos vetores *Prox* e *Ant*, para que possa ser feita outra combinação com algum dos trechos da jornada escolhida. A segunda opção seleciona duas jornadas aleatoriamente e tenta modificar os trechos de trabalho. Após as duas opções terem sido realizadas escolhe-se a que originar uma solução de menor custo. Vale ressaltar que após cada

opção ter sido executada o ATTON realiza um procedimento para verificar se é possível eliminar alguma sobre-cobertura de trechos de trabalho.

A seguir são apresentados detalhes dessas duas opções de modificação na solução.

- **Opção 1: Recombinação dos Trechos de Trabalho**

O ATTON escolhe, de forma aleatória, uma jornada *n* e verifica os vetores *Prox* e *Ant* correspondentes ao horário final do primeiro trecho de trabalho e ao horário inicial do segundo trecho de trabalho, respectivamente. Se houver outra OR viável, contida em um (ou nos dois) desses dois vetores, que atenda à restrição referente aos postos de controle, aos comprimentos do intervalo, ao tamanho máximo para uma jornada de trabalho, e que seja diferente da que já está sendo utilizada pela solução, o ATTON irá efetuar a substituição do trecho de trabalho correspondente. Caso exista OR(s) viáveis nos dois vetores, o programa escolhe aquele que apresentar um menor intervalo para a jornada *n*.

Basicamente o ATTON irá procurar outra OR viável disponível que possa ser utilizada a fim de originar outra jornada. Portanto, dada uma jornada *n*, com dois trechos de trabalho, o programa verifica se dentro do vetor *Prox*, correspondente ao horário final do 1º trecho de trabalho *Tfim1*, existe outra OR viável *t*' que seja diferente da que está sendo considerada atualmente (ou seja, diferente do horário inicial do 2º trecho de trabalho), que pertença ao mesmo posto de controle ao qual pertence *Tfim1* e que resulte um valor para o intervalo que não viole as restrições. Paralelamente, o programa irá verificar se dentro do vetor *Ant*, correspondente ao horário inicial do 2º trecho de trabalho *Tini2*, existe outra OR viável *t*'' que seja diferente da que está sendo utilizada no momento (ou seja, diferente de *Tfim1*) e que atenda às restrições já citadas para o caso da OR viável no vetor *Prox*.

Se existir OR(s) viáveis dentro dos dois vetores que atendam às condições mostradas acima, o ATTON escolherá a OR viável que apresentar o menor intervalo. Ou seja, caso a OR viável pertencente ao vetor *Prox* seja a escolhida o programa

manterá o primeiro trecho de trabalho e o combinará com o novo trecho de trabalho iniciado pela nova OR viável *t*' encontrada no vetor *Prox*. Caso contrário, se a OR viável pertencente ao vetor *Ant* tiver sido escolhida o programa manterá o segundo trecho de trabalho e o combinará com o novo trecho de trabalho que finaliza com a nova OR viável *t*'' encontrada no vetor *Ant*.

Se a jornada escolhida *n* apresentar um único trecho de trabalho, o programa verificará o vetor *Prox* e o vetor *Ant* para o horário final do único trecho de trabalho (o horário final da jornada) e para o horário inicial da jornada respectivamente, e adotará os mesmos processos mencionados anteriormente.

Se não for possível ocorrer nenhuma nova combinação, o programa permanece com a solução corrente e a utiliza para comparar com a solução construída a partir da opção 2 que será detalhada logo a seguir.

- **Opção 2: Troca dos Trechos de Trabalho entre Duas Jornadas**

Nessa opção, além da jornada escolhida *n* na opção anterior, o ATTON escolhe outra jornada de trabalho *n*', também de forma aleatória, verifica se as restrições referentes ao posto de controle (o posto de controle do horário final do primeiro trecho terá que ser igual ao posto de controle do horário inicial do segundo trecho de trabalho), ao comprimento máximo de uma jornada de trabalho e aos valores permitidos para o intervalo não são violadas e tenta combinar os trechos de trabalho das duas jornadas a fim de obter outras duas jornadas diferentes e, conseqüentemente, outra solução.

Portanto, se as restrições mencionadas anteriormente forem atendidas o programa realizará a troca entre os trechos de trabalho das duas jornadas. Através dessa opção é possível realizar quatro combinações possíveis: i) trocar o 1º trecho de trabalho da jornada *n* com o 1º trecho de trabalho da jornada *n*'; ii) trocar o 1º trecho de trabalho da jornada *n* com o 2º trecho de trabalho da jornada *n*'; iii) trocar o 2º trecho de trabalho da jornada *n* com o 1º trecho de trabalho da jornada *n*'; e, iv) trocar o 2º trecho de trabalho da jornada *n* com o 2º trecho de trabalho da jornada *n*'.

Semelhantemente à opção anterior, caso nenhuma troca seja possível, o ATTON considera a solução corrente e a compara com a solução gerada pela opção 1. Para essa opção o programa também considera a possibilidade de uma das jornadas escolhidas apresentar apenas um único trecho de trabalho. Caso as duas jornadas escolhidas apresentarem um único trecho de trabalho não será possível realizar nenhuma troca, pois a solução permanecerá a mesma.

4.4.4. Condições de Parada

As condições de parada são semelhantes às condições mais utilizadas em várias aplicações da metaheurística *Simulated Annealing* para a resolução de diversos tipos de problema.

A primeira condição de parada é o número de iterações externas, que é determinado pelo programador. O ATTON possibilita de 50 a 500 iterações externas. Para cada iteração externa o programa permite de 50 a 500 iterações internas.

Outra condição de parada para o processo *annealing* é a quantidade de iterações em que a melhor solução permanece inalterada. Essa opção pode ser determinada pelo programador no qual o mesmo especifica um valor percentual em relação ao número de iterações externas, determinando assim a quantidade de iterações em que a melhor solução pode permanecer inalterada. O programador também pode não especificar um valor e deixar que o ATTON considere o número total de iterações externas como valor especificado para a quantidade de vezes em que a melhor solução pode permanecer inalterada.

4.5. SOLUÇÃO FINAL E APRESENTAÇÃO DOS RESULTADOS

A solução final é apresentada da mesma forma que é apresentada a solução inicial. Além dos resultados de tempo e custo tanto para a solução global como para cada jornada, o ATTON também mostra alguns resultados relacionados ao processo da metaheurística *Simulated Annealing*. Esses resultados se referem à temperatura inicial e final, número de iterações externas, número de iterações em que a melhor solução não sofreu modificação, tempo total de execução e iteração da melhor solução.

O ATTON também apresenta, como resultados finais, um perfil do processo de otimização informando a temperatura, a taxa de aceitação de uma nova solução e o valor da função objetivo para a melhor solução local e a melhor solução global, em cada iteração externa. Essas informações associadas às citadas no parágrafo anterior servem para que se possa avaliar de forma mais detalhada a influência da metaheurística escolhida no processo de otimização, e, portanto têm apenas um caráter acadêmico.

4.5.1. Estudo de Caso

O ATTON foi aplicado para quatro linhas de ônibus do sistema de transporte coletivo por ônibus do município de Fortaleza. Os dados referentes à programação de veículos de cada linha, como também à alocação de motoristas e cobradores utilizada, até o presente momento, foram cedidos pela empresa Rotaexpressa Transportes de Passageiros.

As linhas escolhidas foram a 052 – Grande Circular 02, 600 – Messejana/Frei Cirilo/Expresso, 613 – Barroso/Jardim Violeta, e 631 – Carlos Albuquerque. Procurou-se escolher linhas que apresentassem um número relativo de veículos. Mesmo assim, optou-se por escolher também uma linha com apenas 1 (um) ônibus.

Antes de apresentar os resultados é importante ressaltar que a operação do sistema de transporte coletivo por ônibus no município de Fortaleza é caracterizada pela co-exploração das linhas. Ou seja, uma mesma linha de ônibus é explorada por mais de uma empresa. Por exemplo, para a operação nos dias úteis da semana, a linha 052 é co-explorada por 8 empresas com uma frota total de 48 veículos. Desse total, a empresa Rotaexpressa participa com apenas 6 ônibus. Portanto, vale ressaltar a dificuldade encontrada para a obtenção de linhas que operassem com um número elevado de ônibus. Todas as linhas escolhidas são exploradas por mais de uma empresa.

A Tabela 4.1 apresenta as características de cada linha escolhida para aplicação do ATTON, considerando-se apenas a empresa Rotaexpressa Transportes de Passageiros.

Tabela 4.1: Dados referentes às linhas de ônibus escolhidas.

Nome da Linha	Dia de Operação	Frota
052- Grande Circular 02	Dias Úteis	06
600 – Messejana/Frei Cirilo/Expresso	Dias Úteis	04
613 – Barroso/Jardim Violeta	Dias Úteis	03
631 – Carlos Albuquerque	Dias Úteis	01

Para suprir esta carência de linhas com um número expressivo de veículos na sua operação aplicou-se, novamente, o ATTON para a linha 052, considerando a frota total (48 veículos) e conseqüentemente a programação de veículos (para a operação em dias úteis) de todas as empresas que exploram a linha. A obtenção dessa programação foi possível através da OPL – Operacional de Linha. Esse documento, que contém toda a programação da linha, é enviado, pela ETTUSA, para todas as empresas que exploram a linha.

Essa nova aplicação foi necessária para que o programa ATTON pudesse ser avaliado diante de um cenário maior, no que se refere ao número de ônibus a ser considerado. Além disso, foi possível analisar melhor a influência da metaheurística *Simulated Annealing*. Porém, não foi possível comparar os resultados de custo fornecidos pelo ATTON e os considerados pelas empresas. Isso porque não foi possível obter as programações de motoristas e cobradores das diversas empresas que operam a linha 052. Os resultados dessa nova aplicação estarão contidos no Anexo 2.

Semelhantemente ao tipo de operação realizado pelas empresas operadoras desse tipo de sistema, foi considerada a opção de proibir a troca de veículos e de considerar os horários de lanche estabelecidos pela ETTUSA. Os dados referentes às restrições de comprimento de trechos de trabalho, intervalo, postos de controle para rendição e lanche foram definidos de acordo com as características de operação de cada linha e são apresentados através da Tabela 4.2, a seguir.

Tabela 4.2: Restrições adotadas para cada linha.

Linha	Comprimento Mínimo do Trecho (min)	Comprimento Máximo do Trecho (min)	Intervalo Mínimo (min)	Intervalo Máximo (max)	Postos de Rendição	Postos de Lanche
600	60	280	15	35	14 e 28	14
613	60	335	10	35	14 e 28	14
631	60	270	15	30	14 e 389	14
52	60	330	15	50	15,16,18 e 26	15,16 e 18

O programa ATTON foi testado em um computador com as seguintes configurações: AMD Duron 1300 MHz e 247 Mb de memória RAM, cedido pelo Departamento de Engenharia de Transportes da Universidade Federal do Ceará – UFC.

O ATTON foi aplicado para cada linha considerando três valores possíveis para o coeficiente de decréscimo α (0,90, 095 e 0,99). Neste item apenas são apresentados os resultados obtidos para uma única linha (linha 600 – Messejana/Frei Cirilo/Expresso). Os demais resultados estão contidos no Anexo 3.

Cada figura apresentada anteriormente (Figuras 4.1, 4.2, 4.3, 4.4, 4.5 e 4.7) representa os dados de entrada e as restrições necessárias para a construção da solução inicial, considerando a linha 600. Sendo assim as Figuras 4.10 e 4.11, a seguir, apresentam os resultados numéricos e gráficos da solução inicial, respectivamente.

Nos resultados numéricos (Figura 4.10) são mostrados tanto os valores totais de tempo e custo para a programação completa, como também os valores de tempo e custo para cada jornada criada. Vale ressaltar que o valor apresentado (em cor vermelha), referente ao custo com a ociosidade, considera o peso atribuído ao fator ociosidade. Porém, os outros valores de custo apresentados não estão sob o efeito dos pesos atribuídos a cada característica indesejável (hora-extra, ociosidade, jornada com apenas 1 (um) trecho de trabalho e sobre-cobertura). O resultado gráfico da Figura 4.11 contém as oportunidades de rendição que constituem as jornadas de trabalho criadas.

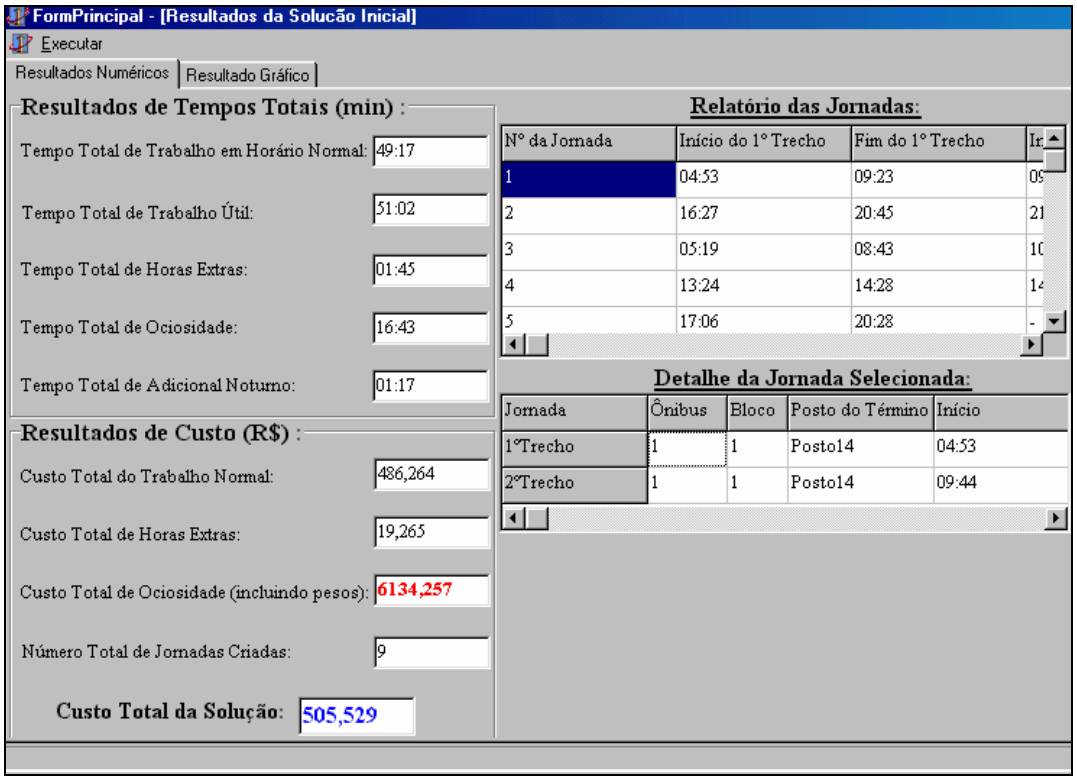

Figura 4.10: Tela dos resultados numéricos da Solução Inicial.

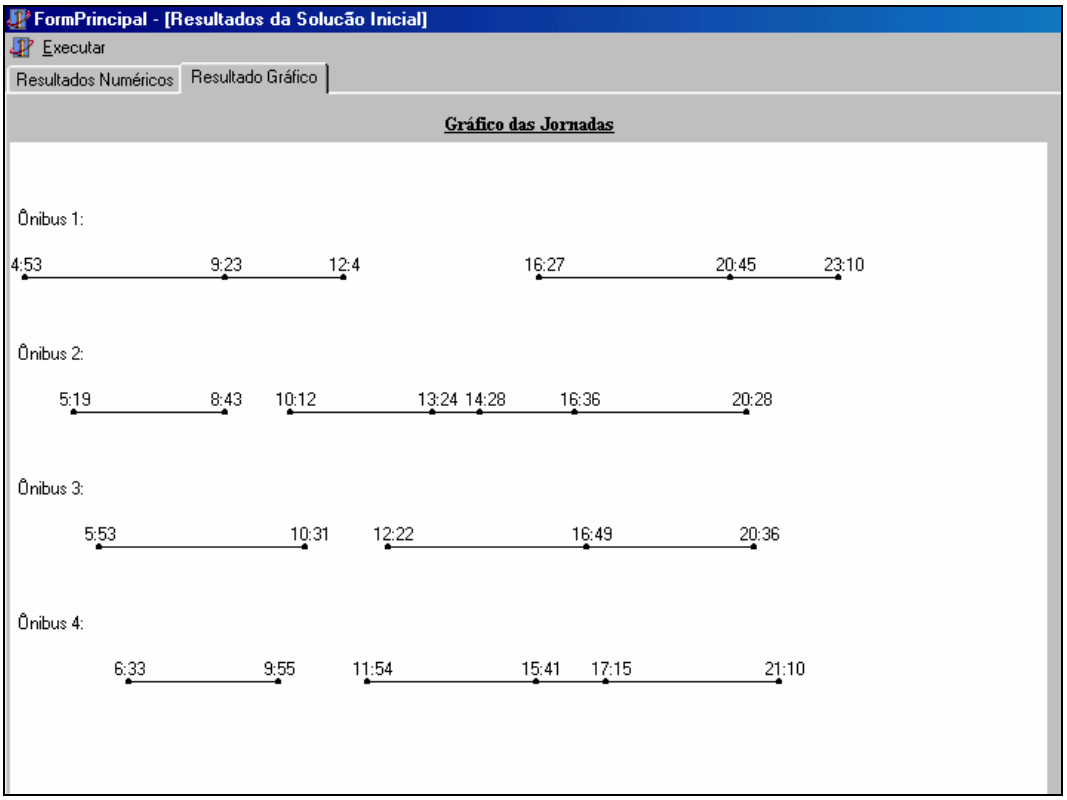

Figura 4.11: Resultado gráfico da solução inicial.

De acordo com a Figura 4.10, o ATTON apresenta os valores obtidos para o que é chamado de “Trabalho Normal” e o “Trabalho Útil”. O valor de trabalho normal compreende o valor de horas trabalhadas desconsiderando as horas-extras. O valor de trabalho útil considera o valor de trabalho normal acrescido do valor de horas-extras. O programa também apresenta um relatório de cada jornada onde ao clicar no número da jornada, o programador obtém um detalhe da jornada escolhida.

A tabela denominada de “Relatório das Jornadas” (ver Figura 4.10) apresenta o horário inicial e final de cada trecho de trabalho, o tamanho do intervalo, o comprimento (hh:mm) da jornada de trabalho (desconsiderando as horas-extras), o comprimento (hh:mm) da hora-extra e da ociosidade (se houver) e os custos de trabalho normal, hora-extra e ociosidade, de cada jornada. Ao escolher uma determinada jornada, o programa apresenta o ônibus, o bloco, o horário inicial e final de cada trecho da jornada escolhida.

Após a obtenção da solução inicial, o programador poderá decidir se continua, ou não, o processo de otimização ordenando ao ATTON que apresente a solução final, através da seleção do item “*Simulated Annealing*” no menu “Executar” na barra de ferramentas. O programador poderá escolher em executar novamente o ATTON para a mesma linha, apenas clicando no item “Executar outra Linha” que existe no menu “Executar”.

A solução final apresenta os resultados numéricos (Figura 4.12) e gráficos da solução gerada (Figura 4.13), como também os resultados relativos ao processo da metaheurística *Simulated Annealing* (Figura 4.14). Esses resultados servem para que se faça uma avaliação da influência da metaheurística no processo de otimização. Portanto, são apresentados (ver Figura 4.14) valores referentes à temperatura inicial e final do processo, tempo de execução, número da iteração em que a melhor solução foi encontrada, o número de iterações sem que houvesse uma melhora na solução e uma tabela contendo as taxas de aceitação de uma nova solução, a temperatura e a melhor solução local e global de cada iteração externa.

As Figuras 4.12, 4.13 e 4.14 apresentam os resultados obtidos para a linha 600, considerando um fator de decrescimento da temperatura (α) de 0,99. As Figuras 4.13 e 4.14 mostram, respectivamente, o resultado gráfico da solução final e o resultado referente ao processo de otimização.

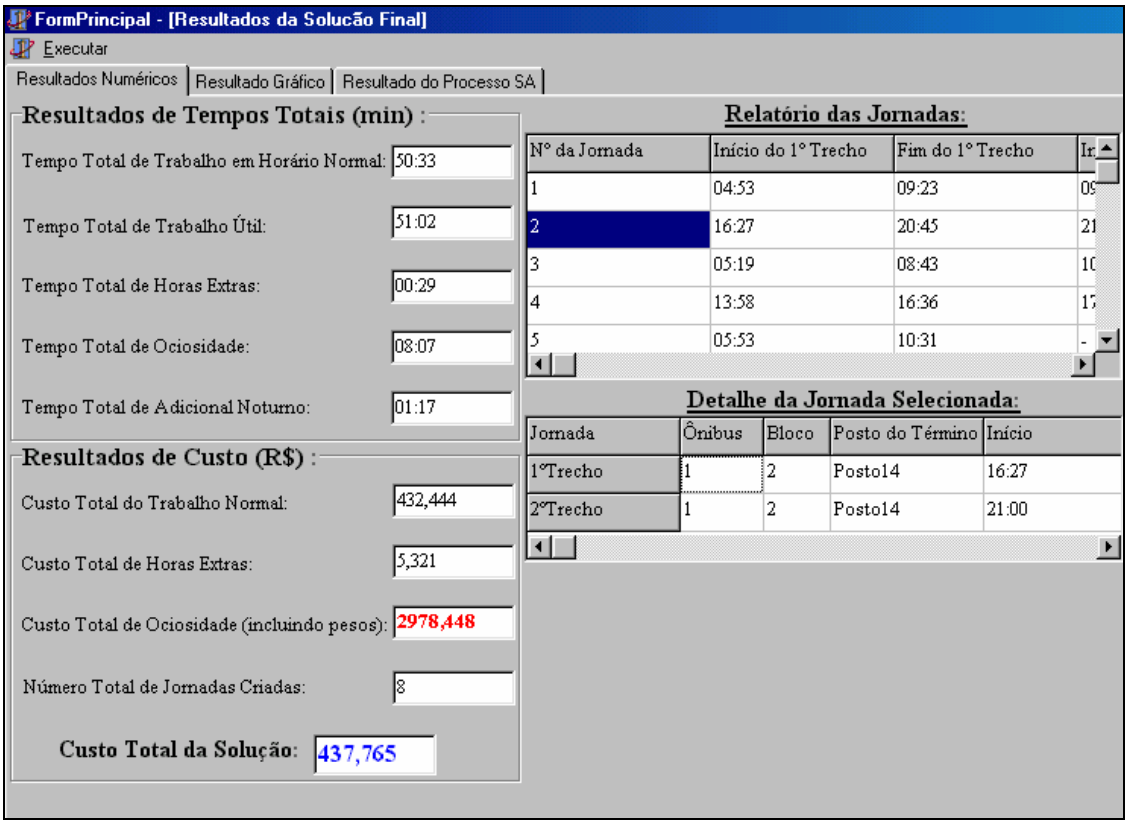

Figura 4.12: Resultados numéricos da solução final.

A tela que apresenta os resultados numéricos da solução final (Figura 4.12) é idêntica à apresentada para os resultados numéricos da solução inicial (Figura 4.10). Como pode ser verificado, houve uma redução de 9 para 8 jornadas de trabalho. O custo total do trabalho normal sofreu uma redução de 11,07%, o custo total de horas-extras de 72,38%, o custo total em ociosidade de 51,44% e o custo total da solução sofreu uma redução de aproximadamente 13,40%.

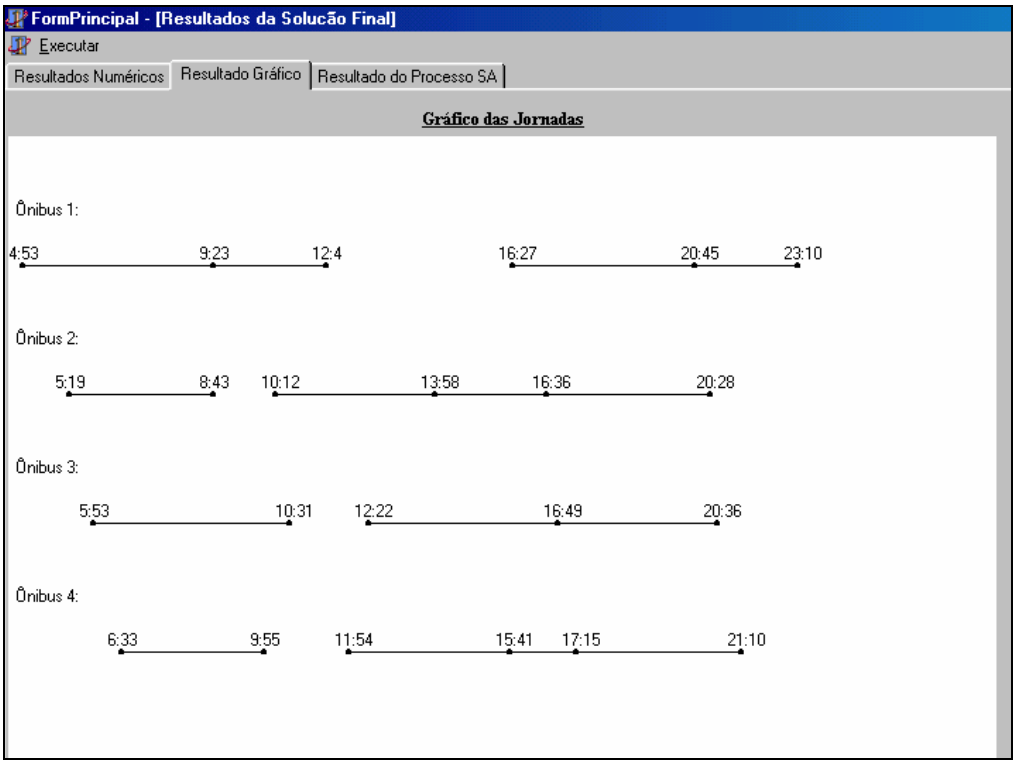

Figura 4.13: Resultado gráfico da solução final.

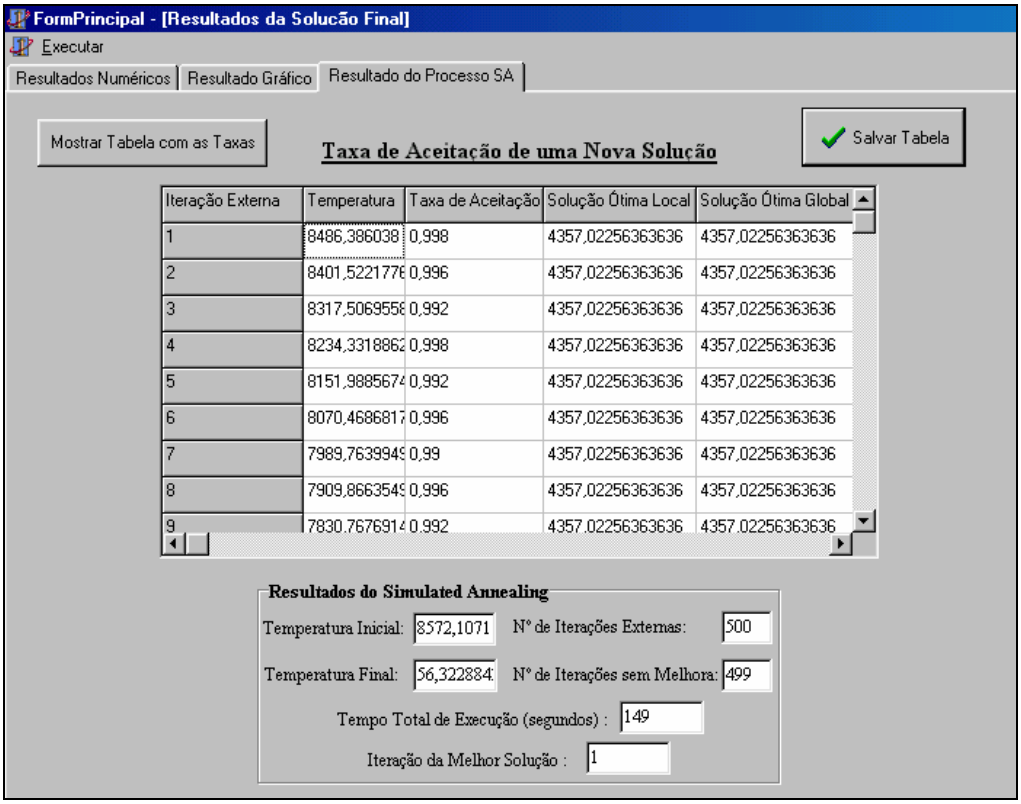

Figura 4.14: Resultado do processo de otimização.

Como pode ser verificado na Figura 4.14, o processo de otimização da metaheurística *Simulated Annealing* apresentou uma temperatura inicial e final de aproximadamente 8572,107 e 56,323 (temperatura final ainda bastante alta), respectivamente. O número de iterações externas foi igual a 500 e a iteração que obteve a melhor solução foi a de número 1 (um), resultando assim em 499 iterações sem que houvesse melhora na solução. Isso significa que a metaheurística *Simulated Annealing* teve pouca influência no processo de melhoramento da solução inicial. Isso se deve ao fato que a linha 600, operada pela empresa Rotaexpressa, apresenta apenas quatro ônibus.

Os resultados apresentados no Anexo 2 mostram que esse aspecto (baixa influência da metaheurística) diminui quando a linha apresenta um número relativo de ônibus, caracterizando assim como um problema de maior porte. Através dos resultados apresentados no Anexo 2 verificou-se que a melhor solução foi encontrada na iteração de número 63 e que a solução é bastante modificada, seja para pior ou para melhor, apresentando uma taxa de aceitação de novas soluções, praticamente constante ao longo de todo processo de otimização. Não foi possível comparar os resultados obtidos pelo programa com os adotados pelas empresas, visto que a linha 052 é operada por 8 empresas nos dias úteis da semana, e 7 empresas aos sábados, domingos e feriados. Mesmo assim, o teste se mostrou bastante válido para os propósitos ao qual tinha sido sugerido.

A seguir são apresentados, na Tabela 4.3, os resultados apresentados pela empresa para todas as linhas analisadas e os obtidos pelo programa. Os valores de custo apresentados são para a operação em um dia útil da semana.

Tabela 4.3: Comparação entre os valores adotados pela empresa e os obtidos com o programa.

Linhas / **Valores**	**Linha 052**		**Linha 600**		**Linha 613**		**Linha 631**	
	ATTON	**Empresa**	**ATTON**	**Empresa**	**ATTON**	**Empresa**	**ATTON**	**Empresa**
Custo Total (R$)	649,25	621,28	437,77	441,10	387,46	347,35	172,89	186,62
Diferença (%)	**4,50**		**0,75**		**11,54**		**7,36**	
Nº de Jornadas	12	11	8	8	7	6	3	3
Tempo Total (seg)	46		44		33		44	

Portanto, o ATTON apresentou uma redução no custo de aproximadamente 0,75% para a linha 600, e 7,36% para a linha 631. Porém, para as linhas 052 e 613, houve um aumento de, aproximadamente, 4,50% e 11,54%, respectivamente. Esses valores, acima do custo adotado pela empresa, deve-se a duas peculiaridades que são adotadas pela empresa e não consideradas pelo ATTON, e que serão tratadas mais adiante.

Algumas características tiveram que ser adotadas na solução apresentada pela empresa. Isso se deve ao fato da empresa adotar duas peculiaridades (fora do padrão "normal" do sistema) que ainda não são abordadas pelo ATTON. Trata-se da possibilidade do motorista trocar não apenas de veículo, mas também de linha e também da existência de jornadas de trabalho com mais de dois trechos de trabalho (no caso, três trechos de trabalho), configurando assim o que se chama de dupla-pegada. Portanto, as linhas que apresentaram essas peculiaridades tiveram que sofrer algumas modificações na estrutura da sua programação (apresentada pela empresa) para que pudessem ser comparadas à solução fornecida pelo ATTON.

No caso de troca de linha, considerou-se apenas a parcela da jornada correspondente à linha 600 e a considerou como uma jornada operada apenas em uma única linha. Sendo assim, essas jornadas apresentavam um tamanho menor do que às 07 horas e 20 minutos (tamanho ideal), caracterizando assim uma ociosidade maior para a solução. O ATTON ainda não trabalha com mais de uma linha de ônibus. Para o caso de jornadas com mais de dois trechos de trabalho, o ATTON fornece uma solução final com um número de jornadas superior ao apresentado pela empresa. Essa diferença é de apenas uma única jornada.

Mesmo assim, o ATTON fornece informações suficientes para que o programador perceba essas peculiaridades e possa realizar pequenos ajustes manuais na solução final, fornecida pelo programa, de forma a obter um melhor aproveitamento dos recursos destinados pela empresa. As Tabelas 4.4 e 4.5 apresentam, respectivamente, as jornadas criadas pela empresa e as jornadas criadas pelo programa para a linha 600.

Tabela 4.4: Resultado da Empresa.

Jornada	Início	Término	Comprimento da Jornada (hh:mm)	Hora-Extra (hh:mm)	Período Noturno (hh:mm)	Custo (R$)
1	04:53	12:04	06:50	00:00	00:07	53,99
2	16:27	23:10	06:28	00:00	01:10	55,53
3	05:19	13:24	06:36	00:00	00:00	53,82
4	13:24	20:28	06:34	00:00	00:00	53,82
5	05:53	10:31	04:38	00:00	00:00	53,82
6	12:22	20:36	07:49	00:29	00:00	59,14
7	06:33	15:41	07:09	00:00	00:00	53,82
8	15:41	21:10	04:58	00:00	00:00	53,82

Tabela 4.5: Resultado do Programa.

Jornada	Início	Término	Comprimento da Jornada (hh:mm)	Hora-Extra (hh:mm)	Período Noturno (hh:mm)	Custo (R$)
1	06:33	16:11	07:39	00:19	00:00	57,32
2	12:22	20:36	07:49	00:29	00:00	59,08
3	16:11	21:10	04:59	00:00	00:00	53,80
4	05:53	10:31	04:38	00:00	00:00	53,80
5	04:53	12:04	07:11	00:00	00:07	53,99
6	05:19	13:24	06:36	00:00	00:00	53,80
7	13:24	20:28	06:34	00:00	00:00	53,80
8	16:27	23:10	06:28	00:00	01:10	55,53

O objetivo da apresentação das tabelas 4.4 e 4.5 é tentar exemplificar uma das peculiaridades citadas anteriormente, a da existência de uma única jornada de trabalho estar sendo operada em linhas diferentes. Verifica-se na Tabela 4.5 que existem duas jornadas (3 e 4) com durações de apenas 4:59 hs e 4:38 hs, respectivamente. Na programação atual da empresa (Tabela 4.4), essas duas jornadas são combinadas com outras jornadas presentes em outras linhas para que haja um melhor aproveitamento dos recursos e para que não existam jornadas de trabalho muito curtas. Porém, foi realizada uma modificação nessa programação atual da empresa de forma a considerar essas jornadas como se as mesmas fossem exclusivamente da linha 600. Essa modificação foi necessária para que fosse possível realizar as comparações de custo entre a programação da empresa e a solução fornecida pelo ATTON.

Verifica-se que essas duas jornadas de trabalho (3 e 4) existentes na programação da empresa (Tabela 4.4) correspondem aproximadamente às jornadas 5 e 8 criadas pelo ATTON (Tabela 4.5). Vale ressaltar que as tabelas 4.4 e 4.5 não

apresentam os valores de intervalo de cada jornada de trabalho. Portanto, diante desses resultados e de sua experiência, o programador poderá realizar, a partir dos resultados de cada linha operada pela empresa, alterações manuais para combinar jornadas curtas de duas linhas diferentes a fim de obter uma única jornada e, conseqüentemente, uma melhor aplicação dos recursos destinados à operação do sistema. Essa combinação terá que ser feita para que se reduza a ociosidade gerada pela produção de jornadas muito curtas, como mostrado nas tabelas 4.4 e 4.5. O Anexo 4 apresenta os resultados das jornadas criadas para as outras linhas estudadas.

O ATTON também informa resultados relativos ao processo de otimização. Esses resultados se referem à temperatura de cada iteração, taxas de aceitação de novas soluções em cada iteração, melhor solução de cada iteração (solução ótima local), solução ótima global e o número da iteração na qual a melhor solução foi encontrada. As Figuras 4.15 e 4.16 apresentam, através de gráficos, o comportamento das taxas de aceitação e da temperatura ao longo do processo de otimização.

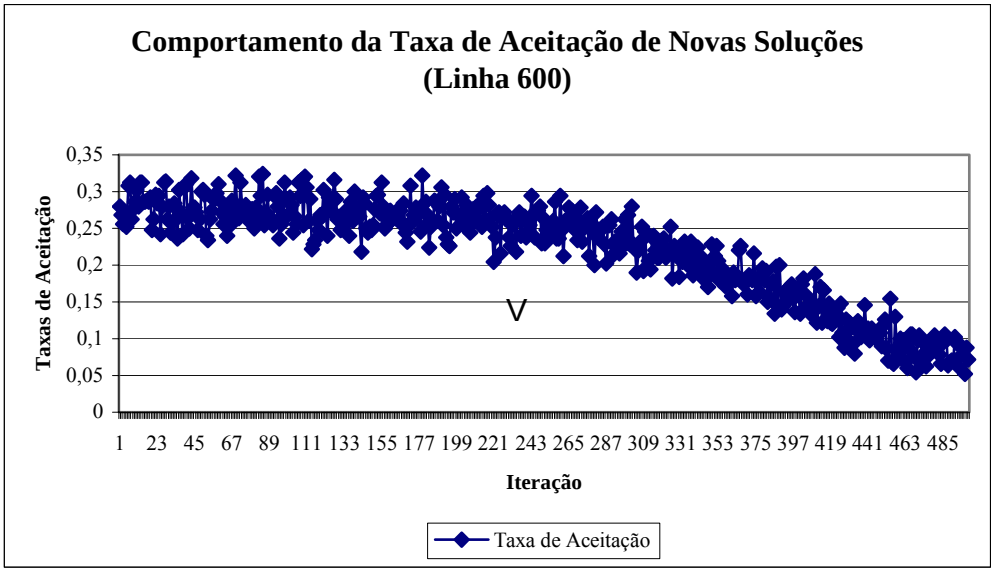

Figura 4.15: Comportamento das taxas de aceitação de novas soluções.

Através da Figura 4.15 pôde-se verificar que a taxa de aceitação de novas soluções inicia com valores baixos (na ordem de 0,30). Isso se deve ao fato de que o processo de melhoramento da solução no ATTON apenas considera soluções viáveis.

Portanto, se o programa não puder realizar nenhuma modificação na jornada escolhida que leve a uma solução geral viável, o programa passa para a próxima iteração.

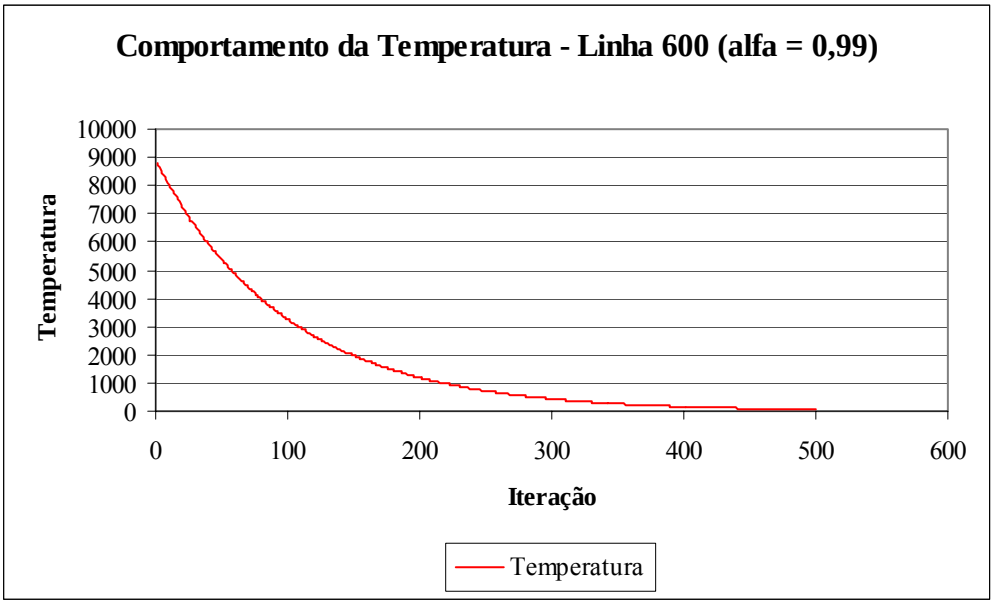

Figura 4.16: Comportamento da Temperatura.

Verifica-se que ao decorrer do processo de otimização, ocorre um resfriamento do sistema (decréscimo da temperatura – Figura 4.16) e conseqüentemente uma diminuição da taxa de aceitação de novas soluções (boas ou ruins). Isso se deve ao fato de que a probabilidade de modificação da solução depende da diferença do valor da função objetivo das duas soluções que estão sendo analisadas e da temperatura. Portanto, quanto mais baixa a temperatura, menor a probabilidade de aceitação de novas soluções.

4.6. CONSIDERAÇÕES FINAIS

O ATTON apresenta uma interface amigável e bastante simples para ser manuseada. O programador não tem que clicar em vários botões, pois o programa já apresenta valores padrões para diversas restrições do problema. Com isso, o programador fica menos propenso a erros relacionados à entrada de dados.

O ATTON não só apresenta resultados referentes às jornadas de trabalho produzidas, mas também resultados gráficos que podem auxiliar o programador na

obtenção de uma solução ótima. O programador tem total liberdade para decidir se deve continuar, ou não, o processo de otimização quando diante de alguns resultados preliminares. Sendo assim, o ATTON permite uma constante interação entre o programador e o modelo computacional. Apesar da opção que permite a troca de veículos ainda não estar totalmente concluída, o resultado obtido pela adoção dessa opção pode servir como um resultado preliminar, de modo que o programador possa idealizar outras formas de operação para a linha que está sendo analisada.

O ATTON ainda apresenta algumas limitações que poderão ser superadas posteriormente e que serão discutidas e apresentadas com mais detalhe no capítulo 5. Porém, o mesmo se mostrou capaz de ser aplicado em diferentes empresas operadoras do sistema de transporte coletivo por ônibus. Com isso, o ATTON ainda necessita de algumas modificações no sentido de torná-lo mais robusto e eficiente. Porém, isso não o impede de ser aplicado e utilizado em diversas empresas operadoras do sistema de transporte coletivo por ônibus do município de Fortaleza.

CAPÍTULO 5

CONCLUSÕES E RECOMENDAÇÕES

O uso de procedimentos computacionais na solução de problemas de alocação de tripulação (motoristas e cobradores) em transporte coletivo possibilita ao programador a obtenção de vários cenários de operação do sistema em um tempo relativamente pequeno, se comparado ao tempo gasto no processo manual de obtenção dessa programação. Ainda não é possível avaliar o programa ATTON quanto ao tempo gasto para a obtenção da programação geral da empresa. Para isso, seria necessário aplicar o programa para todas as linhas da empresa e, depois de realizadas as modificações manuais necessárias, contabilizar o tempo total gasto, desde do início da aplicação do programa até a nomeação de cada dupla (motorista e cobrador) para cada jornada de trabalho visto que o ATTON apenas cria as jornadas de trabalho, deixando a tarefa de alocar o motorista e o cobrador para o programador da empresa.

Diante de várias soluções, o programador poderá avaliar e escolher melhor qual programação adotar e, conseqüentemente, minimizar os custos com a operação do sistema. Além disso, a comparação das diversas soluções geradas pelo procedimento computacional poderá auxiliar na elaboração de uma nova solução que possa ser melhor do que as geradas pelo computador.

O ATTON fornece ao programador uma série de resultados em um tempo hábil de forma que o programador tem total liberdade para interagir com o sistema a fim de aceitar (ou não), e possivelmente continuar o processo, os resultados que estão sendo obtidos. Caso contrário, o programador poderá facilmente retornar ao ponto de partida e reiniciar uma nova tentativa no objetivo de alcançar um resultado melhor.

O programa ATTON se caracteriza pela simplicidade no seu manuseio. Não se faz necessário o clique de vários botões para que se chegue a uma solução. Valores padrões de restrições já são incorporados ao programa de modo a minimizar o surgimento de erros devido à entrada de dados. Porém, é necessário que o programador

tenha um conhecimento prévio sobre o problema de alocação de tripulação para que não sejam fornecidos valores incoerentes em relação a comprimentos de trechos de trabalho e comprimentos de intervalo.

O ATTON também trabalha com dois cenários de operação. O primeiro aborda a proibição da troca de veículos dentro de uma mesma linha (o programa não considera mais de uma linha). Essa forma de operação do sistema é a preferida na realidade brasileira devido a acordos feitos entre empresários e funcionários (motoristas e cobradores) e aspectos relacionados ao controle da operação pela empresa, já mencionados em capítulos anteriores. A segunda opção permite ao motorista e ao cobrador trocarem, ou não, de veículo em uma mesma linha. Essa segunda opção não é tão aceita pelos motoristas e cobradores, pois os mesmos não desejam passar tanto tempo vinculado à empresa. Essa opção acaba gerando um comprimento maior para o intervalo e assim o motorista se sente impossibilitado de procurar outra fonte de renda.

Portanto, diante dessas duas opções, o programador poderá observar diferentes soluções e escolher a melhor, ou até mesmo realizar uma combinação entra as soluções fornecidas por cada cenário. Mesmo a segunda opção necessitando ainda de alguns refinamentos na sua solução final, o programador dispõe de informações que poderão levá-lo a criação de uma boa programação.

Existem duas peculiaridades, tratadas inicialmente no Capítulo 4, que ainda não são consideradas pelo programa proposto neste trabalho. Trata-se da consideração de duas linhas, ou mais, simultaneamente e da criação de jornadas de trabalho com mais de dois trechos.

É preciso realizar algumas modificações na estrutura de leitura da programação de veículos para que o ATTON considere mais de uma linha de ônibus simultaneamente. Mesmo assim, essa consideração não impede que soluções ótimas sejam fornecidas e que o programador não possa, através delas, estabelecer uma programação geral para todas as linhas da empresa que satisfaça os objetivos da mesma, no tocante à redução de custos com pessoal de operação.

Como o ATTON fornece soluções em um tempo razoavelmente pequeno, é possível que o programador, diante das soluções geradas para cada linha, realize uma análise a fim de descobrir se é necessária alguma combinação para que o motorista e o cobrador operem em mais de uma linha. Porém, em visita a algumas empresas e ao analisar alguns trabalhos já publicados, verifica-se que a troca de veículos é um procedimento indesejável não só pela empresa como também pelos motoristas e cobradores. Isso se deve a características já apresentadas na revisão bibliográfica no Capítulo 2.

A outra peculiaridade (jornadas com mais de dois trechos de trabalho) também é um procedimento indesejável por parte dos motoristas e cobradores. Essa peculiaridade pode ser chamada de dupla pegada. Para o motorista e o cobrador é preferível que os mesmos trabalhem o primeiro trecho da sua jornada, tenham seu intervalo e em seguida trabalhem o restante que falta para cumprir a jornada de trabalho. Assim, os mesmos estarão vinculados à empresa apenas o tempo necessário e, posteriormente, poderão exercer outra atividade para que possam aumentar as suas rendas. Sabe-se que vários motoristas e cobradores exercem outras atividades quando estão fora do horário de trabalho. Muitos são taxistas e até mesmos motoristas de transporte alternativo.

A maioria dos métodos computacionais criados tenta minimizar a criação de jornadas com mais de dois trechos de trabalho. Portanto, o ATTON procurou inicialmente desconsiderar este tipo de jornada de trabalho devido a uma maior complexidade no seu tratamento. Dessa forma, o programa poderá, se for o caso, sofrer algumas modificações para que possa atender esse tipo de jornada. Mesmo assim, o programador poderá assumir o mesmo comportamento indicado para o tratamento da outra peculiaridade (troca de linhas) citado anteriormente. Ou seja, diante das várias soluções, é possível verificar quais jornadas poderiam ser combinadas para que se tornassem jornadas de trabalho com mais de dois trechos. Vale ressaltar, que se tratam de jornadas de trabalho com no máximo três trechos de trabalho.

Pode ser verificado, através dos resultados apresentados no Anexo 4, que algumas soluções apresentam jornadas de apenas um único trecho de trabalho e que podem ser combinadas com jornadas de dois trechos sem que excedam às 7:20 hs mais

2:00 horas-extras. Como por exemplo, no caso da linha 613 – Barroso/Jardim Violeta com as jornadas 6 e 7 (obtidas pelo programa), com as jornadas 7, 8 e 9 da linha 052 – Grande Circular 02, que poderiam ser recombinadas para que fosse criada uma jornada com mais de dois trechos de trabalho e com isso igualar, em termo de jornadas, a programação obtida pelo programa e a adotada pela empresa.

Outra característica verificada com o uso do ATTON foi a baixa influência da metaheurística *Simulated Annealing*. Isso se deve tanto a grande quantidade de restrições relacionadas aos horários que podem ser alterados no processo de melhoramento da solução, como também ao pequeno tamanho dos problemas em análise. Como mencionado no capítulo 4, a maioria das linhas de ônibus no município de Fortaleza é explorada por mais de uma empresa e por isso as mesmas apresentam linhas com um baixo número de veículos, resultando assim em um problema de pequeno porte. Contudo, aplicou-se o ATTON na linha 052 – Grande Circular 02 considerando todas as empresas que co-exploram essa linha, e obteve-se resultados mais significativos relacionados à influência da metaheurística. Os resultados apresentados no Anexo 2 comprovam a afirmação de que para problemas maiores pode-se verificar uma maior influência da metaheurística SA.

Quanto aos resultados, o ATTON se mostrou bastante satisfatório. O programa apresentou redução no custo da programação, em algumas linhas analisadas, de aproximadamente 7%, como no caso da linha 631, e de aproximadamente 0,75% como no caso da linha 600. A presença de uma jornada a mais no resultado fornecido pelo programa, em algumas linhas analisadas, deve-se a existência de jornadas de trabalho, existentes na programação adotada pela empresa, com mais de dois trechos de trabalho. Vale salientar que para todas as linhas analisadas foram supridas as características de troca de linhas a fim de que os resultados pudessem ser melhor comparados, considerando assim a jornada de trabalho com apenas um único trecho.

A seguir são abordadas algumas limitações e possíveis recomendações sobre o programa construído.

5.1. LIMITAÇÕES E RECOMENDAÇÕES

5.1.1. Limitações

a) O ATTON não considera mais de uma linha de ônibus simultaneamente.

b) Apenas são consideradas jornadas de trabalho com no máximo dois trechos de trabalho.

c) O ATTON pode ser testado para o caso em que a troca de veículos é proibida e os horários de lanche não são os estabelecidos pela ETTUSA. Os resultados obtidos a partir da escolha dessas opções apenas servem para que o programador possa ter uma alternativa de outros horários de lanche possíveis. Porém, seria necessário acrescentar algumas considerações no programa para que o mesmo pudesse atuar na programação de veículos a fim de evitar intervalos nos períodos de pico.

d) A estrutura do ATTON é um pouco influenciada pela forma em que é operado o sistema de transporte coletivo por ônibus no município de Fortaleza. Ainda não se tem conhecimento se outros municípios adotam o sistema de postos de controle.

e) Para o caso em que a troca de veículos é permitida, a solução final é influenciada pela solução inicial. Isso se deve ao fato de que depois de criados os trechos de trabalho, o processo de otimização apenas realiza trocas entre os trechos, preservando assim características como, por exemplo, trechos de trabalho curtos. Não há uma opção de modificação na solução que altere os horários finais e iniciais de cada trecho, como é visto na opção em que a troca de veículos é proibida. Portanto, se a solução inicial apresentar trechos de trabalho muito curtos, esses trechos estarão presentes na solução final, pois a solução final não pode apresentar nenhum trecho de trabalho descoberto. Porém, esse problema pode ser sanado com a especificação de um valor, não muito pequeno, para o comprimento mínimo do trecho de trabalho.

f) A solução final apresentada, com a escolha da opção de permitir a troca de veículos, pode, em alguns casos, apresentar sobre-cobertura. Para isso, faz-se necessário decidir sobre a modificação na estrutura do processo de melhoramento da solução ou sobre o incremento de um processo de refinamento da solução final gerada quando da escolha dessa opção.

5.1.2. Recomendações

As recomendações sugeridas para este trabalho são no sentido de tornar o programa mais robusto e eficiente. Portanto, são recomendações no sentido de incrementar o processo de otimização.

a) Como citado no Capítulo 4, a formação dos trechos de trabalho (na opção de troca de veículos permitida) só é finalizada quando as restrições relacionadas ao posto de controle são atendidas. Ou seja, o número do posto de controle referente ao horário final do 1º trecho de trabalho tem que ser igual ao número do posto de controle em que se encontra o horário inicial do 2º trecho da jornada de trabalho. Essa restrição poderia ser retirada e, paralelo a isso, ser criada uma forma de contabilizar o custo devido a esse deslocamento, entre os postos de controle, por parte da dupla (motorista e cobrador). Acredita-se que com essa medida, o número de combinações entre os diversos trechos de trabalho seria maior. Mesmo o ATTON tendo que formar os trechos de trabalho quantas vezes for necessário até que as restrições sejam atendidas, verificou-se uma baixa repetição desse processo. Ainda não há nenhum procedimento, dentro do programa, para contabilizar o custo com esse processo.
b) Faz-se necessário criar uma rotina que leia a programação de veículos diretamente da Ordem de Serviço Operacional – OSO, para que não seja necessária a criação de um arquivo texto contendo toda a programação. A digitação desse arquivo texto seria um pouco trabalhosa para casos em que a linha de ônibus apresente um número elevado de veículos.
c) Também se faz necessária uma apresentação dos resultados através de relatórios com todos os valores de tempo e custo, não só da solução como um todo, como também de cada jornada. O programa apenas cria alguns arquivos textos que apresentam as jornadas criadas e o comportamento da taxa de aceitação de novas soluções.

5.2. CONSIDERAÇÕES FINAIS

Espera-se que o ATTON auxilie na tomada de decisão dos responsáveis pela área de tráfego da empresa, de modo que os mesmos alcancem uma melhor alocação do pessoal de operação, minimizando assim o custo e, conseqüentemente, revertendo essa redução para a população carente do sistema de transporte coletivo por ônibus do Brasil, através de uma tarifa mais acessível para essa parcela da população.

Além disso, também é possível estudar a possibilidade de acrescentar outras metaheurísticas ao programa ATTON. Com isso, o programador poderá determinar qual metaheurística será escolhida para realizar o processo de melhoramento da solução inicial. Também será possível avaliar a qualidade das metaheurísticas empregadas na resolução de um único tipo de problema.

Finalmente, mesmo com suas limitações, o ATTON se mostra viável para ser aplicado na realidade brasileira, principalmente nos municípios que apresentem características semelhantes às adotadas em Fortaleza-CE, e capaz de fornecer resultados bastante satisfatórios.

REFERÊNCIAS BIBLIOGRÁFICAS

AZEVEDO FILHO, M. A. N. (1997) Genetic Algorithm for Bus Driver Schedulling. Relatório Técnico não publicado.

AZEVEDO FILHO, M. A. N.; KWAN, R. S. K. e WREN, A. (1994) A Alocação de Ônibus e Motoristas no Brasil: Alguma Experiência Prática, *Anais do VIII Congresso de Pesquisa e Ensino em Transportes*, vol. II, Recife, pp. 231-242, ANPET.

BARCELOS, J. C. H. de (2000) Algoritmos Genéticos Adaptativos: Um Estudo Comparativo. Dissertação de Mestrado. Escola Politécnica da Universidade de São Paulo, São Paulo.

BARROS NETO, J. F. (1997) Análise de Desempenho dos Operadores Genéticos Aplicados ao Problema do Caixeiro Viajante. *Anais do XXIX Seminário Brasileiro de Pesquisa Operacional* – SBPO, Salvador-BA.

BEASLEY, D.; BULL, D. R. e MARTIN, R. R. (1993) An Overview of Genetic Algorithms: Part 1, Fundamentals e Part 2, Research Topics.

BLAIS, J. Y.; LAPORTE, G.; LESSARD, R.; ROUSSEAU, J. –M. e SOUMIS, F. (1976). The problem of assigning drivers to bus routes in an urban transit system. Report, nº 44, Centro de Pesquisa em Transportes, Universidade de Montreal.

BLUM, C. e ROLI, A. (2001) Metaheuristics in Combinatorial Optimisation: Overview and Conceptual Comparison. *Technical Report* TR/IRIDIA/2001-13, IRIDIA,Université Libre de Bruxelles, Belgium, 2001.

BRAGA, C. M. da S.; GOMES, G. C. e VELARDE, L. G. C. (1994) Busca Tabu.

BRAZ JUNIOR, O. O. (2000) Otimização de Horários em Instituições de Ensino Através de Algoritmos Genéticos. Dissertação de Mestrado, UFSC.

BUSSETI, F. (2001) Genetic Algorithms Overview. www.geocities.com/francorbusetti.

BUSSETI, F. (2001a) Simulated Annealing Overview. www.geocities.com/francorbusetti.

CAPRARA, A.; FISCHETTI, M.; TOTH, P. e VIGO, D. (1995) Modeling and Solving the Crew Rostering Problem. Technical Reports, DEIS – OR – 95 – 6(R), Universidade de Bologna, Itália.

CAPRARA, A.; FISCHETTI, M. e TOTH, P. (1995a) A Heuristic Method for the Set Covering Problem. Technical Reports, DEIS, Universidade de Bologna, Itália.

CORRÊA, E. S. (2000) Algoritmos Genéticos e Busca Tabu Aplicados ao Problema das P-medianas. Dissertação de Mestrado. Universidade Federal do Paraná-UFP, Curitiba.

CUNHA, C. B. (1992) O Problema de Escala de Pessoal Operacional no Transporte Coletivo Urbano por Ônibus, *Anais do VI Congresso de Pesquisa e Ensino em Transportes*, Rio de Janeiro pp. 609-622, ANPET.

CURTIS, S. D. (2000) Constraint Satisfaction Approaches to Bus Driver Scheduling. Tese de Doutorado. Universidade de Leeds.

DORIGO, M. e STUTZLE, T. (2000) The Ant Colony Optimisation Metaheuristics: Algorithms, Applications, and Advances. *Technical Report* TR/IRIDIA/2000-32, IRIDIA, Université Libre de Bruxelles, Belgium, 2000.

DORIGO, M.; MANIEZZO, V. e COLORNI, A. (1986) The Ant System: Optimisation by a Colony of Cooperating Agents. Transactions on Systems, Man and Cybernetics – Part B, 26, 1, 29-41. http://iridia.ulb.ac.be/dorigo/dorigo.html

DORIGO, M. e DI CARO, G. (1999) The Ant Colony Optimization Meta-Heuristic. IRIDIA, Université Libre de Bruxelles.

EBTU (1988) *Gerência do Sistema de Transporte Público de Passageiros* – STPP Módulos de Treinamento (nº1 – nº8). Ministério da Habitação, Urbanismo e Meio Ambiente, Brasília, DF.

ELIAS, S. E. G. (1964) The use of digital computers in the economic scheduling for both man and machine in public transportation. Special report nº 49, Kansas State University Bulletin.

ETTUSA (2001) *Anuário de Transportes Urbanos de Fortaleza*. Empresa de Trânsito e Transporte Urbano S/A e Prefeitura Municipal de Fortaleza.

FEO, T. A. e RESENDE, M. G. C. (1995) Greedy Randomized Adaptative Search Procedures. *Journal of Global Optimization*, 6, 109-133.

FEO, T. A. e RESENDE, M. G. C. (1989) A Probabilistc Heuristic for a Computationally Difficult Set Covering Problem. ORL 8,67-71.

FERRAZ, A. C. P. e TORRES, I. G. E. (2001) *Transporte Público Urbano*. Ed. Rima, São Carlos.

FORES, S. (1996) Column Generation Approaches to Bus Driver Scheduling. PhD Thesis. Universidade de Leeds.

FORES, S.; PROLL, L. e WREN, A. (1997) An Improved ILP System for Driver Scheduling. *VII International Conference on Computer-Aided Scheduling of Public Transport*. Boston, 1997.

FORES, S.; PROLL, L. e WREN, A. (2000) Experiences with a Flexible Driver Scheduler. CASPT2000, Junho de 2000, Berlim.

FORSYTH, P. e WREN, A. (1997) An Ant Systems for Bus Driver Scheduling. VII International Workshop on Computer-Aided Scheduling of Public Transport, Boston, 1997.

FRELING, R,; HUISMAN, D. e WAGELMANS, A. P. M. (2000) Applying an Integrated Approach to Vehicle and Crew Scheduling in Practice. *VIII International Conference on Computer-Aided Scheduling of Public Transport*. Berlim, 2000.

FRELING, R.; HUISMAN, D. e WAGELMANS, A. P. M. (2000a) Models and Algorithms for Integration of Vehicle and Crew Scheduling. Econometric Institute Report EI2000-10/A, Erasmus University Rotterdam, Netherlands.

GOLDBARG, M. C. e LUNA, H. P. L. (2000) *Otimização Combinatória e Programação Linear: Modelos e Algoritmos*. Ed. Campus, Rio de Janeiro.

GLOVER, F. (1986) Future Paths for Integer Programming and Links to Artificial Inteligence. *Computers and Operations Research* 13, pp. 533-549.

GLOVER, F. e LAGUNA, M. (s.d.) Tabu Search. www.geocities.com/francorbusetti.

GOSS, S.; ARON, S.; DENEUBOURG, J. L. e PASTEELS, J. M. (1989) Self-Organized Shortcuts in the Argentine Ant. Naturwissenschaften, 76: 579 – 581, 1989.

HANSEN, P. E MLADENOVIC, N. (1997) Variable Neighborhood Search for the p-Median. Les Cahiers du Gerad, G-97-39.

HERTZ, A.; TAILLARD, E. e DE WERRA, D. (1992) A Tutorial on Tabu Search. Centre de Recherche sur les Transports, Montreal, Canada H3C 3J7, Universidade de Montreal.

HILDYARD, P. M. e WALLIS, H. V. (1981) Advances in computer-assisted runcutting in North America. In: Computer Scheduling of Public Transport, ed. Wren, A., pp. 183-192. North-Holland Publishing Company.

HOLLAND, J. H. (1975) Adaptation in Natural and Artificial System. MIT Press.

KIRKPATRICK, S.; GELLAT, D. C. e VECCHI, M. P. (1983) Optimization by Simulated Annealing. *Science*, pp. 671-680.

KWAN, A. S. K., KWAN, R. S. K., PARKER, M. E. e WREN, A. (2000) Proving the Versatility of Automatic Driver Scheduling on Difficult Train & Bus Problems.

KWAN, A. S. K.; KWAN, R. S. K. e WREN, A. (1997) Driver Scheduling using Genetic Algorithms with Embedded Combinatorial Traits. *VII International Conference on Computer-Aided Scheduling of Public Transport*. Boston, 1997.

LAYFIELD, C.; SMITH, B. e WREN, A. (1998) Bus Relief Point Selection Using Constraint Programming. *4th ILOG International Users Meeting*, Paris, France, 1998. http://citeseer.nj.nec.com.

LESSARD, R.; ROUSSEAU, J. –M. e DUPUIS, D. (1981) "Hastus I: A mathematical programming approach to the bus driver scheduling problem". In: Computer Scheduling of public transport urban passenger vehicle and crew scheduling, ed. Wren, A., Elsevier, Amsterdam.

LUEDTKE, L. K. (1985) RUCUS II: a review of system capabilities. In: Computer Scheduling of Public Transport 2, ed. Rousseau, J. –M., pp. 61-116.North-Holland Publishing Company.

MANIEZZO, V. e CARBONARO, A. (1999) Ant Colony Optimization: Na Overview. Scienze dell'Informazione, Universidade de Bologna, Itália.

METROPOLIS, N., ROSENBLUTH, A., ROSENBLUTH, M., TELLER, A., e TELLER, E. (1953) Equation of State Calculations by Fast Computing Machines, Journal of Chemical Physics, 21, 1087-1092.

NORONHA, T.F.; DA SILVA, M.M.; ALOISE, D.J. (2001) Uma Abordagem sobre Estratégias Metaheurísticas. Revista Eletrônica de Iniciação Científica (REIC), Ano I, Vol I, No. I, 2001.

NTU (1998) Anuário de Transportes Urbanos 97/98. Associação Nacional de Transportes Urbanos.

OSMAN, I. H. e LAPORTE, G. (1996) Metaheurístics: A bibliography. Annals of Operations Research, 63:513-623.

PARKER, M. E. e SMITH, B. (1981) Two approaches to computer crew scheduling. In: Computer Scheduling of Public Transport, ed. Wren, A., pp. 193-221. North-Holland Publishing Company.

PEREIRA, L. C. de S. N. (1985) Avaliação do Desempenho de Sistemas de Ônibus Urbanos. Brasília: EBTU.

REEVES, C. R. (1993) *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publications, Osney Mead, Oxford OX2 0EL.

RESENDE, M. G. C. e RIBEIRO, C. C. (2001) Greedy Randomized Adaptive Search Procedures. AT&T Labs Research Technical Report, Setembro, 2001.

RESENDE, M. G. C. (1998) Greedy Randomized Adaptive Search Procedures (GRASP). AT&T Labs Research Technical Report, Dezembro, 1998.

ROUSSEAU, J.; LESSARD,R. e BLAIS, J. Y. (1985) Enhancement to the Hastus crew scheduling algorithm. In: Computer Scheduling of Public Transport 2, Elsevier Amsterdam.

SANTOS, E. e ORRICO FILHO, R. (2000) Avaliando Metodologias da Avaliação de Desempenho de Ônibus Urbano: Os Modelos da EMTU – Recife e da BHTRANS. *In:* Santos, E. e Aragão, J. (*eds.*) *Transporte em Tempo de Reforma. Ensaios sobre a Problemática*. Ed. LGE, Brasília.

SHEN, Y. e KWAN, R. S. K. (2000) Tabu Search for Driver Scheduling. *VIII International Conference on Computer-Aided Scheduling of Public Transport*. Berlim, 2000.

SILVA, G. P.; SOUZA, M. J. F. e BENTO ALVES, J. M. C. (2002) Resolução do problema de Programação Diária da Tripulação de Ônibus Urbano Via *Simulated Annealing*, *Anais do XVI Congresso de Pesquisa e Ensino em Transportes*, vol. II, Natal, pp. 95-104, ANPET.

SIQUEIRA, P. H. (1999) Aplicação do Algoritmo do Matching no Problema da Construção de Escalas de Motoristas e Cobradores de Ônibus. Dissertação de Mestrado. Universidade Federal do Paraná.

SMITH, B, M. (1986) Bus Crew Scheduling using Mathematical Programming. PhD Thesis. University of Leeds.

STÜTZLE, T. (1999) Local Search Algorithms for Combinatorial Problems – Analysis, Algorithms and New Application. DISKI – Dissertationen zur Künstliken Intelligenz.

SYSWERDA, G. (1989) Uniform Crossover in genetic Algorithms. *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 2-9. Morgan Kaufmann.

TOSCANI, L. V. (2000) Teoria da Complexidade: Terminologia adotada pelo CTG. Departamento de Informática Teórica – UFRGS , Porto Alegre, outubro, 2000 www.inf.ufrgs.br/~ctg/pags/definicoes/definicoes.html.

U.S. DEPARTMENT OF TRANSPORTATION (1985) RUCUS: Automated Vehicle and Scheduling System. Urban Mass Transportation Administration. TR News, Maio-Junho 1985.

VIANA, G. V. R. (1998) Meta-Heuristicas e Programação Paralela em Otimização Combinatória. Edições UFC, Fortaleza.

WARD, R. E. e DEIBEL, L. E. (1972) The advancement of computerized assignment of transit operators to vehicles through programming techniques. Joint national Meeting of the Operations Research Society of America, Atlantic City.

WREN, A. (1996a) Bus and train operation – scheduling principles and practice. School of Computer Studies, Universidade de Leeds.

WREN, A. e GUALDA, N. D. F. (1997) Integrated scheduling of buses and drivers. *Technical Report* 97.32, School of Computer Studies, Universidade de Leeds. Apresentado no VII International Workshop on Computer-Aided Scheduling of Public Transport (MIT, Cambridge MA).

WREN, A. e KWAN, R. S. K. (1998) Installing an Urban Transport Scheduling System. http://citeseer.nj.nec.com/wren98installing.html.

WREN, A. e ROSSEAU, J. (1993) Bus Driver Scheduling – on Overview. *VI International Conference on Computer-Aided Scheduling of Public Transport*. Lisboa, 1993.

WREN, A.; KWAN, R. S. K. e KWAN, A. S. K. (2000) Hybrid Genetic Algorithms for Scheduling Bus and Train Drivers. *VIII International Conference on Computer-Aided Scheduling of Public Transport*. Berlim.

ANEXOS

ANEXO 1

ORDEM DE SERVIÇO OPERACIONAL (OSO)

OSO - Ordem de Serviço Operacional

Empresa de Trânsito e Transporte Urbano S/A

Empresa:

Linha: Extensão: Meias:

Programação:

Tabela: **Classe:** **Viagens Programadas:**

Posto de Controle:

HH:MM	HH:MM	HH:MM	HH:MM	HH:MM

Posto de Controle:

HH:MM	HH:MM	HH:MM	HH:MM	HH:MM

Eng. Sebastião Ramos da Silva

EMPRESA

Figura I.1: Ordem de Serviço Operacional - OSO

ANEXO 2

RESULTADOS DA LINHA 052 (TODAS AS EMPRESAS)

Tabela II.1: Solução inicial

SOLUÇÃO INICIAL									
Linha	TT em HN	TT em TU	TT em HE	TT em Oci	TT em AN	CT em HN	CT em HE	Custo Total	Jornadas
52	597:56:00	634:31:00	36:35:00	135:24:00	33:40:00	5390,587	418,438	**5809,025**	100

Tabela II.2: Solução final

SOLUÇÃO FINAL									
Linha	TT em HN	TT em TU	TT em HE	TT em Oci	TT em AN	CT em HN	CT em HE	Custo Total	Jornadas
52	604:13:00	643:33:00	39:20:00	85:07:00	33:40:00	5035,717	449,446	**5485,163**	94

Tabela II.3: Resultados do processo de otimização

SOLUÇÃO FINAL					
Linha	TT de Proc.	Temp. Inicial	Temp. Final	Iteração Melhor	Nº Ite. Sem/M
52	308	87982,881	578,090	63	437

onde:

a) TT em HN: Tempo Total em Horário Normal.
b) TT em TU: Tempo Total em Trabalho Útil.
c) TT em HE: Tempo Total em Horas-Extras.
d) TT em Oci.: Tempo Total em Ociosidade.
e) TT em AN: Tempo Total em Adicional Noturno.
f) CT em HN: Custo Total em Horário Normal.
g) CT em HE: Custo Total em Horas-Extras.
h) TT de Proc.: Tempo Total de Processamento.
i) Temp. Inicial: Temperatura Inicial.
j) Temp. Final: Temperatura Final.
k) Iteração Melhor: Iteração da melhor solução encontrada.

l) Nº Ite. Sem/M: Número de iterações sem ocorrer modificação na melhor solução.

Os gráficos mostrados abaixo ilustram, respectivamente, o comportamento da temperatura, da taxa de aceitação de novas soluções e do valor do custo da solução no decorrer do processo de otimização.

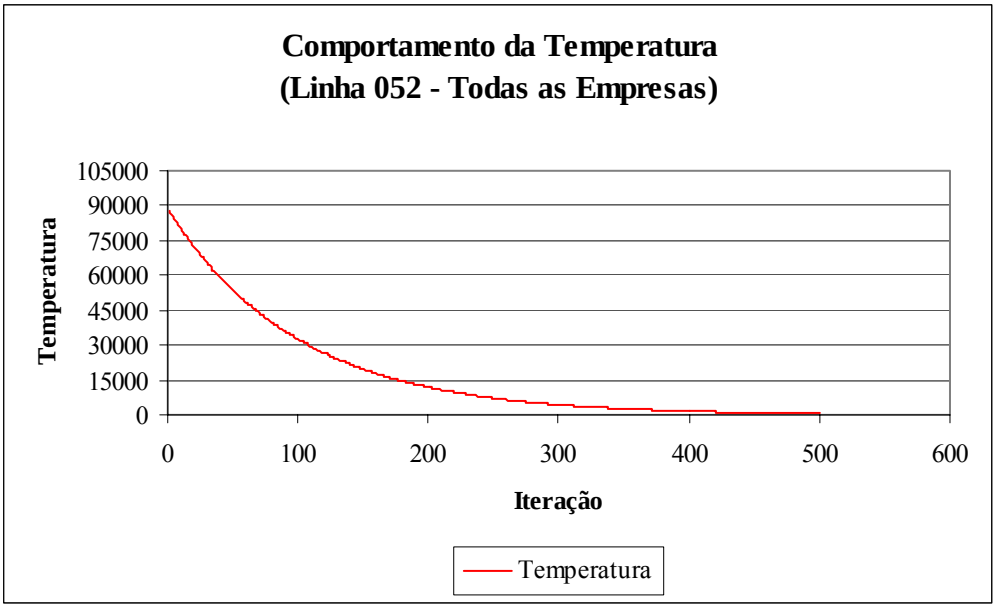

Figura II.1: Comportamento da temperatura

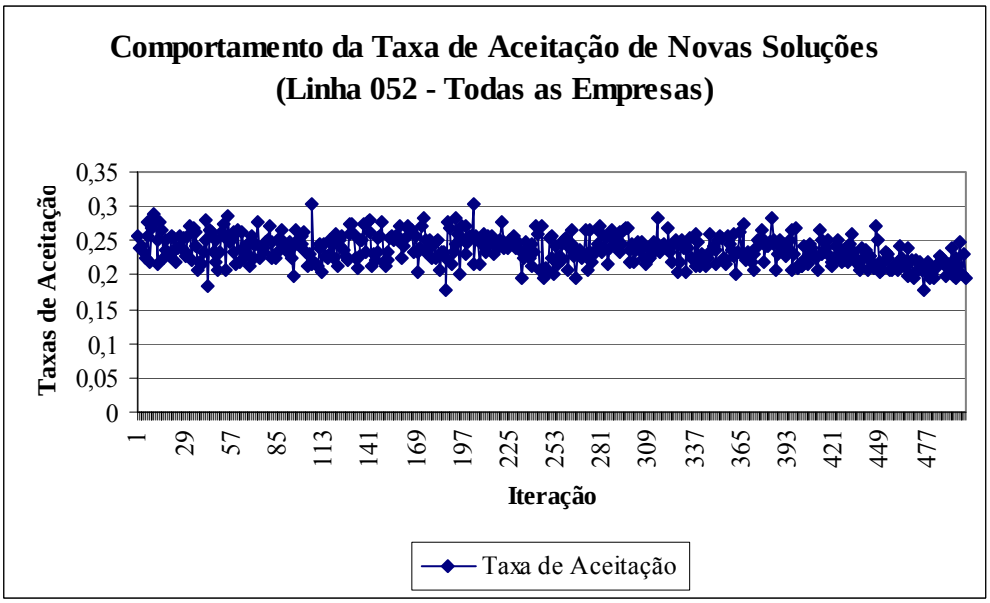

Figura II.2: Comportamento da taxa de aceitação

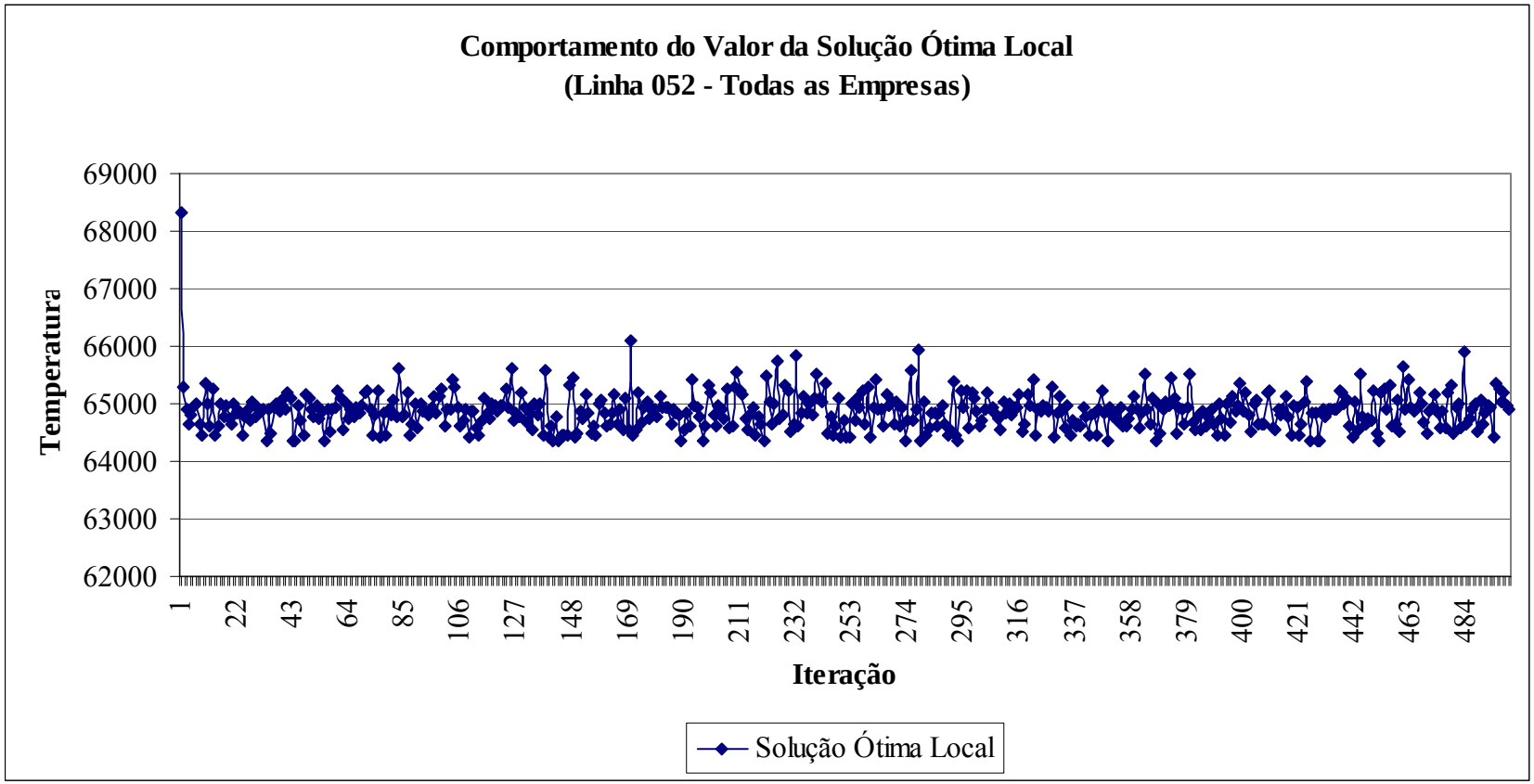

Figura II.3: Comportamento do valor do custo da solução ótima local

Pôde-se verificar, a partir da Figura II.2, que o valor da taxa de aceitação de novas soluções não se aproximou do valor zero, como era esperado. Isso se deve ao fato do processo ainda apresentar uma temperatura final alta (ver Tabela II.3). Além disso, verifica-se também que a solução corrente é bastante modificada, tanto no sentido de melhora quanto de piora da solução (ver Figura II.3), até nas últimas iterações.

Com base nesses resultados, apresentados na Figura II.2 e II.3, aplicou-se o programa ATTON nas mesmas condições e para a mesma linha, considerando agora 1000 iterações externas a fim de que fosse verificado o comportamento da taxa de aceitação de novas soluções.

Sendo assim, as Figuras II.4 e II.5 ilustram, para cada iteração, o comportamento da taxa de aceitação de novas soluções e o comportamento do valor do custo da melhor solução local, respectivamente.

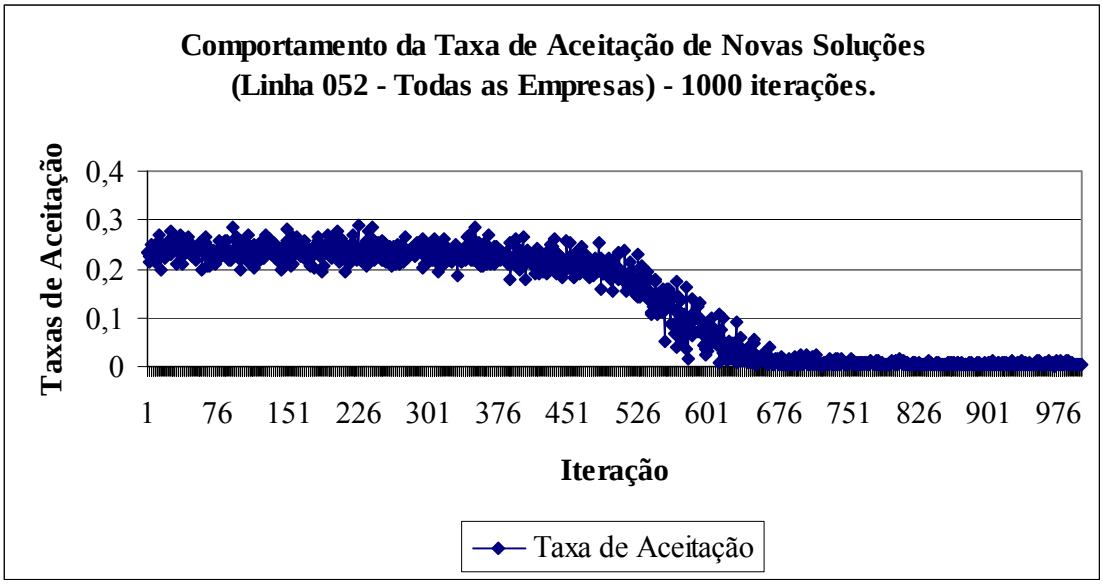

Figura II.4: Comportamento da taxa de aceitação de novas soluções (1000 iterações).

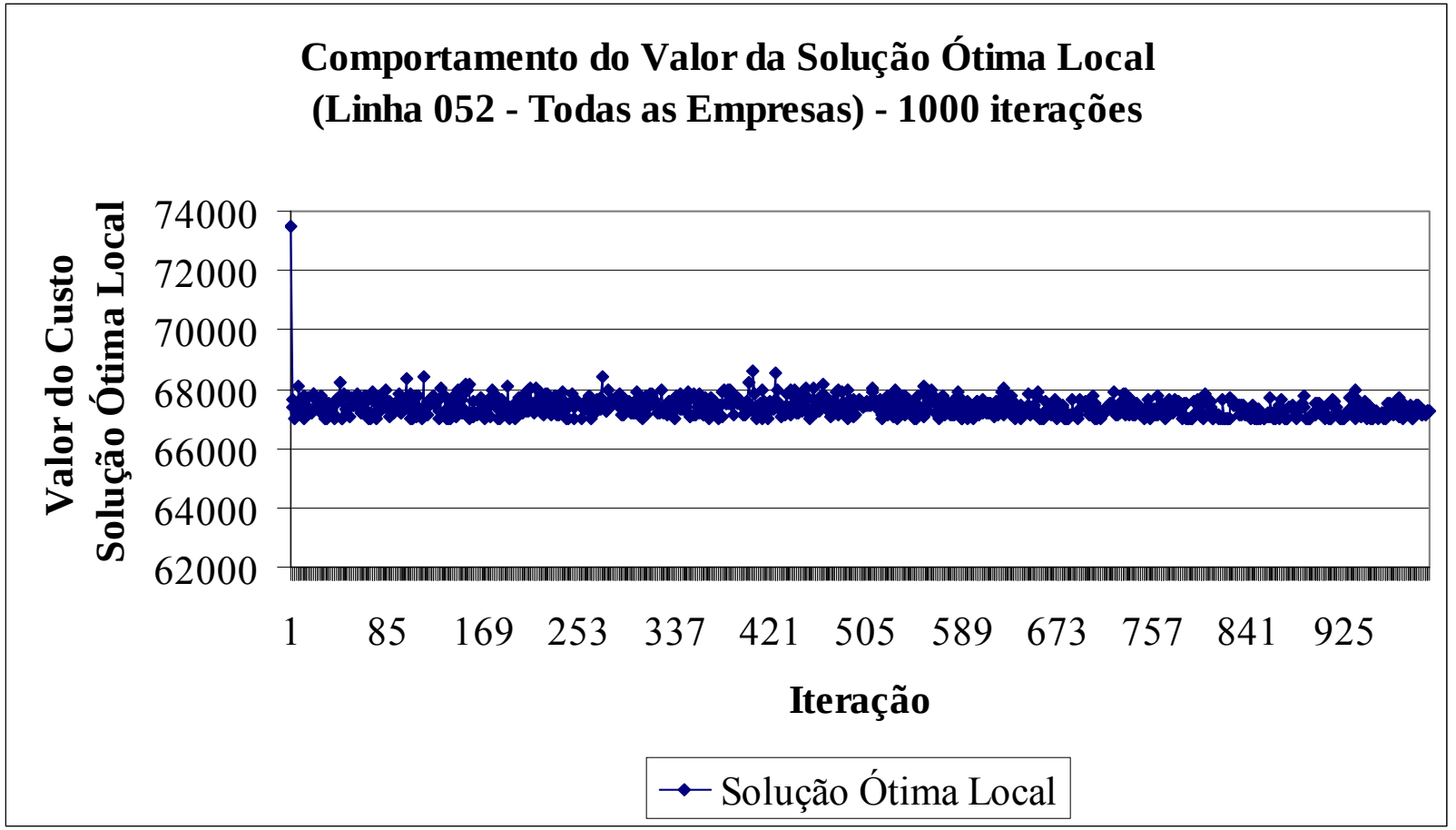

Figura II.5: Comportamento do valor do custo da solução ótima local (1000 iterações).

ANEXO 3

RESULTADOS DE TODAS AS LINHAS

Os resultados das soluções finais foram obtidos considerando as restrições específicas para cada linha estudada. Essas restrições se referem aos comprimentos máximo e mínimo do trecho de trabalho e comprimentos máximo e mínimo do intervalo.

a) Fator de decrescimento da temperatura igual a 0,90

Tabela III.1: Solução inicial

SOLUÇÃO INICIAL									
Linha	**TT em HN**	**TT em TU**	**TT em HE**	**TT em Oci**	**TT em AN**	**CT em HN**	**CT em HE**	**Custo Total**	**Nº de Jornadas**
52	77:51:00	79:55:00	2:04:00	17:29:00	5:26:00	693,642	22,825	**716,467**	13
600	48:57:00	51:02:00	2:05:00	17:03:00	1:17:00	486,264	22,935	**509,198**	9
613	41:18:00	42:53:00	1:35:00	17:22:00	2:46:00	434,621	17,430	**452,051**	8
631	19:06:00	19:45:00	0:36:00	2:51:00	3:17:00	166,279	6,605	**172,885**	3

Tabela III.2: Solução final

SOLUÇÃO FINAL									
Linha	**TT em HN**	**TT em TU**	**TT em HE**	**TT em Oci**	**TT em AN**	**CT em HN**	**CT em HE**	**Custo Total**	**Nº de Jornadas**
52	79:04:00	79:55:00	0:51:00	8:56:00	5:26:00	639,822	9,431	**649,253**	12
600	51:04:00	51:33:00	0:29:00	7:36:00	1:17:00	432,444	5,321	**437,765**	8
613	42:19:00	42:53:00	0:34:00	9:01:00	2:46:00	379,969	7,486	**387,455**	7
631	19:06:00	19:45:00	0:36:00	2:51:00	3:17:00	166,279	6,605	**172,885**	3

Tabela III.3: Resultado do processo de otimização

SOLUÇÃO FINAL					
Linha	**TT de Proc.**	**Temp. Inicial**	**Temp. Final**	**Iteração Melhor**	**Nº Ite. Sem/M**
52	46	8607,628	1,13E-19	1	499
600	43	8383,7371	1,11E-19	1	499
613	12	8730,851	1,15E-19	1	499
631	42	2200,431	2,90E-20	0	500

b) Fator de decrescimento da temperatura igual a 0,95

Tabela III.4: Solução inicial

SOLUÇÃO INICIAL									
Linha	**TT em HN**	**TT em TU**	**TT em HE**	**TT em Oci**	**TT em AN**	**CT em HN**	**CT em HE**	**Custo Total**	**Nº de Jornadas**
52	79:04:00	79:55:00	0:51:00	16:16	05:26	693,642	9,431	**703,073**	13
600	49:17:00	51:02:00	1:45:00	24:03:00	1:17:00	540,084	19,265	**559,349**	10
613	40:38:00	42:53:00	2:15:00	18:02:00	2:46:00	434,621	24,769	**459,390**	8
631	18:45:00	19:45:00	1:00:00	3:15:00	3:17:00	166,279	11,009	**177,288**	3

Tabela III.5: Solução final

SOLUÇÃO FINAL									
Linha	**TT em HN**	**TT em TU**	**TT em HE**	**TT em Oci**	**TT em AN**	**CT em HN**	**CT em HE**	**Custo Total**	**Nº de Jornadas**
52	79:04:00	79:55:00	0:51:00	8:56:00	5:26:00	639,822	9,431	**649,253**	12
600	51:04:00	51:33:00	0:29:00	7:36:00	1:17:00	432,444	5,321	**437,765**	8
613	42:19:00	42:53:00	0:34:00	9:01:00	2:46:00	379,969	7,486	**387,455**	7
631	19:06:00	19:45:00	0:36:00	2:51:00	3:17:00	166,279	6,605	**172,885**	3

Tabela III.6: Resultado do processo de otimização

SOLUÇÃO FINAL					
Linha	**TT de Proc.**	**Temp. Inicial**	**Temp. Final**	**Iteração Melhor**	**Nº Ite. Sem/M**
52	46	8002,764	5,82E-08	1	499
600	35	11321,819	8,24E-08	1	499
613	33	9342,442	6,79E-08	1	499
631	42	2508,6725	1,82E-20	1	499

c) Fator de decrescimento da temperatura igual a 0,99

Tabela III.7: Solução inicial

SOLUÇÃO INICIAL									
Linha	**TT em HN**	**TT em TU**	**TT em HE**	**TT em Oci**	**TT em AN**	**CT em HN**	**CT em HE**	**Custo Total**	**Nº de Jornadas**
52	78:59:00	79:55:00	0:55:00	16:21:00	5:26:00	693,642	10,348	**703,990**	13
600	49:17:00	51:02:00	1:45:00	16:43:00	1:17:00	486,264	19,265	**505,529**	9
613	42:53:00	42:53:00	0:00:00	15:47:00	2:46:00	434,621	0,000	**434,621**	8
631	18:45:00	19:45:00	1:00:00	3:15:00	3:17:00	166,279	11,009	**177,288**	3

Tabela III.8: Solução final

SOLUÇÃO FINAL									
Linha	**TT em HN**	**TT em TU**	**TT em HE**	**TT em Oci**	**TT em AN**	**CT em HN**	**CT em HE**	**Custo Total**	**Nº de Jornadas**
52	79:04:00	79:55:00	0:51:00	8:56:00	5:26:00	639,822	9,431	**649,253**	12
600	51:04:00	51:33:00	0:29:00	7:36:00	1:17:00	432,444	5,321	**437,765**	8
613	42:19:00	42:53:00	0:34:00	9:01:00	2:46:00	379,969	7,486	**387,455**	7
631	19:06:00	19:45:00	0:36:00	2:51:00	3:17:00	166,279	6,605	**172,885**	3

Tabela III.9: Resultado do processo de otimização

SOLUÇÃO FINAL					
Linha	**TT de Proc.**	**Temp. Inicial**	**Temp. Final**	**Iteração Melhor**	**Nº Ite. Sem/M**
52	46	7567,923	49,724	1	499
600	44	8077,9416	53,0759	1	499
613	33	7508,281	49,333	1	499
631	44	2508,6725	3,32E-20	0	500

ANEXO 4

RESULTADO DA PROGRAMAÇÃO DE CADA LINHA DE ÔNIBUS

A seguir são mostrados os dados referentes à programação adotada pela empresa Rotaexpressa S/A e a obtida pelo programa para cada linha.

a) Linha 052 – Grande Circular 02

Tabela IV.1: Programação adotada pela empresa

Resultado da Empresa (Linha 052)						
Jornada	Início	Término	Comprimento da Jornada (hh:mm)	Hora-Extra (hh:mm)	Período Noturno (hh:mm)	Custo (R$)
1	13:30	21:14	07:18	00:00	00:00	53,80
2	05:42	13:08	06:58	00:00	00:00	53,80
3	04:10	13:05	07:59	00:39	00:50	62,17
4	05:40	13:30	07:18	00:00	00:00	53,80
5	13:05	20:22	06:48	00:00	00:00	53,80
6	05:00	14:21	08:00	00:40	00:00	61,17
7	16:50	23:59	06:33	00:00	01:59	56,73
8	15:18	23:11	07:22	00:02	01:11	55,95
9	14:21	23:26	07:49	00:29	01:26	61,60
10	13:08	20:54	07:25	00:05	00:00	54,68
11	05:38	13:24	07:20	00:00	00:00	53,80
					Total	621,28

Tabela IV.2: Programação obtida pelo programa

Resultado do Programa (Linha 052)						
Jornada	Início	Término	Comprimento da Jornada (hh:mm)	Hora-Extra (hh:mm)	Período Noturno (hh:mm)	Custo (R$)
1	04:10	10:59	06:20	00:00	00:50	55,04
2	11:55	20:22	07:58	00:38	00:00	60,79
3	05:40	13:30	07:18	00:00	00:00	53,82
4	13:30	21:14	07:18	00:00	00:00	53,82
5	05:38	13:24	07:20	00:00	00:00	53,82
6	15:18	23:11	07:22	00:02	01:11	42,00
7	05:00	10:52	05:26	00:00	00:00	53,82
8	12:13	20:30	07:31	00:11	00:00	55,84
9	21:00	23:26	02:26	00:00	01:26	55,92
10	05:42	13:30	07:20	00:00	00:00	53,82
11	13:30	20:54	07:03	00:00	00:00	53,82
12	16:50	23:59	06:33	00:00	01:59	56,73
					Total	649,25

b) Linha 600 – Messejana/Frei Cirilo/Expresso

Tabela IV.3: Programação adotada pela empresa

Resultado da Empresa (Linha 600)						
Jornada	Início	Término	Comprimento da Jornada (hh:mm)	Hora-Extra (hh:mm)	Período Noturno (hh:mm)	Custo (R$)
1	06:33	16:11	07:39	00:19	00:00	57,32
2	12:22	20:36	07:49	00:29	00:00	59,08
3	16:11	21:10	04:59	00:00	00:00	53,80
4	05:53	10:31	04:38	00:00	00:00	53,80
5	04:53	12:04	07:11	00:00	00:07	53,99
6	05:19	13:24	06:36	00:00	00:00	53,80
7	13:24	20:28	06:34	00:00	00:00	53,80
8	16:27	23:10	06:28	00:00	01:10	55,53
					Total	441,10

Tabela IV.4: Programação obtida pelo programa

Resultado do Programa (Linha 600)						
Jornada	Início	Término	Comprimento da Jornada (hh:mm)	Hora-Extra (hh:mm)	Período Noturno (hh:mm)	Custo (R$)
1	04:53	12:04	06:50	00:00	00:07	53,99
2	16:27	23:10	06:28	00:00	01:10	55,53
3	05:19	13:24	06:36	00:00	00:00	53,82
4	13:24	20:28	06:34	00:00	00:00	53,82
5	05:53	10:31	04:38	00:00	00:00	53,82
6	12:22	20:36	07:49	00:29	00:00	59,14
7	06:33	15:41	07:09	00:00	00:00	53,82
8	15:41	21:10	04:58	00:00	00:00	53,82
					Total	437,77

c) Linha 613 – Barroso/Jardim Violeta

Tabela IV.5: Programação adotada pela empresa

Resultado da Empresa (Linha 613)						
Jornada	Início	Término	Comprimento da Jornada (hh:mm)	Hora-Extra (hh:mm)	Período Noturno (hh:mm)	Custo (R$)
1	05:19	12:34	06:58	00:00	00:00	53,80
2	14:28	22:46	08:05	00:45	00:46	63,56
3	04:44	10:14	05:30	00:00	00:16	54,21
4	15:50	00:15	07:54	00:34	02:15	63,79
5	12:34	19:57	06:59	00:00	00:00	53,80
6	06:19	14:28	07:44	00:24	00:00	58,20
					Total	347,35

Tabela IV.6: Programação obtida pelo programa

Resultado do Programa (Linha 613)						
Jornada	Início	Término	Comprimento da Jornada (hh:mm)	Hora-Extra (hh:mm)	Período Noturno (hh:mm)	Custo (R$)
1	04:44	10:14	05:30	00:00	00:16	54,21
2	15:50	00:15	07:54	00:34	01:44	63,02
3	05:19	12:34	06:58	00:00	00:00	53,82
4	12:34	19:57	06:59	00:00	00:00	53,82
5	06:19	13:43	06:59	00:00	00:00	53,82
6	13:43	20:53	06:53	00:00	00:00	53,82
7	21:06	22:46	01:40	00:00	00:46	54,95
					Total	387,46

d) Linha 631 – Carlos Albuquerque

Tabela IV.7: Programa adotada pela empresa

Resultado da Empresa (Linha 631)						
Jornada	Início	Término	Comprimento da Jornada (hh:mm)	Hora-Extra (hh:mm)	Período Noturno (hh:mm)	Custo (R$)
1	15:28	00:37	08:53	01:33	02:37	75,84
2	12:08	15:28	03:20	00:00	00:00	53,80
3	04:20	12:08	07:32	00:12	00:40	56,98
					Total	186,62

Tabela IV.8: Programação obtida pelo programa

Resultado do Programa (Linha 631)						
Jornada	Início	Término	Comprimento da Jornada (hh:mm)	Hora-Extra (hh:mm)	Período Noturno (hh:mm)	Custo (R$)
1	04:20	12:32	07:56	00:36	00:40	61,40
2	12:32	20:08	07:20	00:00	00:00	53,82
3	20:08	00:37	04:29	00:00	02:37	57,66
					Total	172,89