LEONARDO ROSA ROHDE

DESENVOLVIMENTO DE HEURÍSTICA PARA SOLUÇÃO DO PROBLEMA DE ESCALONAMENTO DE VEÍCULOS COM MÚLTIPLAS GARAGENS

Tese apresentada como requisito para obtenção do título de Doutor pelo Programa de Pós-Graduação da Faculdade de Administração da Universidade Federal do Rio Grande do Sul.

Orientador: Denis Borenstein

Porto Alegre

2008

LEONARDO ROSA ROHDE

DESENVOLVIMENTO DE HEURÍSTICA PARA SOLUÇÃO DO PROBLEMA DE ESCALONAMENTO DE VEÍCULOS COM MÚLTIPLAS GARAGENS

Tese apresentada como requisito para obtenção do título de Doutor pelo Programa de Pós-Graduação da Faculdade de Administração da Universidade Federal do Rio Grande do Sul.

Aprovada em _____ de _____ de _____ .

BANCA EXAMINADORA:

Prof. Eduardo Ribas Santos

_____

Prof. João Carlos Furtado

_____

Prof. Roberto Protil

_____

Dedico esse trabalho aos meus pais:
Vilmar Mario Rohde e Circe Rosa Rohde.
Meus primeiros educadores
a quem devo tudo que sou.

AGRADECIMENTOS

O processo de doutoramento não é fácil e muitas são as pessoas que contribuem, cada uma a sua maneira, durante todo período em que o trabalho é desenvolvido. Citar aqui cada pessoa que compartilhou suas idéias, ofereceu ajuda, criticou, auxiliou, participou ou incentivou esse processo é uma forma de demonstrar a minha eterna gratidão e todo meu carinho. A essas pessoas fica o meu eterno agradecimento:

- Primeiramente, pela minha religiosidade e crença, gostaria de manifestar meu agradecimento a Deus, pois sem ele nada seria, é e será possível;

- Aos meus pais, Vilmar Mario Rohde e Circe Rosa Rohde, pela educação, apoio e sacrifícios. Eles são os responsáveis pelo que sou e ninguém tem mais mérito do que eles no que diz respeito a minha educação e formação. Muito obrigado, amo muito vocês!

- Aos meus irmãos, Leandro Rosa Rohde, Adriana Rosa Rohde, Daniel Rosa Rohde e Rodrigo Santos Rosa, pela presença constante, apoio e carinho nesses últimos quatro anos. Compartilhar minha vida com vocês é o maior presente que me foi dado!

- Ao meu orientador e amigo Denis Borenstein, que a cada exigência me transformou em alguém melhor – e não existe maior ato de bondade que o de melhorar alguém. Conheço o professor Denis há anos, quando ainda graduava-me em administração e é notória a aprendizagem que me foi proporcionada durante esse convívio. Muito obrigado pela orientação, exigências e amizade. Podes contar sempre comigo;

- Aos professores do Programa de Pós-Graduação em Administração da Universidade Federal do Rio Grande do Sul cujos conhecimentos são inigualáveis. Obrigado por terem contribuído com a minha formação;

- Muitos foram aqueles que me apoiaram nas horas difíceis e me conduziram até aqui durante esses quatro anos. O tempo afastou algumas dessas pessoas, outras ele aproximou. Mas que todas elas saibam que eu reconheço cada sacrifício, paciência e contribuição. Sei quantas pessoas perderam com o meu crescimento. Não vou citá-las... mas a elas fica meu sincero agradecimento acompanhado de um justo sentimento de dívida.

“The reasonable man adapts himself to the world;
The unreasonable one persists in trying to adapt the world to himself.
Therefore all progress depends
on the unreasonable man”.

George Bernard Shaw

RESUMO

Existem vários problemas clássicos na área de pesquisa operacional que trabalham com o tema vinculado à designação de veículos em um sistema logístico, entre eles o Problema de Escalonamento de Veículos com Múltiplas Garagens (MDVSP). Esses modelos são largamente utilizados e representam uma das etapas essenciais para o planejamento de trânsito em massa (HAGHANI e BANIHASHEMI, 2002). Tratando-se de sistemas logísticos reais, dificilmente encontra-se um ambiente onde os veículos devem partir e chegar a uma única garagem, por isso torna-se necessário o planejamento das seqüências de viagens de modo a reduzir os custos de deslocamentos com o aproveitamento das múltiplas garagens distribuídas geograficamente. Infelizmente, considerando a complexidade exponencial do MDVSP, muitas vezes sua aplicação torna-se inviável na solução de problemas reais. Por essa razão, poucos trabalhos abordam o MDVSP de modo a conseguir solucionar o problema para uma grande quantidade de viagens e garagens. A maioria das pesquisas trabalha com instâncias inferiores a 500 viagens e quatro garagens, mostrando-se pouco aplicáveis. Esse estudo refere-se a um trabalho de pesquisa operacional que aborda soluções de problemas de escalonamento de veículos com múltiplas garagens (MDVSP) considerando sua aplicabilidade em sistemas reais. Tendo em vista a complexidade exponencial do MDVSP, nesse estudo optou-se por tratar o problema através de uma abordagem baseada na redução do espaço de estados e na utilização de heurísticas. Durante essa pesquisa três procedimentos de redução do espaço de estados foram adotados. Os resultados apontam que é possível reduzir em até 98% o número de variáveis nesses problemas sem comprometer uma solução satisfatória ou ótima. Além dos procedimentos de redução do espaço de estados, foi desenvolvido um procedimento de buscar a solução do MDVSP. Através desse último procedimento foi possível resolver o MDVSP com até 3000 viagens e oito garagens. Sendo assim, nesse estudo desenvolveram-se modelos que servem para o planejamento de um sistema logístico através da aplicação de cenários, com vistas a permitir a geração e análise de alternativas de escalonamento. Objetivou-se com isso, fornecer ao sistema logístico um modelo amplo que permita a escolha da ação mais conveniente e eficiente a ser tomada em modelos compostos por diversas garagens.

Palavras-Chave: Escalonamento de Veículos com Múltiplas Garagens. Heurística. Espaço de Estados.

ABSTRACT

There are many classics problems in operations research concerning optimal assignment vehicles in logistical system. The multiple depot vehicle scheduling problem (MDVSP) is one of them. This problem is largely used to represent and solve mass transit planning (HAGHANI e BANIHASHEMI, 2002). Considering a real logistical system, it is very difficult to find out a situation where the vehicles must leave and come to only one depot. In general, the shipping company has several depots located at different sites in a network. In this way, it is strongly necessary to reduce cost through the planning of sequence trips taking into account multiple depots geographically distributed. Unfortunately, the exponential complexity of the MDVSP reduces, in the most cases, the applicability of this problem in the real world. For this reason, few researchers address the MDVSP to solve real world problems considering a large number of trips and depots. The majority of the research dealing with the MDVSP works with instances lower than 500 trips and four depots, what can be considered a major constraint for its practical use. The main objective of this work is to solve the MDVSP for very large instances. A state space reduction approach combined with heuristic procedures are developed to obtain a realistic way of solving this complex problem. In this research, three state space reduction procedures were developed. The results appointed that is possible to reduce until 98% of variables in the MDVSP without jeopardizing an optimal solution. Furthermore, heuristic procedures were developed to obtain solutions without relaxing any real-world constraint of the problem. The solution procedure developed was compared with well-known available instances. The method is able to solve the MDVSP with 3000 trips and eight depots in less than 11 minutes. Although the solution process does not obtain the best solution in all tested instances, it is by far the quickest.

Keywords: Multiple Depot Vehicle Scheduling Problem. Heuristic. State Space Reduction.

LISTA DE FIGURAS

Figura 1 – Solução inicialmente proposta – VSP 46

Figura 2 – Possível solução VRSP 47

Figura 3 – Transformação do VRSP em MDVSP 50

Figura 4 – Fluxograma da heurística 57

Figura 5 – Exemplo aleatório de MDVSP com quinze viagens e duas garagens 58

Figura 6 – Interface do módulo de redução 60

Figura 7 – Interface da matriz de folga total 61

Figura 8 – Histogramas por número de classes 63

Figura 9 – Algoritmo do primeiro procedimento de redução do espaço de estados 64

Figura 10 – Interface do primeiro procedimento de redução do espaço de estados 65

Figura 11 – Interface do modelo linear VSP 70

Figura 12 – Interface do interpretador de respostas VSP 71

Figura 13 – Algoritmo de contabilização de respostas 72

Figura 14 – Interface do contador de respostas 73

Figura 15 – Interface da redução do espaço de estados completa 74

Figura 16 – Intersecção dos espaços de estados MDVSPcf, MDVSP e VSP 78

Figura 17 – Interface do módulo de solução MDVSP 80

Figura 18 – Fluxograma da heurística 91

LISTA DE QUADROS

Quadro 1 – Etapas da Pesquisa 23

Quadro 2 – Exemplo de quadro de horários 37

Quadro 3 – Exemplo de quadro de horários 45

Quadro 4 – Descrição da heurística 56

Quadro 5 – Viagens iniciais e finais selecionadas 69

Quadro 6 – Resposta da primeira iteração do MDVSPcf 81

Quadro 7 – Solução do MDVSP a cada iteração. 85

LISTA DE TABELAS

Tabela 1 – Viagens com menor custo por linha e coluna 64

Tabela 2 – Resultados do primeiro procedimento de redução do espaço de estados 66

Tabela 3 – Comparação dos resultados após redução do espaço estados 75

Tabela 4 – Custo de Interrupção do Algoritmo 84

Tabela 5 – Comparação da heurística com o MDVSP 87

Tabela 6 – Aplicação da heurística para instâncias superiores a da literatura 87

Tabela 7 – Comparação da heurística com os resultados de Huisman 88

SUMÁRIO

1 INTRODUÇÃO 13

2 SITUAÇÃO PROBLEMÁTICA E JUSTIFICATIVA DE PESQUISA 15

3 OBJETIVOS 21
3.1 OBJETIVO GERAL 21
3.2 OBJETIVOS ESPECÍFICOS 21

4 MÉTODO DE PESQUISA 22
4.1 ETAPAS DA PESQUISA 23
4.2 GERADOR DE VIAGENS ALEATÓRIAS 27
4.3 IMPLEMENTAÇÃO E VALIDAÇÃO COMPUTACIONAL DOS MODELOS 29

5 REFERENCIAL TEÓRICO 31
5.1 CONCEITOS RELACIONADOS A ESCALONAMENTO DE VEÍCULOS 33
5.2 O VEHICLE SCHEDULING PROBLEM (VSP) 34
5.2.1 Formulação Matemática do VSP 38
5.3 O MULTIPLE DEPOT VEHICLE SCHEDULING PROBLEM (MDVSP) 39
5.3.1 Formulação Matemática do MDVSP 40
5.4 O VEHICLE RESCHEDULING PROBLEM (VRSP) 42
5.4.1 Formulação matemática do VRSP 45
5.5 O VRSP COMO INSTÂNCIA DO MDVSP 49
5.6 TRABALHOS DESENVOLVIDOS EM VSP, MDVSP E VRSP 51

6 PROPOSTA DE HEURÍSTICA PARA SOLUÇÃO DO MDVSP 56
6.1 PRIMEIRA ETAPA – ALGORITMO DE REDUÇÃO DO ESPAÇO DE ESTADOS 58
6.1.1 Redução pelo critério estatístico 60
6.1.2 Definição das viagens iniciais e finais 67
6.1.3 Exclusão de variáveis nunca selecionadas 71
6.2 COMPARAÇÃO DOS RESULTADOS 75
6.3 SEGUNDA ETAPA - ALGORITMO DE SOLUÇÃO MDVSP 76
6.4 APLICAÇÃO DA HEURÍSTICA 85

7 CONSIDERAÇÕES FINAIS 90

REFERÊNCIAS 95

APÊNDICE A – Módulo gerador de viagens aleatórias (GVA) 99

APÊNDICE B – Módulo de redução 103

APÊNDICE C – Procedimento de redução 1 105

APÊNDICE D – Procedimento de redução 2 107

APÊNDICE E – Procedimento de redução 3 109

APÊNDICE F – Modelo linear VSP 111

APÊNDICE G – Modelo linear vsp para a garagem 1 113

APÊNDICE H – Modelo linear vsp para a garagem 2 114

APÊNDICE I – Modelo de redução – procedimentos 1, 2 e 3 115

APÊNDICE J – Modelo linear MDVSPcf 117

APÊNDICE L – Procedimento 4 para de solução do MDVSP 118

1 INTRODUÇÃO

A otimização faz parte da índole humana. Desde seu surgimento, o *homo sapiens* vem se dedicando a minimizar esforços e maximizar os retornos de alguma atividade por ele desenvolvida. Para atingir o melhor resultado em suas atividades, o decisor precisa compreender o sistema em que está inserido, bem como, as relações existentes entre os diversos elementos que compõem este sistema.

Um sistema é representado por um conjunto de elementos relacionados entre si para formar o todo. Estes elementos indicam algum fenômeno num estado do sistema, assim, os estados do sistema são percebidos nas características ou atributos assumidos por seus elementos (FLOOD e CARSON, 1993).

Em sistemas logísticos essas relações podem ser facilmente percebidas. Nesses sistemas a distribuição geográfica das garagens, a quantidade de veículos disponíveis, o tempo de início e término das viagens se inter-relacionam na organização e planejamento das seqüências ou rotas nas quais as viagens devem ocorrer. Tais problemas não podem ser solucionados considerando os aspectos citados de forma isolada, pois essa conduta comprometeria a otimidade do sistema ou mesmo a sua tratabilidade.

Na literatura existem vários problemas clássicos na área de pesquisa operacional que trabalham com o tema vinculado à designação de veículos em um sistema logístico, entre eles o MDVSP (do inglês, *Multiple Depot Vehicle Scheduling Problem* – MDVSP). O MDVSP é largamente utilizado e representa uma das etapas essenciais para o planejamento de trânsito em massa (HAGHANI e BANIHASHEMI, 2002). Infelizmente, considerando a alta complexidade envolvida, muitas vezes é inviável a aplicação computacional de tais modelos em problemas do mundo real.

Entende-se por alta complexidade computacional os problemas chamados não-

polinomiais. Problemas não-polinomiais são aqueles cujo tempo de execução ou a memória computacional requerida pelo algoritmo, em função de sua entrada, não pode ser calculado por equações polinomiais. Desta forma, o tempo de execução ou memória requerida por um algoritmo estende-se, impossibilitando uma resposta imediata do sistema ou até mesmo a tratabilidade do problema (CAMPELLO e MACULAN, 1994; TOSCANI e VELOSO, 2001). A alta complexidade destes problemas obriga-nos a encontrar soluções heurísticas ou meta-heurísticas para sua tratabilidade.

Nessa pesquisa considera-se que as rotas de uma rede de transporte já estão otimizadas ou, se não estão, ao menos são previamente conhecidas e, por isso, o estudo aborda apenas as questões de escalonamento de veículos considerando sua aplicação em um ambiente com múltiplas garagens. Resumidamente, tratando-se de sistemas logísticos reais, dificilmente encontra-se um ambiente onde os veículos devem partir e chegar a uma única garagem, por isso torna-se necessário o planejamento das seqüências de viagens de modo a reduzir os custos de deslocamentos com o aproveitamento das múltiplas garagens distribuídas geograficamente.

2 SITUAÇÃO PROBLEMÁTICA E JUSTIFICATIVA DE PESQUISA

Para entender a problemática que envolve o MDVSP é preciso detalhar uma série de peculiaridades dos sistemas logísticos que utilizam esse tipo de modelo. Inicialmente é preciso compreender que o uso do MDVSP pressupõe a existência de diversas garagens que interferem no planejamento das seqüências com as quais as viagens devem ser executadas. A principal justificativa de desenvolvimento de modelos com múltiplas garagens deve-se ao fato prático de que poucos são os sistemas reais que utilizam uma única garagem. Por exemplo, em sistemas aéreos cada aeroporto pode ser considerado como uma garagem, em um sistema rodoviário cada rodoviária pode ser considerada como uma garagem, em um sistema público de transporte cada ponto de reabastecimento de veículo pode ser considerado como uma garagem, e assim por diante. É possível perceber que são poucos os exemplos de sistemas logísticos que utilizam uma única garagem, de modo que seu planejamento deve fazer uso de modelos capazes de representar essa realidade. Somado a isso, o modelo MDVSP é de grande relevância se considerar os custos envolvidos em um sistema logístico, conforme menciona-se a seguir:

(a) Custo operacional: refere-se aos custos de operação dos veículos, tais como, horas do motorista, combustível, depreciação, entre outros. Quanto maior o deslocamento de veículos por linhas expressas, maior o custo da empresa;

(b) Custo fixo do veículo: a empresa pode possuir vários veículos de reserva para atender as necessidades contingenciais, porém isso implica em custo de investimento imobilizado. O ideal seria encontrar uma solução que equilibre o custo de investimento e o custo operacional dos veículos;

(c) Custo de espera: refere-se ao custo de espera dos passageiros pelo não atendimento no horário planejado. Em linhas gerais é o custo da insatisfação do usuário gerado pelo atraso;

(d) Custo de ociosidade: refere-se ao tempo que o veículo aguarda em um determinado ponto por ter se antecipado ao tempo de início planejado para uma viagem ou, simplesmente, pela não utilização do veículo;

(e) Custo de cancelamento: uma alternativa possível do sistema é cancelar uma viagem. Naturalmente, isso representa um custo na imagem da empresa além da perda de receita oriunda da linha cancelada;

Em vista dos custos envolvidos, o MDVSP exige uma solução considerando sua totalidade, uma vez que a solução individual do modelo considerando cada garagem isoladamente não conduz a otimidade do sistema. Em outros termos, é necessário resolver o problema considerando todas as garagens simultaneamente num modelo único. Essa peculiaridade torna o MDVSP um modelo NP difícil, ou seja, matematicamente ainda não foi provada a possibilidade de encontrar uma solução em tempo polinomial para o problema. Assim, do ponto de vista teórico é possível encontrar soluções que planejem todo o sistema, contudo no que tange os aspectos computacionais, tais soluções podem ser impraticáveis por apresentarem tempo de solução exponencial. Simon (1989) já chamava atenção para essa problemática quando escreveu:

> A modelagem é uma das principais – senão a mais importante – ferramenta para o estudo do comportamento de grandes sistemas complexos. Quarenta anos de experiência no desenvolvimento de modelos em computadores que, ano após ano, têm-se tornado maiores e mais rápidos, ensinaram-nos que somente a força bruta não é suficiente para alcançarmos o caminho nobre para a compreensão de tais sistemas. A natureza é capaz de construir em qualquer escala – do microcosmo ao macrocosmo – sistemas cuja complexidade situa-se muito além do alcance de nossos computadores ou supercomputadores, atuais ou em prospecção (SIMON, 1989).

Alguns modelos construídos são de tamanha magnitude e complexidade que mesmo nossos maiores computadores não conseguem manuseá-los. Isso se deve ao grande número de combinações e variáveis existentes nestes modelos. Deste modo, deve-se identificar e separar o essencial do dispensável na criação de sistemas, permitindo simplificar a realidade focando os objetivos mais importantes (SIMON, 1989). Com o aumento da complexidade dos modelos, os métodos e ferramentas de investigação e solução tornaram-se cada vez mais sofisticados (LANDRY, MALOUIN e ORAL, 1983).

Todas as razões apresentadas explicam a escassez de modelos que conseguem alcançar uma efetiva aplicabilidade em problemas do mundo real. Nota-se que diversas restrições e peculiaridades de sistemas logísticos são ignoradas deixando margem para futuras pesquisas na área. Deste modo, listou-se uma série de problemas ainda não trabalhados ou pouco desenvolvidos na literatura atual que abrem um leque de alternativas para pesquisa, conforme segue:

(a) Poucos modelos consideram o objetivo de multicritério, como, por exemplo, encontrar a melhor relação que minimize os custos de deslocamento, o número de veículos escalonados, a ociosidade dos veículos, o cancelamento de viagens, entre outros fatores;

(b) Poucos trabalhos abordam o MDVSP de modo a conseguir solucionar o problema para uma grande quantidade de viagens e garagens. A maioria das pesquisas trabalha com instâncias inferiores a 500 viagens e quatro garagens;

(c) Existem poucos trabalhos que mostram como adaptar ou aproveitar os modelos de MDVSP para solucionar de forma eficiente os problemas de VRSP;

(d) Percebe-se a quase inexistência de estudos que trabalhem de forma integrada os problemas de VRSP (do inglês, *Vehicle Rescheduling Problem*), MDVSP, VRRP (do inglês, *Vehicle Rerouting Problem*) e CSP (do inglês, *Crew Scheduling Problem*).

Tendo em vista todos os aspectos levantados acima e considerando a difícil tratabilidade do MDVSP, nesse estudo optou-se por tratar o problema descrito no item (b) através de uma abordagem baseada na redução do espaço de estados e na utilização de heurísticas. Por espaço de estados entende-se todas as possibilidades de resposta que um sistema possui, enquanto heurística é um conjunto de procedimentos para solução de um problema de modo a garantir a obtenção de uma resposta satisfatória.

O fato de se trabalhar com modelos de rede conduz a problemas de análise combinatorial e de grande dificuldade de solução (GOLDBARG e LUNA, 2000). Conseqüentemente, é comum encontrar na literatura a resolução de problemas complexos com o uso de uma estratégia conhecida como *Dividir e Conquistar*. Estas estratégias consistem na divisão de problemas muito complexos em problemas menores. O passo seguinte é reagrupar os problemas menores na tentativa de alcançar uma solução exata para o problema original. Este tipo de abordagem na maioria das vezes não conduz a soluções ótimas e, não raro, a soluções inviáveis. Em resposta a esta complexidade e pelos motivos expostos, o uso de heurísticas e reduções do espaço de estados tornou-se um fator chave no desenvolvimento de sistemas e, atualmente, têm sido cada vez mais pesquisadas e utilizadas, razão pela qual adotou-se essa linha de pesquisa para abordar e solucionar o problema MDVSP, foco desse estudo.

Segundo Bertalanffy (1975), de uma maneira ou de outra, somos forçados a tratar com a complexidade dos sistemas em todas as áreas do conhecimento. Por essa razão a modelagem torna-se essencial na tratabilidade de modelos complexos. De acordo com Rothenberg (1989), um modelo é a representação ou abstração da realidade para um determinado fim, uma vez que não pode representar todos os seus aspectos. Assim, os modelos permitem-nos tratar com o mundo de uma maneira simples, evitando a complexidade, os danos e a irreversibilidade da realidade (PIDD, 1998). Ackoff e Sasieni (1977) definiram modelos como representações da realidade e acrescentaram:

> Se os modelos fossem tão complexos e difíceis de controlar como a realidade, não haveria nenhuma vantagem em utilizá-los. Felizmente, podemos em geral construir modelos que são muito mais simples que a realidade e ainda assim conseguimos empregá-los para prever e explicar fenômenos com alto grau de precisão. A razão disso é que, embora seja necessário um grande número de variáveis para prever um fenômeno com exatidão perfeita, um pequeno número de variáveis explica geralmente a maior parte dele. O truque, evidentemente, é achar as variáveis certas e a relação correta entre elas (ACKOFF e SASIENI, 1977).

Diante do exposto a escolha de uma representação adequada do espaço de estados de um sistema torna-se ponto chave no desenvolvimento de um modelo ou de sua

implementação computacional. Por esta razão a escolha da representação deve estar baseada em suas vantagens e desvantagens, levando-se em consideração o objetivo do modelo que será desenvolvido. Entre as representações mais escolhidas encontram-se os grafos, sentenças booleanas, matrizes, equações lineares, entre outras. Muitas vezes, a geração de um espaço de estados torna-se bastante exaustiva, devido à sua representação e pelo crescimento exponencial característico de problemas não polinomiais (MORGAN e RAZOUK, 1987).

Outra justificativa para desenvolvimento desse estudo deve-se à peculiaridade dos problemas VSP, VRSP e MDVSP estarem associados diretamente à solução de problemas logísticos e de áreas afins. Sendo assim, qualquer problema que potencialmente possa ser reduzido a uma representação em rede poderá usufruir dos resultados produzidos nessa pesquisa. Esta diversidade de aplicações chamou a atenção de diversos pesquisadores em todo o mundo. Atualmente, estes pesquisadores investigam algoritmos eficientes para solução destes modelos. Dada a alta complexidade envolvida em algumas instâncias destes problemas, as pesquisas de heurísticas e metaheurísticas também foram, e ainda são, muito comuns (LI *et al.*, 2004).

Algumas das áreas mais comuns de aplicação, apenas como exemplificação, são: recolhimento de lixo residencial e hospitalar, correspondências, transporte público nos seus diversos segmentos de atuação, transporte aéreo, entrega de mercadorias, definição de sistemas de comunicação, entre outros sistemas que assemelham-se às topologias de grafos.

Considerando o MDVSP como uma aplicação ampliada do VSP, então pode-se associar todas as aplicabilidades do último ao primeiro. Da mesma forma, o MDVSP pode ser utilizado na solução de dezenas de problemas encontrados no mundo real. Na literatura é possível encontrar uma série de trabalhos que abordam o VSP (DADUNA e PAIXÃO, 1995), enquanto a literatura de MDVSP é mais escassa. Apesar da ampla literatura, os algoritmos e

modelos de VSP, infelizmente, não podem ser utilizados diretamente no tratamento de problemas MDVSP, face às peculiaridades do último não contempladas pelo primeiro. É justamente por essas características que o desenvolvimento de modelos apropriados de MDVSP se justifica. O tema, além de atual, é oportuno e relevante na área de pesquisa operacional, representando na prática grande potencial para redução de custos e maior qualidade de atendimento no setor logístico, seja público ou privado.

3 OBJETIVOS

3.1 OBJETIVO GERAL

Desenvolver uma heurística capaz de tratar o problema de escalonamento de veículos com múltiplas garagens.

3.2 OBJETIVOS ESPECÍFICOS

1. Avaliar as soluções MDVSP encontradas na literatura;
2. Desenvolver heurísticas que acelerem a solução do MDVSP;
3. Avaliar a heurística desenvolvida para solucionar o MDVSP;
4. Solucionar o MDVSP para instâncias superiores às encontradas na literatura.

4 MÉTODO DE PESQUISA

Uma das abordagens matemáticas mais conhecida para modelagem é a pesquisa operacional. O termo pesquisa operacional foi usado na segunda guerra mundial para descrever um método nascido de grupos interdisciplinares, que pretendia resolver problemas estratégicos e táticos de administração militar. Após a segunda guerra, este método tornou-se um tipo de abordagem comum na solução de problemas estruturados (SHAMBLIN e STEVENS, 1979).

A pesquisa operacional é, em si, considerada uma metodologia de pesquisa. De acordo com Andrade (1998), a pesquisa operacional é uma metodologia administrativa que agrega, em sua teoria, quatro ciências fundamentais para o processo decisório: economia, matemática, estatística e computação. Por apresentar estas características, nessa pesquisa foram aplicados os conceitos de pesquisa operacional para desenvolver os modelos capazes de tratar com o escalonamento de veículos em sistemas logísticos.

O principal objetivo dessa pesquisa é desenvolver heurísticas capazes de tratar o problema de escalonamento de veículos com múltiplas garagens. Estas heurísticas servirão para planejamento de um sistema logístico através da aplicação de cenários, com vistas a permitir a geração e análise de alternativas de escalonamento. Objetiva-se com isso, fornecer ao sistema logístico um modelo amplo que permita a escolha da ação mais conveniente e eficiente a ser tomada em modelos compostos por diversas garagens. Isso significa que o sistema poderá simular possíveis problemas e testar soluções alternativas, procurando responder as questões do tipo: o que acontecerá se (*what-if*). Sendo assim, espera-se obter um modelo capaz de melhorar o escalonamento de veículos e que, possivelmente, pode ser aplicado em outros problemas com características semelhantes.

Resumidamente, a pesquisa se apóia nos preceitos metodológicos de pesquisa o-

peracional para o desenvolvimento de um modelo matemático. Para tanto, foram necessárias quatro etapas distintas, cada uma delas relacionada diretamente com um dos objetivos específicos, a saber: i) avaliar as soluções MDVSP encontradas na literatura; ii) desenvolver heurísticas que acelerem a solução do MDVSP; iii) avaliar a heurística desenvolvida para solucionar o MDVSP; e iv) solucionar o MDVSP para instâncias superiores às encontradas na literatura. O Quadro 1 resume o método de pesquisa, enquanto na seção 4.1 cada etapa é descrita com detalhes.

ETAPA	OBJETIVO
1	**Avaliar as soluções MDVSP encontradas na literatura** 1. Realizar pesquisa bibliográfica sobre modelos MDVSP já desenvolvidos
2	**Desenvolver heurísticas que acelerem a solução do MDVSP** 1. Desenvolver heurísticas de redução do espaço de estados 2. Desenvolver heurísticas de solução para o MDVSP
3	**Avaliar a heurística desenvolvida para solucionar o MDVSP** 1. Construir um módulo capaz de gerar quadros de horários aleatórios 2. Solucionar o MDVSP com algoritmos exatos 3. Analisar e comparar as soluções heurísticas com as exatas
4	**Solucionar o MDVSP para instâncias superiores às encontradas na literatura** 1. Aplicar a heurística desenvolvida no MDVSP

Quadro 1 – Etapas da Pesquisa
Fonte: o autor (2008).

4.1 ETAPAS DA PESQUISA

Primeira Etapa – Avaliar as soluções MDVSP encontradas na literatura

Nesta fase foram revisados e selecionados alguns algoritmos e heurísticas criadas por outros autores para tratar o MDVSP. A finalidade dessa etapa foi encontrar as heurísticas e algoritmos mais conhecidos e eficientes encontrados na literatura atual. Através dos resultados obtidos nessa etapa de pesquisa foi possível efetuar a revisão bibliográfica que serviu de base para todo trabalho, conforme apresentado no capítulo 5.

Segunda Etapa – Desenvolver heurísticas que acelerem a solução do MDVSP

A segunda etapa de pesquisa consiste na criação de heurísticas que efetuem o es-

calonamento de veículos de modo satisfatório considerando múltiplas garagens. Por satisfatório entende-se toda a solução encontrada próxima a otimidade do sistema logístico. Nessa etapa da pesquisa foi elaborada uma heurística para o tratamento do MDVSP com o objetivo de melhorar as escolhas de escalonamento. Todos os algoritmos desenvolvidos nesta etapa foram avaliados e validados, conforme descrito na seção 4.3.

Pode-se dividir essa etapa em duas partes: a primeira refere-se ao desenvolvimento de procedimentos para redução dos espaço de estados do sistema logístico, a segunda à solução do MDVSP propriamente dito. Todo sistema pode ser representado por estados, aos quais correspondem as possíveis situações reais que podem assumir. Contudo, a geração de todos os possíveis estados de um sistema (espaço de estados) pode exigir um severo esforço computacional em sua construção. Deste modo, a idéia por trás de algumas heurísticas é exigir o menor esforço computacional possível selecionando um mínimo de elementos para armazenar o espaço de estados, obtendo-se um modelo de dimensão reduzida que se aproxime do comportamento original do sistema (HWANG, *et al.*, 1991).

Três procedimentos de redução do espaço de estados foram desenvolvidos. Os procedimentos executam uma redução exata do espaço de estados, ou seja, considerando uma base de dados qualquer, a aplicação dos procedimentos sempre conduz ao mesmo resultado, tendo em vista que fatores estocásticos não foram utilizados. Em termos gerais os procedimentos de redução fazem uso de critérios matemáticos e das respostas de problemas de menor complexidade (tal como o VSP) como ponto partida para definição das variáveis que devem ser mantidas ou excluídas da espaço de estados. A seção 6.1 detalha cada um dos procedimentos de redução desenvolvidos.

Considerando que os procedimentos iniciais apenas executam a redução do espaço de estados, outro procedimento foi desenvolvido para encontrar a solução propriamente dita do MDVSP. Esse procedimento trabalha de forma integrada e iterativa até que uma solução satisfatória seja encontrada. Nessa etapa tentou-se solucionar o MDVSP flexibili-

zando a restrição de chegada e saída dos veículos a mesma garagem, ao qual deu-se o nome de MDVSP com flexibilidade (ver seção 6.3). Todos os modelos desenvolvidos nessa etapa foram construídos com programação linear e suas soluções encontradas pelo software LINGO versão oito. O LINGO é um dos softwares de programação linear mais utilizados em pesquisa operacional. O algoritmo de solução utilizado em todos os modelos foi o *branch-and-bound*, garantindo que a otimidade fosse encontrada em todas as respostas obtidas. A máquina utilizada para rodar todos os resultados foi um computador DELL Latitude D630 com processador Centrino-Core2Duo (processador modelo T7300) e um gigabyte de memória RAM operando com o sistema Windows XP *service pack* 2. O LINGO foi configurado para utilizar 64 megabytes de memória RAM.

Terceira Etapa – Avaliar a heurística desenvolvida para solucionar o MDVSP

Uma vez desenvolvida a heurística para solução do MDVSP, iniciou-se a fase de avaliação. Sabe-se que a confiabilidade das análises dos resultados é decorrente, em parte, da confiabilidade da base de dados utilizada. Assim, para que a heurística desenvolvida pudesse mostrar sua capacidade de solucionar os problemas MDVSP tornou-se necessário testá-la em cenários variados. Parte-se da premissa que se a heurística desenvolvida é capaz de solucionar problemas cujos quadros de horário apresentem viagens geradas aleatoriamente, então essa mesma heurística é capaz de resolver problemas onde as viagens já encontram-se seqüencialmente organizadas. Com base nesse raciocínio, primeiramente, desenvolveu-se um módulo capaz de gerar quadros de horários cujas viagens são geradas aleatoriamente ao qual deu-se o nome de GVA (Gerador de Viagens Aleatórias). Foi através do GVA que todas as amostras utilizadas nessa tese foram geradas, em outras palavras não houve interferência por parte do autor nas amostras e nem foram selecionadas amostras com as quais os procedimentos desenvolvidos pudessem obter melhores resultados. Vale lembrar que a geração aleatória de viagens já foi utilizada em outros trabalhos (LI, 2006) e é recomendada por Carpeto, *et al*. (1989). A seção 4.2 detalha o módulo GVA apresentando suas funcionalidades.

De posse das amostras geradas pelo GVA iniciou-se o processo de avaliação dos resultados obtidos pela heurística. Entende-se que a avaliação efetiva da heurística só pode ser confiável e honesta se os resultados obtidos forem confrontados com as soluções ótimas obtidas por algoritmos exatos. Como base nisso, nessa fase tornou-se essencial o desenvolvimento e solução de modelos MDVSP por algoritmos exatos e sem qualquer forma de redução do espaço de estados. Como a heurística apresentada nessa tese dividi-se em quatro procedimentos, as comparações e avaliações foram realizadas, quando necessário, para cada procedimento individualmente. Nessas avaliações, num contexto onde a complexidade dos problemas MDVSP é de grande relevância, duas dimensões receberam maior destaque: o tempo necessário para solucionar os problemas e os custos total de deslocamento encontrados. No capítulo 6, onde explica-se detalhadamente o funcionamento de cada procedimento, mais detalhes são fornecidos sobre a amostra e avaliações realizadas. Resumidamente, as avaliações consistiram na comparação de resultados exatos e heurísticos obtidos das soluções de cenários gerados aleatoriamente, buscando-se sempre que possível avaliar, também, cada um dos procedimentos individualmente.

Quarta Etapa – Solucionar o MDVSP para instâncias superiores às encontradas na literatura

Na quarta etapa de pesquisa demonstrou-se a aplicabilidade da heurística desenvolvida resolvendo o MDVSP para instâncias superiores as encontrada na literatura. Analogamente a segunda etapa de pesquisa os modelos nessa etapa também foram gerados pelo GVA. O objetivo dessa etapa é demonstrar que a heurística permite a análise das alternativas de escalonamento de veículos para instâncias de uma ordem não encontrada na literatura.

4.2 GERADOR DE VIAGENS ALEATÓRIAS

Para testar todos os modelos desenvolvidos na pesquisa de modo a verificar a eficiência da heurística, foi necessária a implementação de um módulo responsável em gerar quadros de horários de viagens aleatórias. Esse módulo consiste na interpretação de um plano cartesiano como um região geográfica na qual são sorteadas quatro coordenadas (D_{ixV}, D_{fxV}, D_{iyV} e D_{fyV}). As coordenadas se referem a:

D_{ixV}: coordenada x para o início da viagem V;

D_{fxV}: coordenada x para o fim da viagem V;

D_{iyV}: coordenada y para o início da viagem V;

D_{fyV}: coordenada y para o fim da viagem V;

De posse de cada coordenada é possível calcular a distância de cada viagem (Tempo de Viagem - TV) pela seguinte equação:

$$TV_V = \sqrt{(D_{fxV} - D_{fyV})^2 + (D_{ixV} - D_{iyV})^2}$$

Trata-se do cálculo de distâncias diagonais no plano cartesiano através da fórmula de Pitágoras. Como nos modelos logístico existe correlação entre a distância percorrida e o tempo necessário, assume-se que as distâncias calculadas são idênticas aos respectivos tempos de viagem. Para efeitos de simplificação os valores obtidos foram truncados. Outro elemento aleatório no módulo GVA refere-se ao Tempo de Inicio (TI) de cada viagem, uma vez que o Tempo de Término (TT) pode ser definido por:

$$TT_V = TI_V + TV_V$$

Entretanto o módulo trabalha com alguns valores não aleatórios que são de responsabilidade do usuário informá-los. São eles:

- Janela de Tempo: essa variável pode ser interpretada como o horizonte de planejamento das seqüências de viagens. Sendo assim todos os valores sor-

teados para o Tempo de Início de uma viagem devem estar dentro do horizonte de planejamento. No GVA a janela de tempo é calculada através da multiplicação da maior viagem possível dentro do plano cartesiano por um valor escolhido pelo usuário. Por exemplo, se a maior viagem possível é de 100 minutos e o valor atribuído pelo usuário é igual a 15, então tem-se uma Janela de Tempo igual a 1500 minutos para planejamento das viagens. Cabe ressaltar que quanto maior o horizonte de planejamento, maior o número de viagens compatíveis em um sistema, nesse caso, o valor atribuído pelo usuário pode ser compreendido como indicador de densidade da rede logística.

- Número de garagens: trata-se do número de garagens existentes no modelo. Para o posicionamento das garagens, sorteia-se duas coordenadas (x, y) correspondentes a localização das garagens no plano cartesiano.

- Número de viagens: trata-se do número de viagens desejadas no modelo.

- Custo por veículo: trata-se do custo fixo de uso de cada veículo.

- Comprimento D_x: Trata-se do comprimento do eixo x no plano cartesiano, ou seja, todos os valores sorteados para D_{ix} e D_{fx} devem estar entre zero e D_x.

- Comprimento D_y: Trata-se do comprimento do eixo y no plano cartesiano, ou seja, todos os valores sorteados para D_{iy} e D_{fy} devem estar entre zero e D_y.

Após o sorteio de todas as viagens, o módulo inicia um segundo processo responsável pelo cálculo da Folga Total, Ociosidade e Viagem Expressa entre as viagens sorteadas. Para ilustrar esses conceitos considere duas viagens A e B, onde B pode ser executada após o término de A, a viagem A tem seu término no momento 50 e a viagem B tem seu início no momento 100. Por Folga Total (FT) entende-se o tempo total disponível desde a partida do veículo do fim da viagem A até sua chegada no início da viagem B. Sendo assim a Folga Total entre as viagens A e B pode ser calculada subtraindo-se o Tempo de Início da

viagem B pelo Tempo de Término da Viagem A, nesse caso, $FT_{AB} = TI_B - TT_A$. Por Viagem Expressa (VE) entende-se o tempo gasto com deslocamento entre as viagens A e B. O cálculo da viagem expressa pode ser dado por:

$$VE_{AB} = \sqrt{(D_{fxA} - D_{ixB})^2 + (D_{fyA} - D_{iyB})^2}$$

A ociosidade é tempo desperdiçado pelo veículo quando chega à próxima viagem de destino antes do tempo previsto, matematicamente o Tempo Ocioso (TO) pode ser definido por:

$$TO_{AB} = FT_{AB} - VE_{AB}.$$

Quando o Tempo Ocioso apresenta valor negativo, então significa que as viagens são incompatíveis, ou seja, a viagem B não pode ser realizada após a viagem A. Em seção posterior, um exemplo da interface do módulo GVA é apresentado na Figura 5 enquanto a interface da matriz de Folga Total é apresentado na Figura 7.

4.3 IMPLEMENTAÇÃO E VALIDAÇÃO COMPUTACIONAL DOS MODELOS

Para execução da pesquisa foram implementados algoritmos para o sistema operacional Windows XP utilizando a linguagem DELPHI (*Object Pascal*). A linguagem DELPHI foi escolhida, pois é aquela de melhor domínio do proponente. Todavia, cabe ressaltar que a escolha não prejudicou os resultados da pesquisa, pois a linguagem escolhida proporciona o trabalho com ponteiros e *dynamic array*, facilitando a programação de modelos lineares. Um ponteiro é uma variável que contém um endereço da memória computacional, possibilitando armazenar diversos valores para uma única variável declarada no código (ALBUQUERQUE, 1991). O *dynamic array* pode ser compreendido como uma variável vetor que pode ter seu tamanho alterado em tempo de execução (REISDORPH, 1999). Cada algoritmo e heurística

será desenvolvido em módulo independente, validado e testado conforme descrito a seguir.

A validação dos algoritmos implementados ocorreu de duas maneiras. A primeira através do acompanhamento de cada variável e procedimento dentro de cada algoritmo. Este acompanhamento é possível graças aos métodos de controle (*step over*, *trace into*, *etc.*) embutidos no aplicativo DELPHI da Borland. Deste modo, foi possível acompanhar detalhadamente cada variável dentro do programa em tempo de desenvolvimento. A segunda forma de validação deu-se através das saídas de cada algoritmo. O objetivo desta validação foi verificar se os algoritmos implementados executam corretamente as funções para as quais foram programadas. Esta forma de verificação através das saídas dos algoritmos confirma a aplicabilidade dos códigos implementados.

5 REFERENCIAL TEÓRICO

Existem vários problemas clássicos na área de pesquisa operacional que trabalham com o tema vinculado à designação de veículos em um sistema logístico. Alguns modelos buscam otimizar as rotas que devem ser percorridas pelos veículos, tais como, o Problema de Roteamento de Veículos (do inglês, *Vehicle Routing Problem* – VRP[1]) e o Problema de Reroteamento de veículos (do inglês, *Vehicle Rerouting Problem* – VRRP). Outros, por sua vez, buscam minimizar os custos de ociosidade ou espera entre viagens previamente organizadas em um quadro de horários, como por exemplo, o Problema de Escalonamento de Veículos (do inglês - *Vehicle Scheduling Problem* – VSP) e o Problema de Reescalonamento de Veículos (do inglês, *Vehicle Rescheduling Problem* – VRSP). Cada um desses problemas lida com algum aspecto logístico. O problema de roteamento busca identificar a melhor rota que vários veículos devem executar de modo a reduzir os custos operacionais, o problema de escalonamento almeja identificar qual veículo deve executar qual conjunto de viagens já pré-estabelecidas. O VRSP e o VRRP podem ser considerados, se olharmos de modo simplificado a reaplicação dos modelos VSP e VRP de forma dinâmica, face às contingências ocorridas no sistema de transporte.

Para melhor compreensão dos dois tipos de problemas considere os seguintes exemplos: (a) uma empresa de transporte público e (b) uma empresa de entrega e coleta de correspondência. No exemplo (a), não é possível redefinir as rotas dos veículos por restrições práticas do sistema de transporte. Neste exemplo, o roteamento de veículos se torna inviável face aos pontos de parada já determinados e a impossibilidade de deslocar os passageiros por rotas não definidas. No segundo exemplo (b), apesar de um possível atraso na entrega/coleta, não há empecilhos práticos que impeçam a redefinição das rotas. Estes dois exemplos deixam claro que, apesar de ambos, VRSP e VRRP, serem utilizados no

[1] O VRP foi primeiramente formulado por Dantzig e Ramser (1959) e tornou-se um dos problemas de análise combinatória mais estudados no mundo (POOT, KANT e WAGELMANS, 2002; CORDEAU *et al.*, 2002).

planejamento logístico, os modelos possuem aplicabilidade distinta que são peculiares a certos sistemas.

Em linhas gerais, os modelos VRP e VSP são utilizados para planejamento do sistema de transporte e os modelos VRRP e VRSP são utilizados na tratabilidade de contingências. Embora os problemas de VRP e VSP sirvam para organizar planejar sistemas logísticos, cabe ressaltar que não são os únicos com essa aplicabilidade. Alguns problemas tradicionais de pesquisa operacional, tais como o Problema do Carteiro Chinês, modelos de Fluxo Máximo e Mínimo, entre outros, também podem ser utilizados com o mesmo objetivo. Boffey (1984) ainda complementa afirmando que se a capacidade de carga de um veículo é muito grande (suficientemente capaz de atender todas os pontos de demanda), então o VRP assemelha-se ao problema do caixeiro viajante. É fácil chegar a esta conclusão, pois se a capacidade excede a demanda, basta apenas um veículo para concluir o percurso e, neste caso, estamos diante do clássico problema do caminho mais curto com retorno ao ponto de saída.

Nesse capítulo aborda-se os principais modelos relacionados com o escalonamento de veículos: VSP, MDVSP e VRSP. A escolha em referenciar esses três modelos deve-se as relações existentes entre eles. Sabe-se que o MDVSP é a solução do problema VSP considerando diversas garagens. Por sua vez o VRSP trata da solução dinâmica de um VSP, ou seja, trata da reprogramação dos escalonamentos em caso de contingência. Conforme aborda-se mais adiante, pode-se compreender VRSP como uma instância do MDVSP. Sendo assim, as soluções encontradas na literatura para solucionar o VRSP podem ser aplicáveis na solução do MDVSP, razão pela qual optou-se em referenciar os três modelos.

Pelo exposto, dividi-se esse capítulo da seguinte forma: a seção 5.1 dedica-se à apresentação de conceitos utilizados em modelos VSP, MDVSP e VRSP; após apresentação dos conceitos, as seções 5.2, 5.3 e 5.4, apresentam em detalhes, respectivamente, os mo-

delos de VSP, MDVSP e VRSP propriamente ditos; a seção 5.5 mostra a relação entre os modelos MDVSP e VRSP; finalizando o capítulo, a seção 5.6 revisa a literatura e aborda os principais trabalhos científicos desenvolvidos na área.

5.1 CONCEITOS RELACIONADOS A ESCALONAMENTO DE VEÍCULOS

Percebe-se a existência de uma estreita relação entre os modelos VSP, MDVSP e VRSP. Essa seção tem a finalidade de familiarizar o leitor com alguns termos peculiares à revisão de literatura em problemas de escalonamento de veículos. A seguir cada termo é explicado.

- Veículo (*vehicle*): o termo refere-se a qualquer meio de transporte, variando conforme a aplicação do problema. Por exemplo, ônibus em transporte público, caminhão de lixo em sistema de coleta, avião em tráfego aéreo, etc.

- Passageiros (*passengers*): neste ensaio teórico utiliza-se o termo genericamente para o conteúdo transportado por um veículo. Deste modo, o termo “passageiro” pode referir-se a pessoas, produtos, lixo, combustível, entre outros, dependendo do contexto da aplicação do modelo.

- Viagem: (*trip*): no VRP o termo viagem refere-se à rota que um veículo deve executar, estando associada a um conjunto de demandas que precisam ser atendidas. No VSP, onde as rotas já estão pré-definidas, o termo viagem corresponde a uma série de viagens (*trip sequence*) previamente organizadas num quadro de horários (*scheduling*).

- Viagem expressa (*deadhead*): O termo inglês *deadhead* refere-se ao período de deslocamento entre duas viagens ou o período em que um veículo está movendo-se entre a garagem e o ponto de saída de uma viagem. Este trajeto não agrega va-

lor à empresa, pois o veículo trafega vazio. Utiliza-se o termo “viagem expressa” para designar estes percursos. O termo “viagem expressa” é freqüentemente utilizado por empresas privadas do ramo de transporte público, e por isso optou-se por esta tradução.

- Garagem (*depot*): como próprio nome indica, o termo refere-se ao ponto inicial de saída de todos os veículos e para o qual os mesmos devem retornar quando cumprirem suas viagens.

5.2 O VEHICLE SCHEDULING PROBLEM (VSP)

O Problema de Escalonamento de Veículos baseia-se na atribuição de um conjunto de veículos para a realização de um conjunto de viagens, onde cada viagem está associada a um único veículo buscando-se a minimização dos custos operacionais (BAITA *et al.*, 2000). Por se tratar de um problema de designação, o VSP é considerado um problema clássico de otimização. Apesar da semelhança com o problema clássico de designação, o VSP possui uma série de características que o tornam mais difícil e desafiante de ser resolvido, tais como as restrições de prazos e capacidade. Os prazos em um VSP referem-se ao tempo de início e término de execução das viagens. Os locais de saída e chegada dos veículos também devem ser informados no problema. A restrição de capacidade, como o próprio nome sugere, refere-se à capacidade que cada veículo tem para atender a demanda de uma viagem.

O objetivo do VSP é minimizar os custos operacionais e de investimento em veículos. Entretanto, como definido por Baita *et al.* (2000), os custos operacionais podem ser expresso de várias maneiras tais como tempo ocioso, tempo de deslocamento sem passageiros (*deadhead*), consumo de combustível, número de motoristas exigidos, etc. Esta diversidade de opções na definição de custos caracteriza o VSP como um problema de deci-

são multicritério. Para exemplificar esta diversidade, Baita *et al.* (2000) listou os principais critérios de otimização em VSP aplicados em empresas de transporte público. A seguir reproduziu-se estes critérios em ordem de relevância, entretanto tomou-se a liberdade de generalizar os conceitos apresentados, de modo que os objetivos pudessem se ajustar a qualquer tipo de aplicação VSP.

(a) Minimizar o número de veículos: este é o principal objetivo, consistindo na minimização do tamanho da frota (BERTOSSI, CARRARESI e GALLO, 1987). O número de veículos afeta fortemente planos de investimentos devido ao alto capital que será imobilizado.

(b) Minimizar o número de mudanças de viagens: Quando não existe uma restrição que obrigue o veículo a executar a mesma rota todo o dia, é aconselhável minimizar o número de mudanças de linhas, pela conveniência dos motoristas. Mudanças constantes de linha não são bem vistas pelos motoristas, além disso, um VSP que considera esta questão facilita a decisão subseqüente de escalonar a tripulação (*Crew Scheduling Problem* - CSP).

(c) Minimizar o número ou tempo de viagens expressas: viagens expressas, por definição, geram para as empresas apenas custos operacionais (combustível e gastos com funcionários). Segundo Baita *et al.* (2000), além dos custos operacionais, os passageiros não gostam de olhar veículos circulando vazios enquanto esperam, causando desgaste para a imagem da empresa. Somado a isso, geralmente, os incentivos governamentais ao transporte público estão associados apenas aos trajetos regulares, conseqüentemente o deslocamento em viagens expressas não gera lucros nem recebe apoio financeiro.

(d) Minimizar o tempo ocioso dos veículos em seus terminais: da mesma forma que as viagens expressas, o tempo de espera em um terminal não agrega valor para a empresa e gera o custo operacional com funcionários. Não obstante, o

tempo de espera nos terminais gera insatisfação nos passageiros que aguardam o deslocamento do veículo, impactando negativamente na imagem da empresa.

Estes quatro principais objetivos de otimização podem ser conflitantes algumas vezes (BAITA *et al.*, 2000). Na maioria dos casos é inviável minimizar o número de veículos e o tempo de ociosidade simultaneamente. Existe uma relação de perda entre estes objetivos, o gestor pode optar em colocar um veículo adicional para reduzir o número de trocas de veículos entre as linhas ou para reduzir o tempo gasto com viagens expressas, mas estaria deixando de minimizar o tamanho da frota.

Outro exemplo de objetivos conflitantes refere-se à escolha entre reduzir o tempo de ociosidade nos terminais ou de deslocamento com viagens expressas. Quando há tempo suficiente entre duas viagens, os veículos podem ter autorização para retornar a garagem, o que aumenta o deslocamento com viagens expressas, porém reduz os riscos de assalto em terminais ou a insatisfação dos passageiros ao verem um veículo parado enquanto aguardam na fila de embarque. Encontrar a relação de custo exata ou aproximada destas relações de troca pode ser o maior obstáculo para a tomada de decisão.

Outra dificuldade intrínseca à aplicação do VSP refere-se aos horários de pico com maior movimento de veículos. Estes picos causam engarrafamento e atrasos nos grandes centros urbanos e, conseqüentemente, o tempo de espera e os custos operacionais elevam-se nestes horários. A solução para reduzir o tempo de espera é o acréscimo de veículos no sistema de transporte, mas tão logo este período termine, os veículos adicionais tornam-se ociosos. Sendo assim, os critérios de otimização de um VSP são fortemente dependentes do contexto de aplicação e conflitante em muitos casos. É comum nestes casos, que diversos cenários sejam gerados e avaliados para que o decisor tenha maior capacidade de avaliação e, conseqüentemente, tome a decisão mais acertada possível, ainda que sabidamente dispendiosa devido à natureza de alguns sistemas de transporte (BAITA *et al.*, 2000).

Segundo Huisman *et al.* (2004) geralmente o VSP é executado após a elaboração de um novo quadro de horário de viagens e a solução encontrada será utilizada enquanto este novo quadro de horários for válido. Infelizmente, para horários estáticos, o atraso de alguma viagem perpetua-se nas viagens subseqüentes. Uma das formas de resolver este problema é atribuir uma folga de tempo entre as viagens que garanta o cumprimento dos horários estabelecidos, mesmo diante das contingências naturais do sistema. Entretanto, estas folgas representam ociosidade quando o sistema apresenta funcionamento normal, aumentando os custos operacionais. O exemplo de um quadro de horários necessário para aplicação do VSP foi extraído de Huisman *et al.* (2004), conforme mostra o Quadro 2.

Viagem	Horário de Saída	Horário de Chegada	Local de saída	Local de chegada
1	09:00	10:00	B	A
2	09:15	10:00	C	A
3	10:05	11:05	A	B
4	10:15	11:00	A	C

Quadro 2 – Exemplo de quadro de horários
Fonte: Huisman *et al.* (2004)

O exemplo mostra como nem sempre é óbvia a determinação das folgas entre viagens. Suponha que a empresa trabalhe com uma garagem contendo dois veículos, existem duas soluções estáticas para o exemplo. Na primeira solução considera-se as viagens 1 e 3 para o primeiro veículo e as viagens 2 e 4 para o segundo veículo. A segunda alternativa considera as viagens 1 e 4 para o primeiro veículo e 2 e 3 para o segundo veículo. Na primeira solução, caso ocorra um atraso superior a cinco minutos na viagem 1 seria preciso solicitar um terceiro veículo para impedir o atraso da viagem 2. Na alternativa 2 nossa folga permitiria atrasos de até 15 minutos. Entretanto, se atrasos não acontecessem, aumentar-se-ia a ociosidade dos veículos no sistema. No Mundo real nem sempre é possível prever em quais viagens ocorrerão os atrasos e, portanto as soluções estáticas nem sempre são eficientes. Em virtude disso, a possibilidade de redirecionar veículos - solução dinâmica - apresenta forte vantagem sobre as soluções estáticas.

Se o VSP é resolvido dinamicamente, ou seja, com a possibilidade de redirecionar

veículos alterando a designação atribuída inicialmente, pode-se anular o “efeito dominó” de atrasos entre viagens reduzindo, concomitantemente, a ociosidade característica das folgas entre viagens. Atualmente a maioria das empresas não trabalha com modelos de reescalonamento, simplesmente, enviando um veículo adicional direto da garagem para evitar atrasos encadeados (LI *et al.*, 2005). Ressalta-se que, este procedimento de envio direto de um veículo da garagem para compensar uma contingência no sistema pode ser considerado uma resposta VRSP, tendo em vista que ocorreu, na prática, reescalonamento de veículos no sistema logístico. A seção 5.4 dá mais detalhes sobre o VRSP.

5.2.1 Formulação Matemática do VSP

Existem diversas formulações matemáticas para o problema de VSP. A formulação matemática apresentada foi extraída de Freling *et al.* (2001). Matematicamente, o VSP pode ser definido da seguinte maneira: considere s_i, c_i, st_i e ct_i como sendo o local de saída, local de chegada, horário de saída e horário de chegada, respectivamente, que um veículo precisa cumprir para executar a viagem i. Duas viagens i e j são compatíveis quando um mesmo veículo pode executar estas duas viagens em seqüência, isto é, $ct_i + viag(c_i, s_j) \leq st_j$ onde $viag(c_i, s_j)$ é tempo ou distância gasta com a viagem expressa entre as localidades c_i e s_j. Uma seqüência de viagens é possível se cada par consecutivo de viagens nesta seqüência são compatíveis.

Considere $N = \{1, 2,... n\}$ um conjunto de viagens numeradas em ordem crescente de horário de saída e considere $E = \{(i, j) \mid i < j$, i e j são compatíveis, $i \in N, j \in N\}$ o conjunto de arcos correspondentes as viagens expressas. Considere que os nós s e d representam, ambos, a garagem localizada em d. Pode-se definir uma rede VSP como $G = \{V, A\}$ que é uma rede dirigida e acíclica com os nós $V = N \cup \{s, t\}$ e os arcos $A = E \cup (s \times N) \cup (N \times t)$. Um caminho de s para t na rede representa um possível escalonamento de um veículo e a completa possibilidade de escalonamentos de veículos é um conjunto de cami-

nhos disjuntos de s para t, tal que cada nó em N é coberto.

Considere c_{ij} o custo da viagem equivalente ao arco $(i, j) \in A$, que geralmente está associada ao tempo de ociosidade da viagem. Usando a variável de decisão y_{ij}, onde $y_{ij} = 1$ se um veículo executou a viagem j logo após a viagem i, e 0 caso contrário. O VSP pode ser formulado como um problema de *quasi-assignment* como segue:

$$\min \sum_{(i,j)\in A} c_{ij} y_{ij} \tag{1}$$

$$\sum_{j:(i,j)\in A} y_{ij} = 1 \qquad \forall i \in N \tag{2}$$

$$\sum_{i:(i,j)\in A} y_{ij} = 1 \qquad \forall j \in N \tag{3}$$

$$y_{ij}\{0,1\} \qquad \forall (i,j) \in A \tag{4}$$

As restrições (2) e (3) garantem que cada viagem é atribuída a exatamente uma viagem predecessora e sucessora, isto é, as restrições garantem que a rede será particionada em um conjunto de caminhos disjuntos de s a t. A função objetivo (1) minimiza os custos operacionais das viagens expressas. A restrição (4) define as variáveis do modelo como binárias.

5.3 O MULTIPLE DEPOT VEHICLE SCHEDULING PROBLEM (MDVSP)

A própria nomenclatura do Problema de Escalonamento de Veículos com Múltiplas Garagens (do inglês, *Multiple Depot Vehicle Scheduling Problem* - MDVSP) já revela adequadamente a peculiaridade adicional deste problema. Ao contrário do tradicional VSP, também chamado SDVSP (do inglês, *Single Depot Vehicle Scheduling Problem*), o MDVSP trabalha com mais de um ponto de reabastecimento ou garagens.

Segundo Huisman *et al.* (2004), Bodin *et al.* (1983) e Ball e Bodin (1983) o MDVSP possui, além das restrições do SDVSP, outras duas restrições fundamentais: (a) algumas viagens precisam ser realizadas por veículos de determinadas garagens; (b) cada veículo pertence a apenas uma garagem. Estas restrições são importantes, pois ao final do horizonte de tempo estudado no problema, cada veículo deve retornar a sua garagem de origem, ainda que durante a execução das viagens ele possa reabastecer ou parar em garagens as quais não pertença.

Estas restrições adicionais fazem com que o MDVSP torne-se um problema NP-Difícil (HAGHANI e BANIHASHEMI, 2002). Apesar da complexidade desta instância do problema, o MDVSP é bastante utilizado em diversas áreas de aplicação. Como pode ser observado em Huisman *et al.* (2004), o MDVSP pode ser aplicado em quase todos os problemas de logística que trabalhem com quadro de horários. Existem duas abordagens seguidamente utilizadas para solução do MDVSP, baseadas na formulação do SDVSP, chamadas: *Cluster First - Scheduling Second* (CARRARESI e GALLO, 1984), e *Scheduling First-Cluster Second* (GAVISH e SHIFLER, 1978). A maioria dos trabalhos aborda o problema de MDVSP dividindo-o em problemas menores de SDVSP, onde cada um desses problemas pode ser formulado ou solucionado separadamente.

O MDVSP pode ser formulado como um problema de programação inteira. Contudo, devido ao grande número de restrições e variáveis, a aplicação em problemas reais faz com que os modelos de programação inteira não possam encontrar uma solução exata. Sendo assim, procedimentos heurísticos são necessários para encontrar uma solução aceitável para estes problemas. De modo geral os objetivos de otimização do MDVSP são os mesmo do SDVSP. Entretanto adiciona-se a estes problemas a possibilidade de minimizar-se os custos de investimentos com garagens.

5.3.1 Formulação Matemática do MDVSP

A formulação matemática para o MDVSP apresentada a seguir foi extraída da tese

de Huisman (2004). Huisman (2004), por sua vez, baseou-se nos trabalhos de Bodin *et al.* (1983) e Ribeiro e Soumis (1994) para formulação matemática do MDVSP. Matematicamente, o MDVSP pode ser definido da seguinte maneira: considere um conjunto de grafos G^k=(V^k, A^k) para cada garagem $k \in K$, onde V^k e A^k um conjunto de vértices e arcos, respectivamente. Considere que V^k possui um vértice para cada viagem $i \in T$ e um par de vértices $o(k)$ e $d(k)$ representando, respectivamente, o início e o fim do escalonamento de um veículo associado a uma garagem k. Deste modo, $V^k = \{o(k), d(k)\} \cup T$. Considere que A^k tem três tipos de arcos: saída, chegada e de conexão. Os arcos de saída são definidos como ($o(k)$, i) para cada viagem $i \in T$. Simetricamente, os arcos de chegada são definidos como (i, $d(k)$) para cada viagem $i \in T$. Por fim, define-se um arco de conexão (i,j) para cada par de viagens, onde $i \in T$ e $j \in T$, tal que $a_i + l_i + e_{ij} < a_j$, onde a indica o tempo de início de uma viagem, l indica a duração da viagem e e indica a duração da viagem expressa entre duas viagens. O custo do arco (i,j) $\in A^k$ é registrado em c_{ij}. A variável v_k indica o número de veículos disponível por garagem, tal que $k \in K$. Assim, o MDVSP pode se formulado como segue:

$$\min \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij} x_{ij}^k \tag{1}$$

$$\sum_{k \in K} \sum_{j:(i,j) \in A^k} x_{ij}^k = 1 \qquad \forall i \in T \tag{2}$$

$$\sum_{j:(o(k),j) \in A^k} x_{o(k),j}^k \leq v_k \qquad \forall k \in K \tag{3}$$

$$\sum_{j:(j,i) \in A^k} x_{j,i}^k - \sum_{j:(i,j) \in A^k} x_{i,j}^k = 0 \qquad \forall i \in V^k \left| \{o(k), d(k)\}, k \in K \right. \tag{4}$$

$$x_{i,j}^k \in \{0;1\} \qquad \forall (i,j) \in A^k, k \in K \tag{5}$$

A restrição (2) garante que cada viagem é executada no modelo por exatamente um veículo. A restrição (3) determina o número máximo de veículo por garagem. A restrição (4) refere-se à condição de igualdade entre o número de veículos que chega e sai de cada vértice da rede. A função objetivo (1) minimiza os custos operacionais entre viagens. A

restrição (5) define as variáveis do modelo como binárias.

5.4 O VEHICLE RESCHEDULING PROBLEM (VRSP)

O VRSP é utilizado em problemas logísticos que sofrem algum tipo de contingência em seu planejamento inicial e por isso torna necessário o reescalonamento de veículos para atendimento de viagens já pré-estabelecidas, respeitando-se o quadro de horários inicialmente planejado. Devido à contingência, geralmente, ocorrem atrasos de veículos dentro do sistema de transporte e, portanto, é preciso que as viagens de cada veículo sejam reprogramadas para evitar atrasos nas viagens subseqüentes. Alguns trabalhos (HUISMAN *et al*, 2004) definem VRSP como sinônimo para VSP dinâmico.

Existe uma série de peculiaridades abordadas pelo VRSP que não são contempladas pelos modelos VSP. Para entender melhor as diferenças entre os dois problemas, as características adicionais encontradas no VRSP são listadas conforme segue:

(a) O momento em que ocorreu a interrupção de uma viagem;
(b) A capacidade dos veículos de substituírem o veículo inoperante;
(c) O VSP considera todos os veículos inicialmente na garagem, enquanto no VRSP os veículos já estão em movimento e cumprindo suas rotas;
(d) A posição de cada veículo no momento da interrupção;
(e) Compatibilidade de itinerários entre as viagens.

O Problema de Reescalonamento de Veículos consiste, em linhas gerais, na reaplicação do modelo VSP frente a uma contingência. O VRSP surge quando ocorre a interrupção de uma viagem previamente estabelecida. Assim, pode-se afirmar que o VRSP é a resposta a um VSP que não pôde ser cumprido conforme planejado. A inviabilidade de aplicação da solução originária de um VSP deve-se a diversas contingências encontradas

em problemas do mundo real que impedem a conclusão de uma ou mais rotas planejadas inicialmente, tais como acidentes e emergências, quebra de veículos, congestionamentos, entre outros fatores. Este ambiente de contingências gera situações dinâmicas que precisam de soluções em tempo real. Então, pode-se imaginar o VRSP como uma abordagem dinâmica da versão clássica do VSP, tendo em vista que as atribuições de veículos são geradas dinamicamente (LI *et al.*, 2004).

Pode-se definir o VRSP como segue: considerando uma garagem e uma série de viagens com tempos de início e fim previamente estabelecidos, bem como os tempos de viagem entre cada par de localizações, e considerando uma viagem não concluída, deve-se encontrar o menor custo possível de reescalonamento onde: (a) cada veículo executa uma seqüência possível de viagens; (b) todos os pontos de demanda não atendidos (se existirem) devem ser visitados.

As contingências existentes em aplicações VRSP sugerem, naturalmente, a adição de terminologia inexistente no VSP. Os termos são apresentados nos tópicos a seguir e foram retirados da literatura estrangeira. Infelizmente, não foi encontrada literatura nacional que utilizasse termos semelhantes e, por isso, tomou-se a liberdade de traduzi-los. Cuidou-se para que o termo original em inglês fosse colocado entre parênteses para sua melhor identificação. Felizmente a maioria dos termos é auto-explicativa, tornando as descrições mais detalhadas desnecessárias.

(a) Veículo inoperante (*disable vehicle*): é o veículo que se tornou inoperante e não pode concluir a viagem inicialmente planejada;
(b) Viagem interrompida (*cut trip*): o termo se refere à viagem que não pôde ser executada pelo veículo inoperante;
(c) Viagem assistida (*backup trip*): refere-se à demanda não atendida na viagem interrompida;
(d) Viagem atual (*current trip*): é a viagem que um veículo qualquer está percorren-

do, incluindo os movimentos com ou sem passageiros;

(e) Veículo assistente (*backup vehicle*): é o veículo designado a executar a viagem assistida;

(f) Ponto de interrupção (*breakdown point*): é o momento em que a viagem sofreu a contingência.

Assim o VRSP consiste em encontrar um veículo capaz de concluir a viagem que não pode ser executada. Este veículo, que recebe a denominação de Veículo Assistente, precisa concluir sua viagem atual e, logo após, executar a viagem assistida (pontos de demanda que não foram atendidos). Dentro destas definições, pode-se afirmar que as viagens i e j são compatíveis se um mesmo veículo pode alcançar o ponto de início da viagem j depois de concluir a viajem i, onde i é a viagem atual e j é a viagem assistida. Assim, cada par de viagens i e j torna-se uma possível alternativa de solução do VRSP. Estas alternativas podem ser interpretadas como uma rede de soluções (*feasible networks*), uma para cada veículo da frota ainda operante. Sendo assim, a decisão mais importante de um modelo de VRSP é encontrar o veículo que irá realizar a viagem interrompida substituindo o veículo que se tornou inoperante.

O VSP e VRSP também se diferenciam no que diz respeito ao critério de otimização. Enquanto, como vimos nas seções anteriores, o VSP preocupa-se, geralmente, com a minimização do capital imobilizado e custo de operações tendo objetivos de planejamento, o VRSP na maioria das vezes consiste no reescalonamento de veículos para o atendimento de clientes com o mínimo custo de espera quando um dos veículos inicialmente destinados para o cumprimento da viagem não pôde executar a tarefa, seja por quebra, ou qualquer outra razão (LI *et al*, 2004).

Face à condição intrínseca do VRSP de minimizar os tempos de espera, justifica-se plenamente sua tratabilidade em tempo real. Por ser tratar de um reescalonamento, pode se afirmar que os clientes não atendidos em uma viagem devem ser atendidos conforme o

plano inicial e, portanto, os algoritmos de solução devem apresentar uma solução rápida condizente com a realidade do problema (LI *et al*, 2004).

Apesar de suas peculiaridades, existe uma série de fatores comuns entre o VSP e o VRSP. Por exemplo, o VRSP trabalha com as mesmas restrições de janela de tempo do VSP, ou seja, a atribuição de um novo veículo a uma viagem interrompida terá que respeitar os horários de início e fim da viagem.

5.4.1 Formulação matemática do VRSP

Para melhor exemplificar a formulação matemática do VRSP primeiramente é apresentado um exemplo de quadro de horários e possíveis soluções para de reescalonamento de veículos em caso de contingência. O exemplo apresentado foi retirado de Li *et al*. (2005). Considere o seguinte quadro de horários para cinco viagens:

Viagem	Início	Fim	Duração
1	8	14	6
2	1	16	15
3	18	25	7
4	20	28	8
5	3	11	8

Quadro 3 – Exemplo de quadro de horários
Fonte: Li *et al*. (2005)

Considere ainda as seguintes informações sobre as viagens:

- A contingência ocorreu na viagem 1 no tempo 11;
- As viagens 1 e 2 possuem itinerários comuns entre os tempos 9 e 13. Por itinerário comum entende-se que uma parcela do caminho percorrido pelos dois veículos é a mesma. Por exemplo, num sistema logístico, vários ônibus utilizam os mesmos corredores de tráfego.
- O tempo de viagens expressas (deslocamento sem passageiros) entre quaisquer duas viagens ou entre uma viagem e a garagem é igual a 4.
- Os nós S (*start*) e T (*termination*) representam, respectivamente, os pontos de largada e chegada dos veículos.

A Figura 1 representa o escalonamento inicial do sistema (solução VSP). Percebe-se que 3 veículos são necessários para atender as demandas. O primeiro veículo executa as viagens 2 e 4. O segundo veículo executa a viagem 5. O terceiro veículo executa a viagem 1 e 3. Na Figura 1 e na Figura 2, o quadrante esquerdo indica o número da viagem, o quadrante direito superior o tempo de chegada, e o quadrante direito inferior o tempo de saída. Todas as arestas possuem tempo de deslocamento 4.

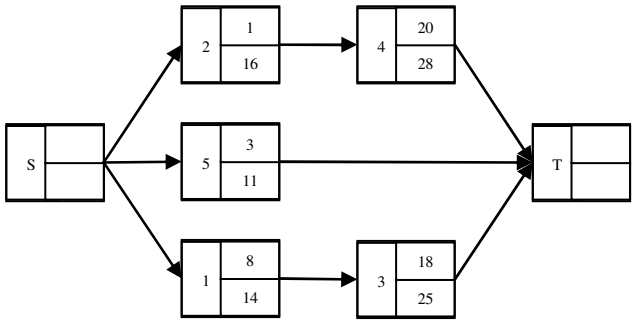


Figura 1 – Solução inicialmente proposta – VSP
Fonte: o autor (2008)

Considerando que a contingência ocorreu na viagem 1 e no tempo 11, sabe-se que ainda restam 3 unidades de tempo para conclusão da viagem 1. Tem-se as seguintes alternativas de reescalonamento:

a) <u>utilizar o primeiro veículo que está executando a viagem 2</u>: como os veículos que executam as viagens 1 e 2 possuem itinerários comuns no tempo 11, então uma alternativa viável seria utilizar esse veículo como backup. Essa é a melhor solução para o problema, pois não envolve a necessidade de reescalonamento de veículos. A questão é que nem sempre os pontos de contingência irão coincidir com os itinerários comuns dos veículos, exigindo do modelo a possibilidade de outras soluções. Desta forma, a partir de agora vamos desconsiderar a possibilidade de itinerário comum entre as viagens 1 e 2.

b) <u>Sem itinerário comum entre as viagens 1 e 2, uma segunda alternativa viável seria enviar um veículo da garagem</u>: como o acidente ocorreu no tempo 11 e o deslocamento do veiculo da garagem até o ponto de contingência é igual a 4 e ainda restam 3 unidades

para finalizar a viagem 1, então, se enviássemos um veículo diretamente da garagem, a viagem 1 só seria concluída no tempo 18 (18=11+4+3). Pela solução VSP inicial, o veiculo que executou a viagem 1 deveria executar a viagem 3, entretanto o tempo de deslocamento entre viagens é igual a 4. Sendo assim a viagem 3 iniciaria apenas em 22 (22=18+4). Conclusão, mesmo existindo a possibilidade de enviar um veículo diretamente da garagem, a solução exigiria um reescalonamento de veículos para não perpetuar o atraso sofrido na viagem 1 para as viagens subseqüentes. Conforme exemplificado na Figura 2.

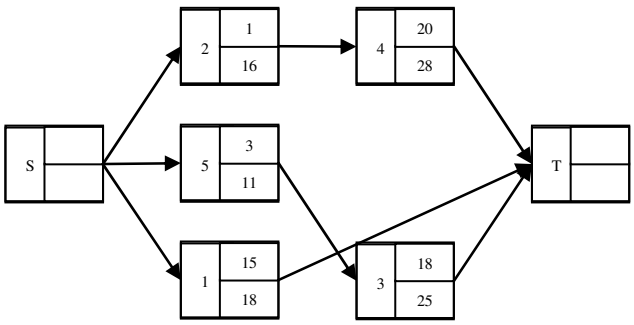


Figura 2 – Possível solução VRSP
Fonte: o autor (2008)

Pelo exemplo acima fica claro que o veículo que executou a viagem 5 pode ser reescalonado para a viagem 3 (11+4<18). E pela simplicidade do exemplo, essa é a única resposta viável, uma vez que o veículo 2 não poderia atender a viagem 3 (16+4>18). Isso leva-nos a uma peculiaridade na modelagem dos problemas VRSP inexistente nos modelos VSP: o tempo permitido de atraso nas viagens. Quanto maior o atraso permitido na execução das viagens, maior as possibilidades de reescalonamento dos veículos. Contudo, a maioria dos modelos VRSP trabalha com a seguinte condição: as viagens no sistema, com exceção da viagem interrompida, não podem sofrer atrasos. Essa condição é imposta aos modelos para evitar que atrasos em uma viagem sejam repassados para as viagens posteriores e para evitar o aumento na complexidade do sistema. Entretanto, é importante ressaltar que existe a possibilidade dos veículos em funcionamento no sistema não conseguirem absorver as viagens posteriores do veículo que se tornou inoperante.

No exemplo anterior existia uma possibilidade de solução, pois o veículo que executou a viagem 5 também pôde executar a viagem 3. Entretanto, se a viagem 5 terminasse após o tempo 15 então, esse veículo ficaria impossibilitado de cumprir a viagem 3 (15 + 4 > 18). Nesse caso, teríamos que, obrigatoriamente, assumir a possibilidade de cancelamento ou atraso na viagem 3.

O motivo dessa reflexão é melhor compreender a solução matemática de um modelo VRSP. Para cada tolerância de atraso em uma viagem dentro do sistema logístico, uma nova rede de possibilidades de reescalonamentos pode ser criada. Como existe uma série combinações de valores e viagens que podem ser atrasadas, conseqüentemente, teremos um conjunto de redes de reescalonamento, a qual dá-se o nome de *Feasible Networks*.

Posto isso, o que diferencia a formulação e solução matemática dos modelos VSP e VRSP é que o segundo pode ser considerado a solução do primeiro sobre diversas condições, ou seja, o VRSP pode ser resolvido com a solução de múltiplos modelos VSP (uma solução para cada possível *Feasible Network*). Agora que a relação entre o VSP e VRSP é mais bem compreendida, segue a notação matemática do modelo VRSP segundo o modelo desenvolvido por Li *et al*. (2005).

O modelo VSP pode ser definido da seguinte forma: considere $N = \{1, 2, \ldots n\}$ um conjunto de viagens numeradas em ordem crescente de horário de saída e considere $E = \{(i, j) \mid i < j$, i e j são compatíveis, $i \in N, j \in N\}$ o conjunto de arcos correspondentes as viagens expressas. Considere que os nós s e d representam, ambos, a garagem localizada em d. Pode-se definir uma rede VSP como $G = \{V, A\}$ que é uma rede dirigida e acíclica com os nós $V = N \cup \{s, t\}$ e os arcos $A = E \cup (s \times N) \cup (N \times t)$. Um caminho de s para t na rede representa um possível escalonamento de um veículo e a completa possibilidade de escalonamentos de veículos é um conjunto de caminhos disjuntos de s para t, tal que cada nó em N é coberto.

Considere c_{ij} o custo da viagem equivalente ao arco $(i, j) \in A$, que geralmente está

associada ao tempo de ociosidade da viagem. Usando a variável de decisão y_{ij}, onde y_{ij} = 1 se um veículo executou a viagem j logo após a viagem i, e 0 caso contrário.

Como a solução do VRSP é solução de diversos VSP. Então ao contrário do VSP onde tínhamos um único grafo $G = \{V, A\}$, para o problema VRSP teremos $G(x) = \{V, A(x)\}$ onde x indica a *Feasible Network*. Assim a formulação matemática para o VRSP é dada por:

$$\min G\{\min \sum_{(i,j)\in A(x)} c_{ij} y_{ij} + d_x\}$$

$$\sum_{j:(i,j)\in A(x)} y_{ij} = 1 \qquad \forall i \in N$$

$$\sum_{i:(i,j)\in A(x)} y_{ij} = 1 \qquad \forall j \in N$$

$$y_{ij}\{0,1\} \qquad \forall (i,j) \in A(x)$$

Onde d_x é custo associado ao atraso permitido na *Feasible Network* $G(x)$. Através da modelagem é facilmente perceptível que o modelo VRSP é idêntico solução de múltiplos modelos VSP.

5.5 O VRSP COMO INSTÂNCIA DO MDVSP

Existe uma relação direta entre o MDVSP e VRSP. Basicamente, ao se considerar o sistema de transporte já em funcionamento, ou seja, com todos os veículos já em movimento executando as suas rotas, e diante de uma contingência, pode-se interpretar o VRSP como um MDVSP baseando-se na seguinte analogia:

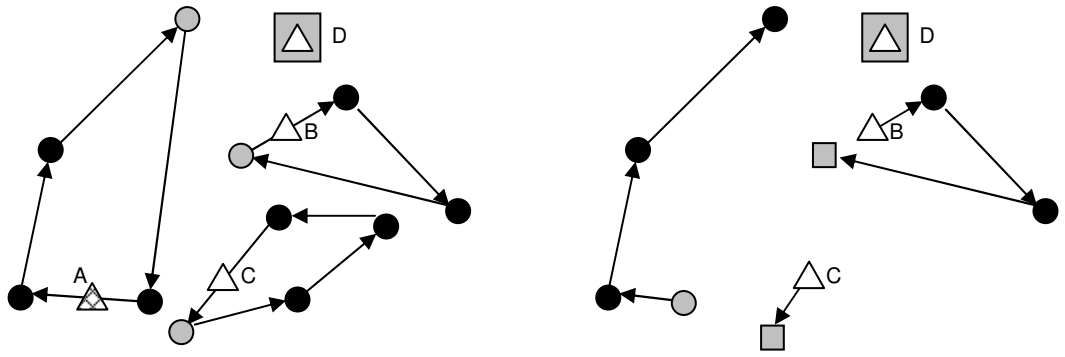
(a) no momento da contingência (ponto de interrupção) a posição de cada veículo no sistema pode ser interpretada como uma garagem com capacidade igual a um;

(b) as demandas não atendidas em cada rota, a partir do momento de interrupção,

poderiam ser interpretadas como novas viagens do sistema.

Sendo assim, com as devidas alterações, o modelo de VRSP pode ser considerado uma instância do MDVSP. Entretanto, se cada posição do veículo torna-se uma garagem teremos uma quantidade de garagens igual ao número de veículos do sistema. Essa peculiaridade torna o número de garagens bastante grande e, conseqüentemente, aumenta a dificuldade de solução dos problemas. Infelizmente, na literatura existente, a maioria dos trabalhos de MDVSP utiliza entre duas e cinco garagens. A seção 5.6 revisa com mais detalhes a literatura de MDVSP.

A Figura 3 esboça uma solução de VRSP transformada em um modelo MDVSP. Considere as circunferências pretas como sendo pontos de demanda e as circunferências cinzas como sendo pontos de saída e chegada dos veículos. As arestas indicam o sentido do deslocamento dos veículos. Os triângulos em branco são os veículos em funcionamento e o triângulo hachurado o veículo inoperante. Cada quadrado menor equivale a uma garagem e o quadrado maior representa a garagem onde se encontra o veículo assistente. Nesse problema o custo de espera pelo atendimento tem maior importância, pois o exemplo refere-se a um sistema de transporte público, onde, ao chegar no ponto final da viagem, o veículo aguarda e, só então, recomeça seu trajeto.



(a) solução VSP com contingência

(b) solução VRSP como MDVSP

Figura 3 – Transformação do VRSP em MDVSP
Fonte: o autor (2008)

A Figura 3a mostra um VSP onde o veículo A tornou-se inoperante (triângulo hachurado). A solução mais usual e simples seria o reenvio do veiculo assistente D até o ponto de interrupção (veículo A). Entretanto essa solução não é adequada, pois os passageiros do veículo A precisariam esperar o deslocamento do veículo D (que é bastante demorado devido à distância entra A de D).

Outra possível solução seria enviar o veículo C para atender A tão logo ele terminasse sua viagem. Ao chegar à parada final o veículo C não aguardaria como usualmente ocorre, mas seguiria diretamente em auxílio de A. Concomitantemente, D se deslocaria para dar continuidade às viagens de C. Se o tempo de descanso de C na parada (antes de recomeçar o trajeto) for maior que o tempo de deslocamento de D até C, então as viagens de C não sofreriam atraso algum na resposta VRSP e o tempo de espera dos passageiros do veículo A seria minimizado.

Esta solução de reescalonamento ilustra bem as vantagens do VSP dinâmico sobre o estático. Aproveitando o exemplo, a solução via MDVSP torna cada ponto de chegada e saída de um veículo uma possível garagem, conforme mostra a Figura 3b. Realizado o reescalonamento, cada veículo poderia continuar a executar os quadros de horários e rotas previamente estabelecidos conforme a solução originalmente planejada.

5.6 TRABALHOS DESENVOLVIDOS EM VSP, MDVSP E VRSP

Nessa seção são abordadas conjuntamente as revisões de literatura para os modelos VSP, VRSP e MDVSP. Como já citamos anteriormente o VSP é um problema clássico de otimização (BAITA *et al.*, 2000) e pode ser considerado como um problema de designação de veículos para um conjunto predeterminado de viagens com tempos e locais de saída e chegada.

Sabe-se que algoritmos eficientes para algumas instâncias do VSP já foram desenvolvidos e aplicados, principalmente, para problemas onde o número de veículos iguala-se ao número de viagens e uma única garagem é utilizada como ponto de reabastecimento (BODIN *et al.*, 1983; CARRARESI e GALLO, 1984). Contudo, na prática, na maioria das vezes existem mais viagens para serem executadas do que veículos, obrigando que cada veículo responsabilize-se por uma seqüência de viagens (*sequence trip*). Sendo assim, as aplicações no mundo real, freqüentemente, são complexas demais para serem modeladas, considerando suas particularidades e, muitas vezes, na literatura estas particularidades não são abordadas.

Entretanto, por se tratar de um problema de designação, o VSP pode facilmente ser transformado em outros problemas, tais como: fluxo mínimo, *quasi-assignment* e emparelhamento. Diversos livros na área de pesquisa operacional demonstram a transformação de um problema de designação em outros correlatos, como, por exemplo, Boffey (1984). Tendo em vista esta variedade de modelos, muitos autores já têm trabalhado com o VSP nas mais diversas abordagens. Bokinge e Hasselstrom (1980) propuseram uma abordagem utilizando os modelos de fluxo com mínimo custo que apresentou uma significativa redução no número de variáveis, mas, em contrapartida, aumentou no número de restrições. Amico *et al.* (1993), Jonker e Volgenant (1986), Song e Zhou (1990) propuseram soluções para o VSP com a utilização de modelos e algoritmos que encontrassem o caminho mais curto em uma rede.

Paixão e Branco (1987) desenvolveram um algoritmo que trabalha com modelos de *quasi-assignment* para solução do VSP. Este algoritmo apresenta complexidade O(n^3) e foi desenvolvido especificamente para os modelos SDVSP. Freling *et al.* (2001) fez uso do modelo *quasi-assignment* desenvolvido e resolveu o VSP com um algoritmo paralelo chamado de *forward/reverse auction algorithm*. Atualmente o modelo de *quasi-assignment* e o *forward/reverse auction algorithm* são considerados as melhores abordagens para solucionar o SDVSP (FRELING *et al.*, 2001). Além desta aplicação, outros algoritmos para solucio-

nar o VSP em algumas de suas instâncias podem ser encontrados em Baita *et al.* (2000), Bodin e Golden (1981), Daduna e Paixão (1995), Freling *et al.* (2001). Entretanto, as soluções apresentadas são estáticas e computacionalmente ineficientes para uma resposta em tempo real.

Sabe-se que quando existe interrupção de viagem é preciso que o modelo redirecione os veículos do sistema em tempo real. Neste contexto, os modelos de VRP dinâmico chamaram a atenção de pesquisadores no final dos anos oitenta (ICHOUA *et al.*, 2000). Recentes pesquisas sobre VRP dinâmicos podem ser encontradas em Psaraftis (1995), Ghiani *et al.* (2003) e Gendreau e Potvin (1998). Entretanto, este tipo de modelo para recuperação automática de viagens é relativamente novo como estratégia operacional e, conseqüentemente, a literatura ainda é escassa nesta área de estudo. A maioria das pesquisas conduzidas nesta área está relacionada com problema de tráfego aéreo.

A literatura sobre interrupção no tráfego aéreo inclui Carlson (2000), Lettovsky (1997), Rosenberger *et al.* (1987), e Teodorovic e Stojkovic (1995). Nestes trabalhos foram desenvolvidos modelos que redirecionam e recriam as rotas das aeronaves minimizando os custos de viagem e de cancelamentos. Todavia, uma vez que os algoritmos foram desenvolvidos por companhias aéreas específicas, envolvendo um número limitado de viagens e aeronaves, não podem ser utilizados para o tratamento de problemas genéricos que envolvem milhares de rotas e centenas de veículos.

Naturalmente, apesar da relativa novidade dos modelos VRSP, alguns autores já trabalharam no desenvolvimento de modelos genéricos. Em seu trabalho, Ichoua *et al.* (2000) desenvolveu heurísticas baseadas na Busca Tabu para resolver problemas de despacho de veículos em tempo real. Yang *et al.* (2004) elaborou uma estrutura geral para o VRP dinâmico, na qual novas solicitações poderiam chegar durante a execução das operações e o tempo de viagem era considerado como uma variável. Modelos de programação estocástica também têm sido utilizados para tratar com a incerteza no roteamento e reesca-

lonamento de veículos (LAPORTE e LOUVEAUX, 1998). Apesar de todos estes autores terem estudado aspectos dinâmicos do VSP, as melhores contribuições para a solução do SDVRSP são encontradas em Huisman *et al.* (2004) e Li *et al.* (2004). Huisman *et al.* (2004), propôs uma abordagem para o VRSP resolvendo uma seqüência de problemas de otimização. Li *et al.* (2004) desenvolveu um algoritmo paralelo baseado no *auction algorithm* para reescalonamento da frota de veículos. Neste artigo, o problema de reescalonamento é modelado como diversos problemas VSP separados, onde um veículo deve dar assistência à viagem interrompida terminando de visitar as demandas não atendidas pelo veículo que se tornou inoperante. No ano seguinte, Li *et al.* (2005) desenvolveu um sistema de apoio à decisão baseado em seu modelo de tratamento do VRSP.

Algoritmos paralelos têm mostrado eficiência computacional, mesmo para grandes problemas e por isso, eles têm potencial de uso em ferramentas automatizadas. Entretanto, devido à complexidade do problema, existe uma lacuna entre a existência de algoritmos eficientes e suas aplicações em problemas do mundo real (DESROCHERS *et al.*, 1999). Para uma aplicação efetiva destes algoritmos é necessário contar com muita experiência. Além disso, a implementação dos algoritmos desenvolvidos não é direta, visto que necessidades específicas de algumas companhias de transporte e logística precisam ser consideradas e atendidas. Como resultado, poucas companhias utilizam sistemas de reescalonamento automatizado.

Assim, o desenvolvimento de uma ferramenta informatizada capaz de fornecer um ambiente amigável para os usuários e que, concomitantemente, enfatize a flexibilidade, eficiência e adaptabilidade, tornou-se um aspecto importante em termos do uso de estratégias de escalonamento de veículos em tempo real. Embora alguns sistemas de apoio à decisão já tenham sido desenvolvidos na área de roteamento (BASNET *et al.*, 1996; NUSSBAUM *et al.*, 1997; RUIZ *et al.*, 2004) e escalonamento (BAITA *et al.*, 2000) de veículos, eles não abordam a problemática de VRSP.

No que se refere aos estudos de MDVSP, os primeiros trabalhos apareceram no início dos anos 70 e diversas alternativas de solução através de métodos heurísticos foram propostas até os anos 90. Entre essas propostas destacam-se os estudos de Bodin *et al*. (1978), Bodin *et al*. (1983), Bertossi *et al*. (1987), Mesquita e Paixão (1992), Lamatsch (1992), e Dell'Amico *et al*. (1993). A tentativa de solucionar os problemas com múltiplas garagens de forma exata ocorreu nos anos 90 com aumento na capacidade de processamento dos computadores, onde destacam-se os trabalhos de Carpaneto *et al*. (1989), Ribeiro e Soumis (1994), Löbel (1997), Löbel (1998), Hadjar *et al*. (2006), e Kliewer *et al*. (2006).

Apesar da quantidade de pesquisadores na área é difícil, considerando a complexidade do MDVSP, encontrar trabalhos que consigam solucionar instâncias superiores a 800 viagens. Segundo Hadjar *et al*. (2006) esse é o máximo de viagens que se pode resolver para instâncias não estruturadas, ou seja, geradas aleatoriamente. Contudo para instâncias estruturadas já foi possível resolver problemas com até 7000 viagens, conforme mostram os resultados atingidos por Löbel (1997), Löbel (1998) e Kliewer *et al*. (2006). Por instâncias estruturadas entende-se aquelas que geralmente facilitam os procedimentos de busca da solução ótima, em outras palavras, trata-se de instâncias na maioria das vezes retiradas de exemplos reais na área de transporte público. Nessas instâncias, é fácil determinar as viagens sucessoras, tendo em vista que, quando uma viagem termina já encontra-se outra viagem pronta para iniciar.

Em linhas gerais os melhores trabalhos encontrados na literatura foram desenvolvidos por Bodin *et al*. (1983), Odoni *et al*. (1994), Desrosiers *et al*. (1995) e Desaulniers e Hickman (2007) e Huisman (2004). Entre esses o de maior destaque é de Huisman (2004) que conseguiu solucionar o problema MDVSP para instâncias não estruturadas considerando até 1500 veículos e 8 garagens utilizando o algoritmo de *column generation*.

6 PROPOSTA DE HEURÍSTICA PARA SOLUÇÃO DO MDVSP

Conforme abordou-se nos capítulos anteriores, o MDVSP é considerado pela literatura um problema NP difícil, conseqüentemente a garantia de solução ótima para o problema não pode ser encontrada em tempo polinomial. Razão pela qual não são encontradas na literatura pesquisas que tratem o MDVSP considerando números superiores a 1500 viagens ou 8 garagens.

Considerando a dificuldades intrínsecas ao problema, se propôs o desenvolvimento de heurísticas capazes de solucionar o MDVSP para instâncias superiores e com tempos inferiores àqueles encontrados na literatura. Pode se dividir a heurística apresentada nessa tese, em duas etapas: a primeira voltada à redução do espaço de estados do problema, a segunda voltada à identificação da solução MDVSP propriamente dita através da definição das seqüências de viagens que devem ser executadas considerando o espaço de estados já reduzido. Para auxiliar no entendimento desse capítulo, a primeira etapa da heurística foi dividida em procedimentos, conforme mostra o Quadro 4.

Primeira Etapa - Algoritmo de redução do espaço de estados	
1º Procedimento	Redução pelo critério estatístico: busca identificar viagens de alto custo que possuem baixa chance de pertencer à solução ótima do problema através do critério de dimensionamento de classes na geração de histogramas
2º Procedimento	Definição das viagens iniciais e finais: busca identificar as possíveis viagens candidatas para viagens iniciais e finais de cada seqüência de viagens através das soluções individuais de cada problema VSP
3º Procedimento	Exclusão de variáveis nunca selecionadas: exclui do espaço de estados as viagens que jamais foram escolhidas nas soluções individuais dos modelos VSPs
Segunda Etapa - Algoritmo de solução MDVSP	
4º Procedimento	Acréscimo de restrições de convergência e definição do limite inferior: trata-se da solução do MDVSP de modo iterativo, onde relaxa-se a restrição de saída e chegada de um veículo à mesma garagem.

Quadro 4 – Descrição da heurística
Fonte: o autor (2008).

Para facilitar a leitura e compreensão da heurística, estruturou-se esse capítulo na mesma ordem com a qual os quatro procedimentos são executados. Assim cada um dos procedimentos é explicado em uma seção separada, apesar de interligados e voltados a um objetivo comum - solucionar o MDVSP. Em cada seção explica-se a origem, definições e

funcionamento do procedimento, bem como sua implantação computacional. O fluxograma mostrado na Figura 4 detalha os procedimentos computacionais que são apresentados nesse documento. No capítulo “Considerações Finais” a Figura 4 é reapresentada para comentários sobre a complexidade e possibilidade de pesquisas futuras.

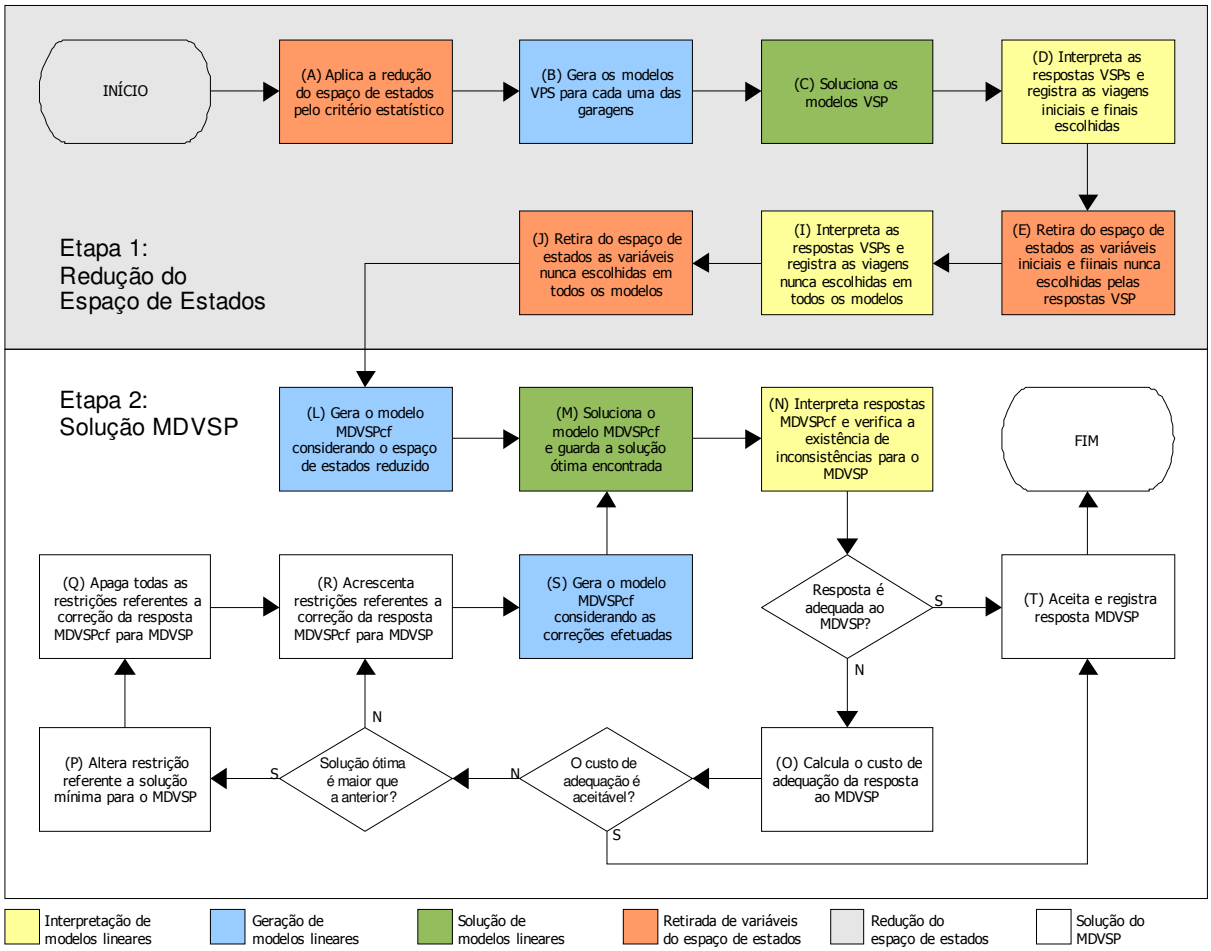
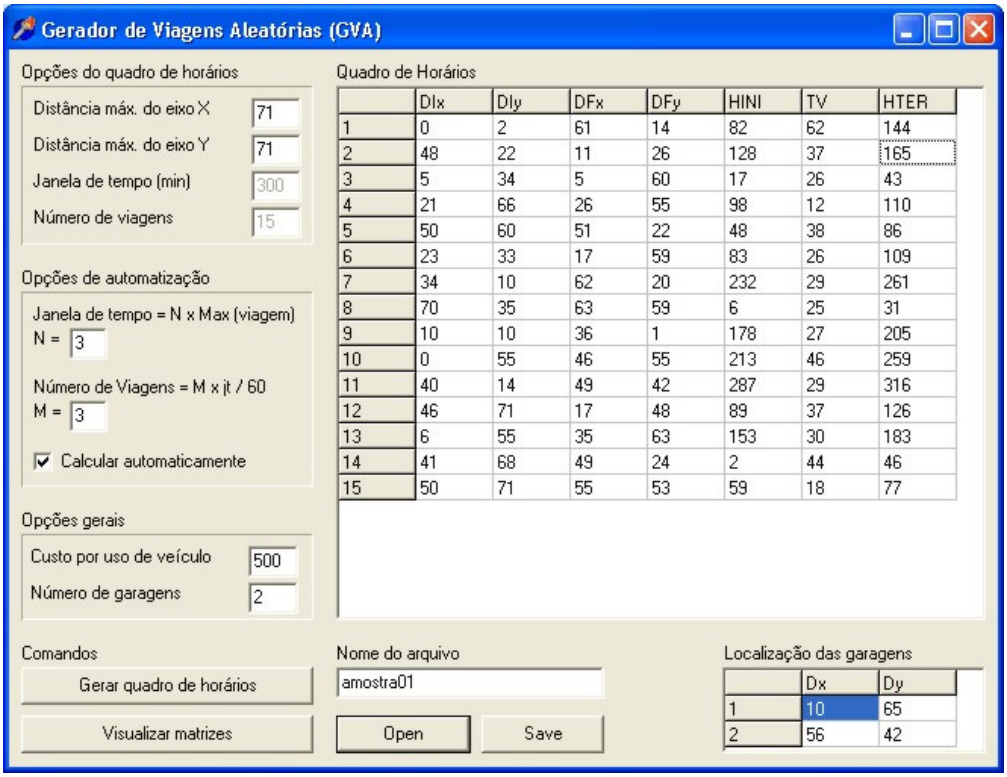


Figura 4 – Fluxograma da heurística
Fonte: o autor (2008).

Finalmente, para exemplificar e facilitar o entendimento da parte computacional implementada julgou-se oportuno utilizar um exemplo com apenas quinze viagens e duas garagens, conforme detalha a Figura 5. Esse exemplo é utilizado na explicação de todos os cinco procedimentos que compõem a heurística apresentada nessa tese.



	DIx	DIy	DFx	DFy	HINI	TV	HTER
1	0	2	61	14	82	62	144
2	48	22	11	26	128	37	165
3	5	34	5	60	17	26	43
4	21	66	26	55	98	12	110
5	50	60	51	22	48	38	86
6	23	33	17	59	83	26	109
7	34	10	62	20	232	29	261
8	70	35	63	59	6	25	31
9	10	10	36	1	178	27	205
10	0	55	46	55	213	46	259
11	40	14	49	42	287	29	316
12	46	71	17	48	89	37	126
13	6	55	35	63	153	30	183
14	41	68	49	24	2	44	46
15	50	71	55	53	59	18	77

	Dx	Dy
1	10	65
2	56	42

Figura 5 – Exemplo aleatório de MDVSP com quinze viagens e duas garagens
Fonte: o autor (2008).

As seções 6.1.1, 6.1.2 e 6.1.3 tratam dos procedimentos voltados a redução do espaço de estados do problema, enquanto a seção 6.3 trata do procedimento para solucionar o MDVSP propriamente dito. As seções 6.2 e 6.4 apresentam os resultados obtidos, respectivamente, com a redução do espaço de estados e com a solução do MDVSP para instâncias superiores àquelas encontradas na literatura.

6.1 PRIMEIRA ETAPA – ALGORITMO DE REDUÇÃO DO ESPAÇO DE ESTADOS

O espaço de estados representa todas as possibilidades de resposta de um problema. Em problemas de alta complexidade esse espaço de estados atinge grandeza exponencial, das quais apenas uma ou algumas soluções podem ser consideradas as chamadas soluções ótimas ou satisfatórias. Diante desse quadro a redução do espaço de estados é de grande relevância. Reduções adequadas permitem que problemas de complexidade expo-

nencial possam ser resolvidos sem comprometer a solução ótima, ou, pelo menos garantindo que soluções próximas à ótima sejam encontradas. O ponto chave na redução do espaço de estados é trabalhar com espaços menores que contenham essas soluções. Tais reduções, na prática, alteram as instâncias de problemas reduzindo-as de modo a permitir sua tratabilidade e, conseqüentemente, que soluções ótimas ou satisfatórias sejam encontradas em menos tempo. Os três próximos procedimentos descritos tratam justamente da redução do espaço de estados para o MDVSP, são eles:

1) Redução pelo critério estatístico – busca identificar viagens de alto custo que possuem baixa chance de pertencer à solução ótima do problema através do critério de dimensionamento de classes na geração de histogramas;
2) Definição das viagens iniciais e finais – busca identificar as possíveis viagens candidatas para viagens iniciais e finais de cada seqüência de viagens através das soluções individuais de cada problema VSP;
3) Exclusão de variáveis nunca selecionadas – exclui do espaço de estados as viagens que jamais foram escolhidas nas soluções individuais dos modelos VSPs.

Para implementação desses procedimentos foi desenvolvido um módulo capaz de executar as reduções do espaço de estados – ao qual foi dado o nome de Módulo de Redução. O módulo de redução é responsável pela leitura do arquivo gerado pelo módulo GVA (Gerador de Viagens Aleatórias) apresentado na metodologia. Após a leitura do arquivo, o módulo executa a redução do espaço de estados. A Figura 6 mostra a interface do módulo de redução, enquanto no Apêndice B encontra-se o código implementado.

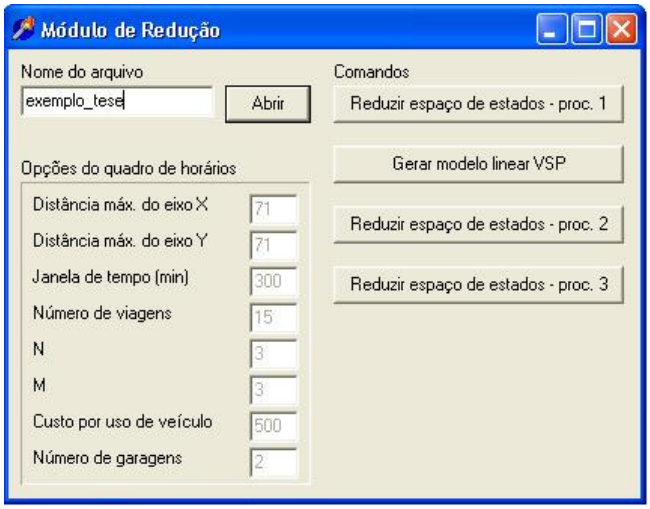


Figura 6 – Interface do módulo de redução
Fonte: o autor (2008).

Nas três seções subseqüentes apresenta-se cada um dos procedimentos de redução do espaço de estados sem impedir que a solução ótima ou satisfatória seja encontrada, conforme os resultados apresentados na seção 6.2.

6.1.1 Redução pelo critério estatístico

Em problemas de escalonamento de veículos é natural que algumas viagens estejam geograficamente mais distantes de outras. Além da questão geográfica, existem também questões de incompatibilidade de horários. Isso significa que viagens próximas geograficamente podem estar distantes em relação ao tempo em que devem iniciar, enquanto outras podem ser compatíveis em termos de horários, mas inviáveis geograficamente. Sendo assim é comum que muitas variáveis, seja por uma questão de distância ou de horário, tornem-se custosas demais para serem consideradas como alternativas viáveis em modelos que buscam a otimização das seqüências em que essas viagens são executadas. Esse procedimento busca justamente identificar tais variáveis com o intuito de descartá-las e, assim, acelerar a busca pela resposta do MDVSP. Num primeiro momento o procedimento busca identificar viagens de maiores custos desconsiderando os deslocamentos até as garagens, tanto de partida como de chegada, ou seja, apenas os deslocamentos entre viagens são considerados por esse procedimento. Seguindo o exemplo de MDVSP escolhi-

do com quinze viagens e duas garagens, a Figura 7 mostra a matriz de folga total entre as viagens, onde as células vazias representam viagens incompatíveis por questões de horário. Sendo assim, por exemplo, é possível perceber através da Figura 7 que a viagem expressa entre o fim da viagem três e início da viagem dois tem um custo (duração) de 85 minutos. Por outro lado, é impossível executar a viagem quatro após finalizar a viagem dois pela incompatibilidade de horários (célula vazia). A Figura 7 ainda mostra duas matrizes na parte inferior, essas matrizes correspondem aos custos de deslocamento entre as garagens e o início e fim de cada viagem.

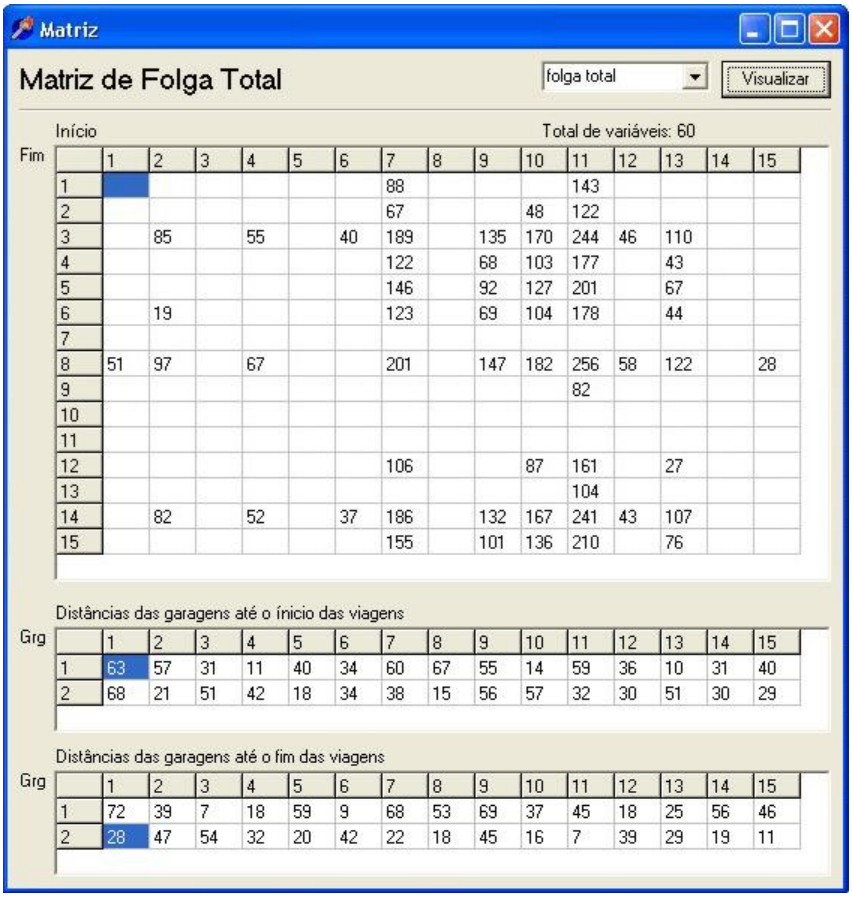


Figura 7 – Interface da matriz de folga total
Fonte: o autor (2008).

Sabe-se que o modelo VSP pode ser resolvido como um modelo de designação. Em modelos de designação, apenas uma resposta por linha e coluna são consideradas, mesmo que o problema possua um número elevado de viagens. Considere que um problema qualquer tenha n viagens, então apenas uma viagem em n é escolhida enquanto n-1

viagens são descartadas como resposta para cada linha e coluna do modelo. Em outras palavras, em modelos com n viagens, tem-se aproximadamente (n x n)/2 variáveis das quais apenas n são escolhidas para compor a solução ótima. Entende-se que o processamento de tantas variáveis é desnecessário para encontrar uma solução satisfatória para o MDVSP. Posto isso o primeiro procedimento procura escolher um grupo de m variáveis por linha e coluna, obedecendo ao critério de menor custo, que acredita-se serem as potenciais respostas do modelo e por isso àquelas que devem permanecer no espaço de estados. Todas as demais (n-m) variáveis por linhas e colunas, de acordo com o procedimento devem ser descartadas como possíveis soluções.

Quando adota-se esse tipo de procedimento surge a seguinte questão: qual valor de m é o mais adequado para manter o espaço de estados com soluções satisfatórias e concomitantemente acelere o tempo de solução do modelo?

Se o valor de m é baixo em relação ao número de viagens, então limita-se o espaço de estados demasiadamente podendo prejudicar a busca por uma solução satisfatória. Por exemplo, ao se considerar m=1, então ter-se-ia apenas uma alternativa em n para tomarmos nossa decisão. Isso significa encontrar a resposta de forma direta simplesmente pegando o valor mais baixo por linha ou coluna. Ainda nem sempre o valor escolhido na linha com menor custo será necessariamente o mesmo valor escolhido para a coluna. Podendo gerar uma resposta inadequada para a solução do problema. Por outro lado um valor m próximo a n não iria contribuir significativamente para a redução do espaço de estados e, conseqüentemente, seria de pouca utilidade ao procedimento.

Para solucionar a questão do valor de m, procurou-se entender cada linha e coluna de uma matriz de folga total como uma distribuição amostral. Entendendo cada linha e coluna dessa forma adotou-se o critério estatístico para construção de histogramas como valor mais adequado para m. Os estatísticos já enfrentavam um problema similar ao tentar dimensionar o número de classes mais adequado para dividir uma amostra de modo a encontrar

um histograma representativo a essa amostra. Se o número de classes é grande ou pequeno demais tende-se a espalhar ou agrupar os valores amostrais de forma inadequada no histograma. A Figura 8 mostra três histogramas referentes a uma mesma amostra, com números de classes distintos. O primeiro mostra um número baixo de classes, o segundo um número adequado e o terceiro um número de classes excessivas.

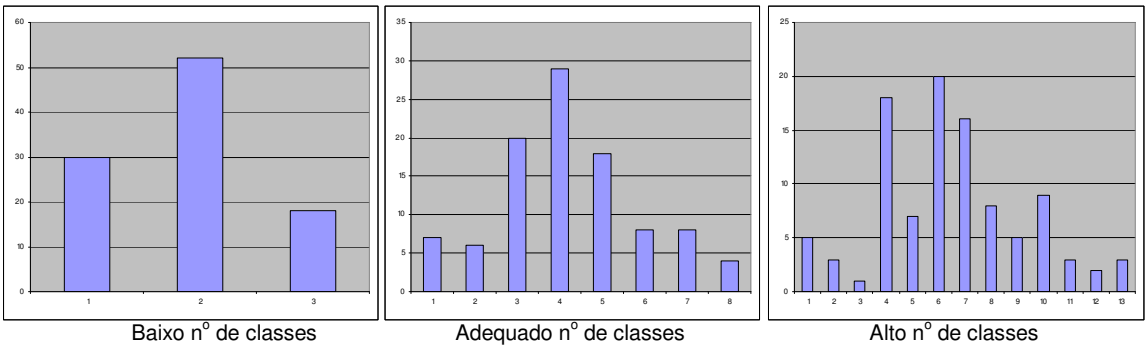


Figura 8 – Histogramas por número de classes
Fonte: o autor (2008).

Segundo (MARTINS, 2002) existem vários critérios para definição do número de classes para elaboração de um histograma. Um desses critérios pode ser obtido tirando-se a raiz quadrada do número de observações amostrais. Nesse caso, se temos 100 amostras é adequado que o número de agrupamentos seja igual a dez, com cada agrupamento contendo em média 10 valores amostrais, por exemplo. Se cada linha e coluna de um modelo MDVSP são entendidas como uma distribuição amostral, então para definição do valor de m aplicou-se o mesmo critério definido por Martins (2002). Isso significa que o procedimento irá considerar como viagens candidatas à solução do modelo apenas as m melhores respostas em cada linha e coluna da matriz de folga total. No exemplo, o valor de m é igual a quatro (~3,87), pois existem 15 viagens a serem executadas, devendo-se marcar as quatro melhores respostas em cada linha e coluna da matriz. As células em amarelo indicam as melhores respostas em cada linha, enquanto as células marcadas em verde indicam a melhor resposta para a linha e coluna simultaneamente. Por exemplo, o valor 85 marcado em verde é considerado um dos quatro menores tanto na linha 3 como na coluna 2, já o valor 146 em amarelo mostra um dos quatro menores valores apenas na linha 5. A Tabela 1

apresenta a redução de espaço de estados completa realizada pelo primeiro procedimento.

Tabela 1 – Viagens com menor custo por linha e coluna

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1							88				143				
2							67			48	122				
3		85		55		40	189		135	170	244	46	110		
4							122		68	103	177		43		
5							146		92	127	201		67		
6		19					123		69	104	178		44		
7															
8	51	97		67			201		147	182	256	58	122		28
9											82				
10															
11															
12							106			87	161		27		
13											104				
14		82		52		37	186		132	167	241	43	107		
15							155		101	136	210		76		

Fonte: o autor (2008).

O algoritmo responsável por essa redução tem complexidade O(n^2) por tratar-se de uma busca direta do menores m valores em cada coluna e linha. A Figura 9 mostra o algoritmo utilizado.

```
For i←1 to n do                                  {para cada linha faça}
    For j←1 to n do                              {olhe os valores em todas as colunas}
        If M(i,j)≤ l then                        {se o valor da célula for menor que l então}
            Begin
            L_i←L_i-{l}                          {retira-se l do conjunto L_i}
            L_i←L_i + {M(i,j)}                   {acrescenta-se M(i,j) no conjunto L_i}
            For k←1 to m-1 do                    {procura o novo valor de l no novo conjunto L_i}
                If L_i(k)≤ L_i(k+1) then l← L_i(k+1)
            End
l←ø                                              {reinicia l}
For j←1 to n do                                  {para cada coluna faça}
    For i←1 to n do                              {olhe os valores em todas as linhas}
        If M(i,j)≤ l then                        {se o valor da célula for menor que l então}
            Begin
            C_j←C_j-{l}                          {retira-se l do conjunto C_j}
            C_j←C_j + {M(i,j)}                   {acrescenta-se M(i,j) no conjunto C_j}
            For k←1 to m-1 do                    {procura o novo valor de l no novo conjunto C_j}
                If C_j(k)≤ C_j(k+1) then l← C_j(k+1)
            End
```

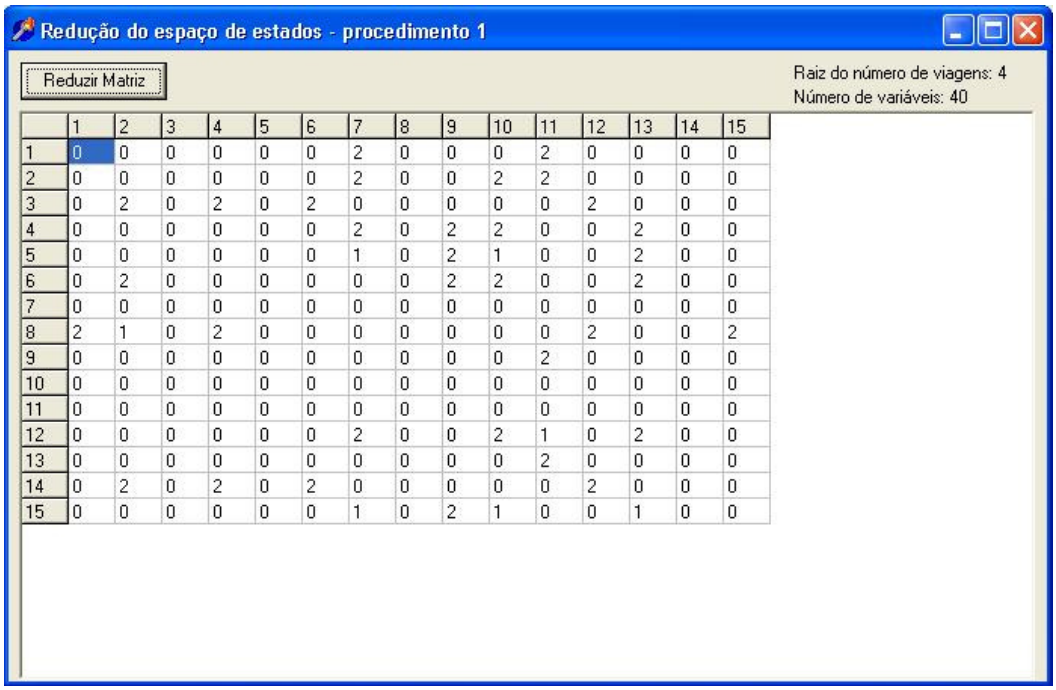
Figura 9 – Algoritmo do primeiro procedimento de redução do espaço de estados
Fonte: o autor (2008).

Onde:

m : é o número de células que devem ser selecionadas por linha e coluna;
n : é o número de viagens;
M: é a matriz de folga total n x n;
L_i: é o vetor das m menores células selecionadas na linha i;
C_j: é o vetor das m menores células selecionadas na coluna j;
l : é uma variável auxiliar que guarda a célula de maior valor em L_i e P_j;

i, j : são indexadores da matriz M;
k : é o indexador da matriz L_i e P_j.

O primeiro comando do módulo de redução trata da redução pelo critério estatístico conforme mostra a Figura 10. Por definição, na Figura 10 os valores iguais a dois indicam que a célula está entre as m menores tanto para a linha como para a coluna, enquanto o valor um indica que a célula está entre as m menores para a linha ou para a coluna, os valores iguais a zero indicam que a célula não pertence ao espaço de estados reduzido. No Apêndice C encontra-se todo o código implementado para desenvolvimento do procedimento.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	2	0	0	0	2	0	0	0	0
2	0	0	0	0	0	0	2	0	0	2	2	0	0	0	0
3	0	2	0	2	0	2	0	0	0	0	0	2	0	0	0
4	0	0	0	0	0	0	2	0	2	2	0	0	2	0	0
5	0	0	0	0	0	0	1	0	2	1	0	0	2	0	0
6	0	2	0	0	0	0	0	0	2	2	0	0	2	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	2	1	0	2	0	0	0	0	0	0	0	2	0	0	2
9	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	2	0	0	2	1	0	2	0	0
13	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
14	0	2	0	2	0	2	0	0	0	0	0	2	0	0	0
15	0	0	0	0	0	0	1	0	2	1	0	0	1	0	0

Figura 10 – Interface do primeiro procedimento de redução do espaço de estados
Fonte: o autor (2008).

Como esse algoritmo considera apenas a matriz de folga total entre viagens, ou seja, os custos de deslocamento de ida e volta às garagens são ignorados. Não é necessário medir a eficiência dessa redução de espaço de estados considerando as múltiplas garagens envolvidas nos problemas MDVSP. Com isso a eficiência desse procedimento de redução do espaço de estados foi testada considerando uma gama de problemas VSP que foram gerados aleatoriamente. Outra razão de desconsiderarmos as múltiplas garagens neste primeiro procedimento é poder comparar os resultados da redução do espaço de estados

com o espaço de estados não reduzido em instâncias com valores até 1500 viagens. Tal comparação só é possível em instâncias desse tamanho tratando-se de problemas VSP por possuírem solução ótima encontrada em tempo polinomial através do algoritmo húngaro de complexidade assintótica O(n^3) ou do algoritmo *auction* de complexidade assintótica O(n^2 log n) (FRELING *et al*, 2001). Como a matriz de folga total entre viagens independe do número de garagens de um problema, entende-se que a avaliação desse procedimento através do modelo VSP é adequada. Sendo assim, acredita-se que o procedimento de redução do espaço de estados não afeta significativamente a solução ótima em relação à solução ótima encontrada considerando o espaço de estados completo. A Tabela 2 mostra os resultados obtidos quando os modelos foram executados no software LINGO.

Tabela 2 – Resultados do primeiro procedimento de redução do espaço de estados

amostra		solução sem redução			solução com redução		
Número de viagens	Janela de tempo	Número de variáveis	Solução ótima	Número de iterações	Número de variáveis	Solução ótima	Número de iterações
500	5	92432	48023	7684	12285	48023	3995
500	10	107915	40290	8699	12866	40290	3871
500	15	113828	43682	8647	13061	43682	3365
500	20	116885	35273	9554	12847	35273	3401
500	25	118683	38015	7376	12801	38015	3251
700	5	180126	65591	12821	20241	65591	6420
700	10	210548	50329	12482	20947	50329	6261
700	15	222886	48320	11910	20791	48320	5580
700	20	228206	51669	16251	20961	51669	5222
700	25	232043	47843	14644	21048	47843	5607
900	5	295474	75477	20254	29647	75477	10068
900	10	347976	63196	21222	30656	63196	8478
900	15	367280	64505	20092	30902	64505	7709
900	20	377001	58330	23983	31064	58330	7893
900	25	382772	60909	22896	30860	60909	8038
1100	5	441608	90332	25961	40289	90332	12791
1100	10	523061	75037	24706	41019	75037	11090
1100	15	549089	62873	29140	41190	62873	11846
1100	20	563883	59879	26761	40885	59879	11846
1100	25	571032	63970	30070	41377	63970	11153
1300	5	623417	103206	29293	51055	103206	16047
1300	10	729746	79815	35029	52617	79815	14703
1300	15	766663	72176	33357	53379	72176	14819
1300	20	785831	70974	36443	53212	70974	13793
1300	25	798223	71977	38537	52820	71977	13730
1500	5	820545	118698	38782	62111	118698	23109
1500	10	964879	88482	39935	64974	88482	22014
1500	15	1018628	81812	46243	64732	81812	18086
1500	20	1045808	78804	47228	64868	78804	17447
1500	25	1062337	80768	41517	64722	80768	16544

Fonte: o autor (2008).

Percebe-se através dos resultados obtidos que a redução não afetou a solução óti-

ma de nenhum dos problemas da amostra, pois todas as respostas obtidas com o espaço de estados reduzido e completo foram idênticas. Apesar das respostas idênticas entre o modelo reduzido e completo, não é possível garantir que esse procedimento de redução irá sempre manter a solução ótima no espaço de estado reduzido. Contudo, por trabalhar-se com a m melhores respostas em cada linha e coluna, é possível afirmar que as respostas próximas a ótima não ficam comprometidas com o procedimento. Percebe-se que os modelos reduzidos trabalham com aproximadamente 8% das variáveis existentes nos modelos sem redução do espaço de estados. Além da redução no número de variáveis foi possível perceber uma redução significativa no número de iterações ocorridas, onde o modelo reduzido necessitou de aproximadamente 57% menos iterações que o modelo com o espaço de estados completo. Cabe ressaltar que em termos de tempo essa relação é ainda maior, pois cada iteração no modelo reduzido é mais rápida que no modelo completo tendo em vista a diferença entre a quantidade de variáveis nos dois modelos. Apesar da tabela não mostrar uma comparação entre tempos, para problemas com 1500 viagens observou-se uma duração média de 2 minutos e 28 segundos para obtenção da solução ótima em problemas com espaço de estados completo comparados com uma média de apenas 21 segundos para os modelos reduzidos. Pelas razões expostas, é possível afirmar que o procedimento consegue atingir uma redução significativa do espaço de estados tendo em vista a aceleração na obtenção das soluções e tratabilidade dos problemas. Assim, pode-se concluir que o primeiro procedimento mostrou-se de grande relevância pois consegue manter um espaço de estados satisfatório reduzindo eficientemente a instância de problemas da ordem n x n para a ordem m x m, onde m é a raiz quadrada de n.

6.1.2 Definição das viagens iniciais e finais

O procedimento anterior buscou descartar as viagens expressas de maior custo que apresentam baixa contribuição para a solução do problema. Até o momento ignorou-se as viagens candidatas como viagens iniciais e finais de uma seqüência de viagens. O número de viagens desse tipo, naturalmente varia de acordo com a posição e número de gara-

gens no modelo MDVSP. A explicação para isso é bastante simples, com um número maior de garagens aumenta-se a quantidade de viagens próximas a essas garagens. Como viagens próximas apresentam custos de deslocamento menores que viagens distantes, então é natural que os modelos MDVSP façam uso dessa vantagem na minimização dos custos de deslocamento. Outra fato que pode ser observado, refere-se ao horário das viagens. Viagens com ocorrência mais cedo tem maior chance de tornarem-se viagens iniciais de uma seqüência de viagens, enquanto viagens mais tardias tem maior chance de tornarem-se viagens finais de uma seqüência de viagens.

Resumidamente, pode-se afirmar que um grande número de viagens não é candidata a iniciar ou finalizar uma seqüência de viagens seja pela distância geográfica da garagem ou pelo horário em que será executada. O segundo procedimento busca justamente identificar que viagens devem permanecer no modelo reduzindo ainda mais o espaço de estados. Nesse procedimento, a identificação das viagens candidatas à viagem inicial e final dá-se da seguinte maneira:

1) Transforma-se o MDVSP em diversos problemas VSP – um para cada garagem;

2) Soluciona-se cada um dos problemas VSP separadamente;

3) Registra-se a viagens iniciais e finais encontradas em cada solução VSP.

Ao analisar as respostas separadamente de cada VSP é possível identificar quais viagens foram escolhidas em cada modelo como as melhores candidatas considerando-se uma garagem específica. Deste modo é possível identificar para cada garagem separadamente quais viagens apresentam melhor horário e distância para serem consideradas alternativas de resposta como viagens iniciais e finais. O exemplo apresentado nessa tese tem duas garagens e quinze viagens. O Quadro 5 mostra as seqüências de viagens obtidas ao solucionar o modelo VSP correspondente a cada uma das garagens.

Solução VSP para a garagem 1	Solução VSP para a garagem 2
G1 - 3 - 12 - 13 - G1 G1 - 4 - G1 G1 - 5 - 9 - 11 - G1 G1 - 8 - 1 - 7 - G1 G1 - 14 - 6 - 2 - 10 - G1 G1 - 15 - G1	G2 - 3 - 6 - 2 - 10 - G2 G2 - 5 - G2 G2 - 8 - 1 - 7 - G2 G2 - 12 - 13 - G2 G2 - 14 - 4 - 9 - 11 - G2 G2 - 15 - G2
Viagens iniciais	**Viagens iniciais**
3, 4, 5, 8, 14 e 15	3, 5, 8, 12, 14 e 15
Viagens finais	**Viagens finais**
13, 4, 11, 7, 10, 15	10, 5, 7, 13, 11, 15

Quadro 5 – Viagens iniciais e finais selecionadas
Fonte: o autor (2008).

Através do exemplo é possível notar que o posicionamento das garagens impacta na chance que uma viagem tem de tornar-se inicial ou final dentro de uma seqüência de viagens. De acordo com o segundo procedimento as viagens 1, 2, 6 e 9 não são consideradas viagens candidatas a iniciar ou finalizar uma seqüência de viagens.

Para execução desse procedimento de redução foram necessárias a implementação de dois comandos dentro do módulo de redução (ver Figura 6). O primeiro comando “Gerar modelo linear VSP” gera os modelos VSPs necessários ao funcionamento do procedimento e salva-os em arquivos distintos, conforme mostra a Figura 11. Tais arquivos são chamados e solucionados pelo software LINGO, que por sua vez gera um número igual de arquivos contendo as respostas dos modelos. Os Apêndices G e H apresentam os modelos VSP gerados para cada uma das garagens. O Apêndice F apresenta o código referente a esse comando.

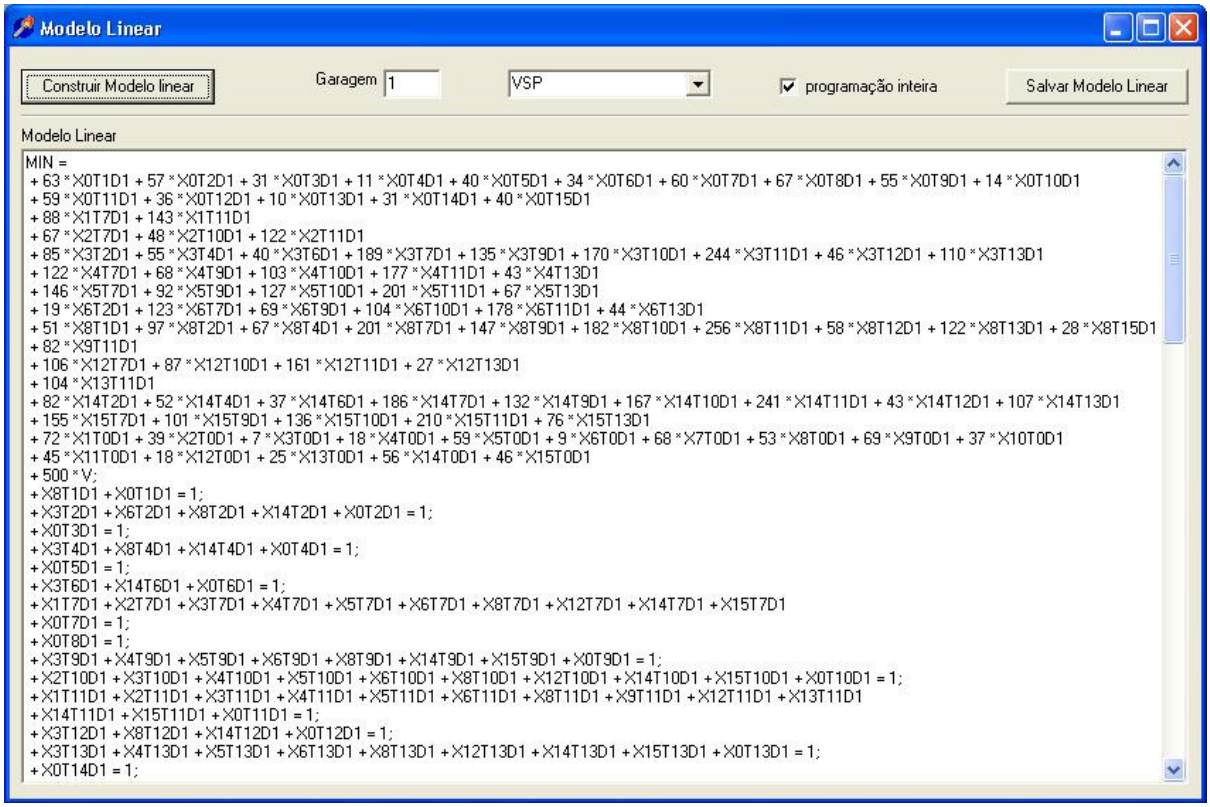


Figura 11 – Interface do modelo linear VSP
Fonte: o autor (2008).

O comando “Reduzir espaço de estados – proc 2” chama o interpretador de arquivos de respostas. O interpretador lê os arquivos de resposta gerados pelo LINGO, um para cada modelo VSP, e constrói as seqüências de viagens identificando suas viagens iniciais e finais. Ao final do procedimento, é gerado um arquivo único contendo a lista de viagens iniciais e finais de todos os modelos VSP. A Figura 12 mostra a implementação do comando “Reduzir espaço de estados – proc 2”. O Apêndice D apresenta o código referente a esse comando.

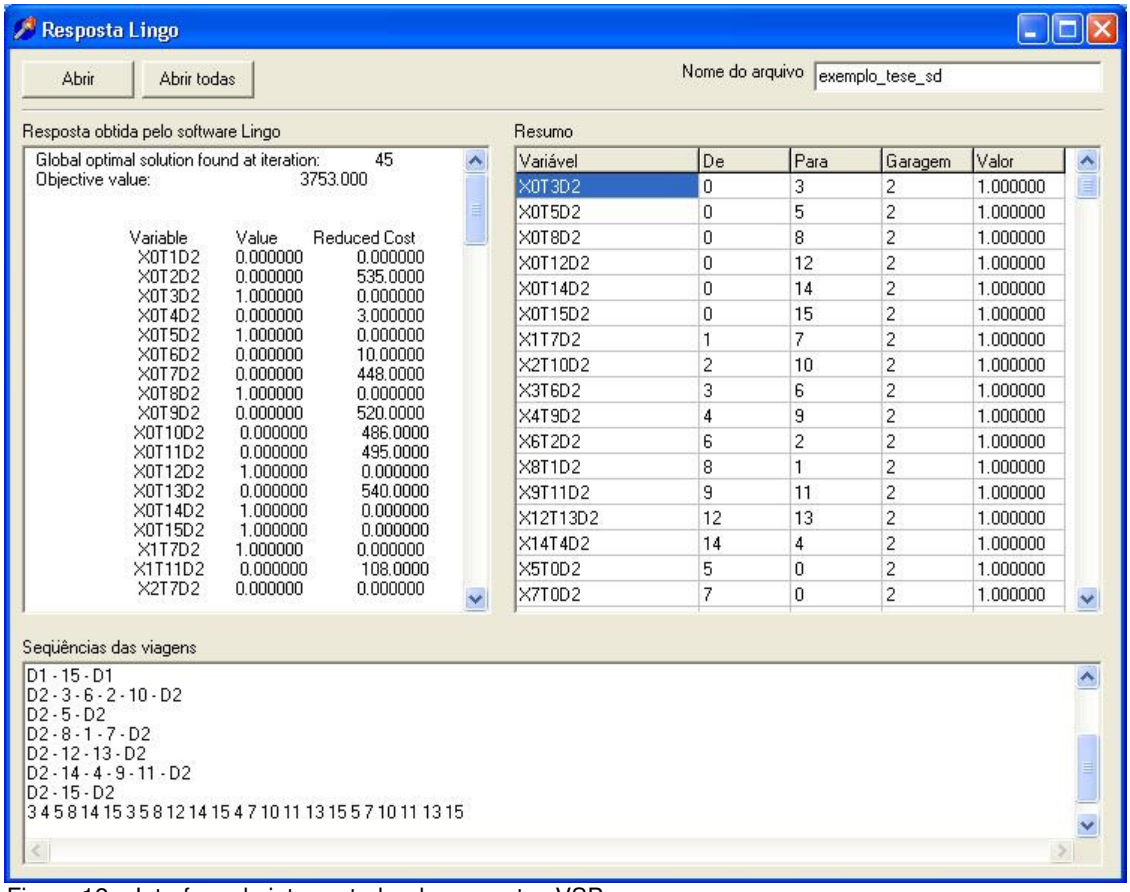


Figura 12 – Interface do interpretador de respostas VSP
Fonte: o autor (2008).

Resumidamente o segundo procedimento de redução do espaço de estados consiste na execução individual de g modelos VSP, onde g indica o número de garagens no modelo MDVSP. Como citamos anteriormente os modelos VSP são de complexidade polinomial $O(n^2 \log n)$, de modo que esse procedimento não compromete computacionalmente a heurística de solução para o MDVSP. Vale salientar que nenhuma comparação de eficiência foi apresentada nessa sessão, tendo em vista que a eficiência desse procedimento é testada na seção 6.2 conjuntamente com o terceiro procedimento.

6.1.3 Exclusão de variáveis nunca selecionadas

O terceiro e último procedimento de redução do espaço de estados também faz uso dos modelos VSPs individualizados. O objetivo desse procedimento é identificar variáveis (que representam deslocamento entre viagens) que nunca foram selecionadas nos modelos VSP. Entende-se que se uma variável não é escolhida como possível resposta para os

modelos VSP individuais, então ela não será uma candidata como resposta ao modelo MDVSP. Com base nesse pressuposto, o terceiro procedimento retira tais variáveis do espaço de estados.

Para implementação desse procedimento foi elaborado um novo interpretador das respostas obtidas pelo Software LINGO para cada um dos modelos VSP. O interpretador contabiliza e compara as respostas obtidas por esses modelos, conforme mostra o algoritmo a seguir.

For $i \leftarrow 1$ to n do	{para cada linha faça}
For $j \leftarrow 1$ to n do	{para cada coluna faça}
For $k \leftarrow 1$ to g do	{olha os valores em todos as modelos}
If $X(i,j,k) = 1$ then $V(i,j) \leftarrow V(i,j)+1$	{se o valor da célula for igual a um então incrementa-se o $V_{i,j}$}

Figura 13 – Algoritmo de contabilização de respostas
Fonte: o autor (2008).

Onde:

g : é o número de garagens;
n : é o número de viagens;
X: é a matriz (n x n x g) que contém os resultados dos modelos VSP;
V: é a matriz (n x n) que guarda o número de vezes que cada combinação foi escolhida;
i, j, k: são indexadores da matriz X e V.

A grade apresentada na Figura 14 contabiliza quantas vezes cada variável foi escolhida, onde o número dentro de cada célula indica quantas vezes a variável foi escolhida como resposta. Como nesse procedimento analisam-se as respostas obtidas pelos modelos VSPs (uma para cada garagem), então o maior valor possível em cada célula é igual a g, onde g é o número de garagens. Percebe-se que após contabilizar as respostas algumas variáveis nunca foram escolhidas (células com valor zero) por nenhum dos modelos, enquanto outras foram escolhidas por todos os modelos (células com valores igual a 2).

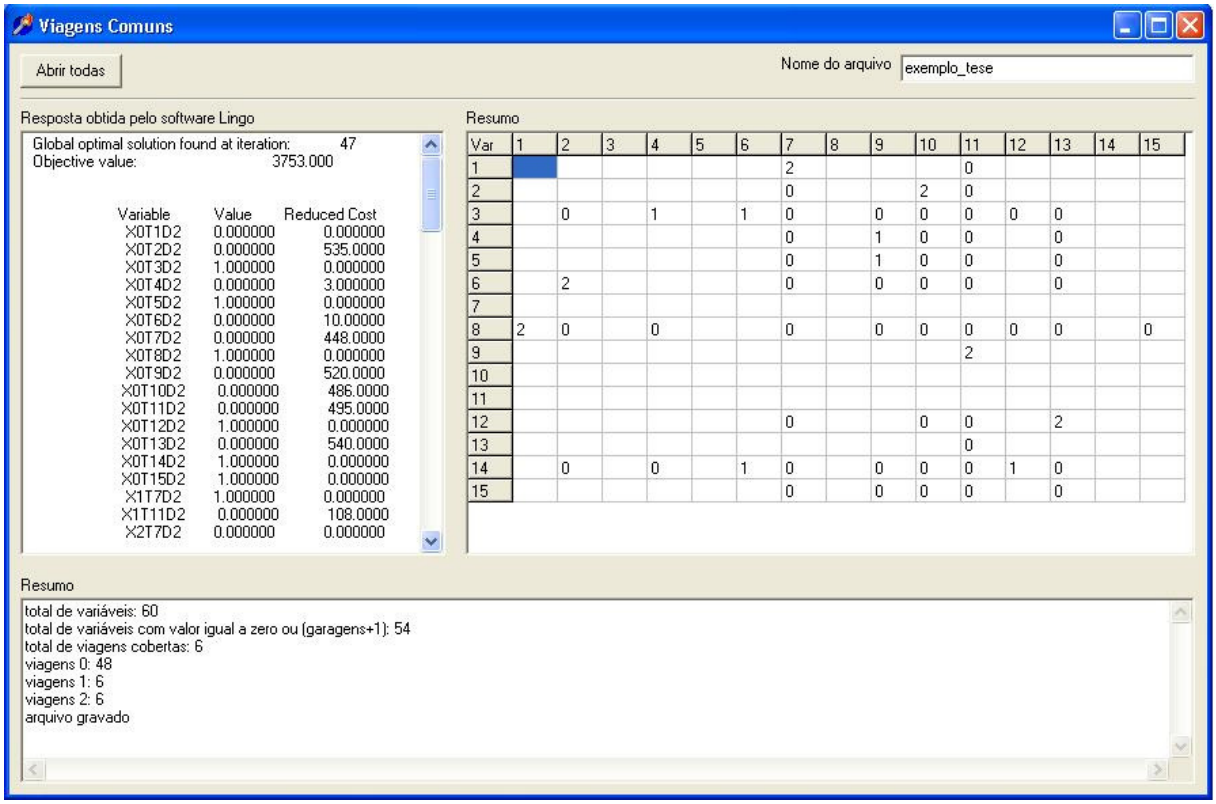
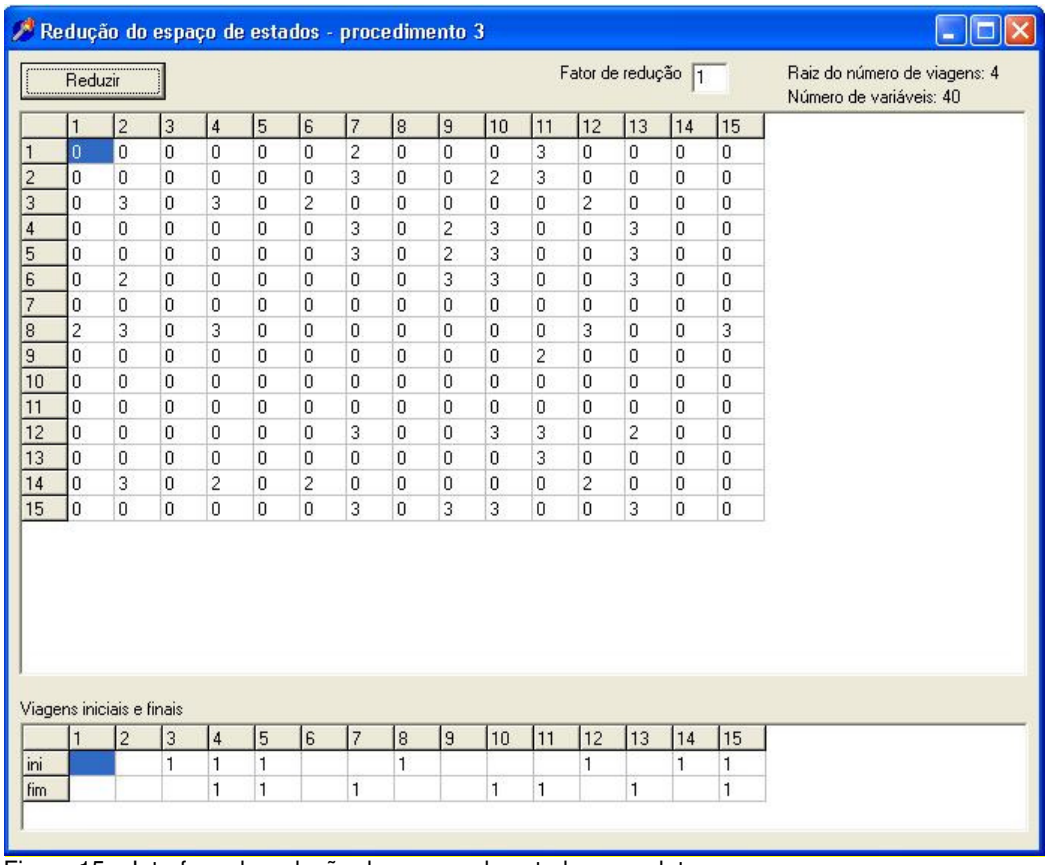


Figura 14 – Interface do contador de respostas
Fonte: o autor (2008).

Esse procedimento gera um arquivo que armazena todas as variáveis que nunca foram selecionadas por nenhum dos modelos. Subseqüentemente essas variáveis são descartadas do espaço de estados conforme mostra a Figura 15. O apêndice E apresenta o código implementado para o terceiro procedimento.

.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	2	0	0	0	3	0	0	0	0
2	0	0	0	0	0	0	3	0	0	2	3	0	0	0	0
3	0	3	0	3	0	2	0	0	0	0	0	2	0	0	0
4	0	0	0	0	0	0	3	0	2	3	0	0	3	0	0
5	0	0	0	0	0	0	3	0	2	3	0	0	3	0	0
6	0	2	0	0	0	0	0	0	3	3	0	0	3	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	2	3	0	3	0	0	0	0	0	0	0	3	0	0	3
9	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	3	0	0	3	3	0	2	0	0
13	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0
14	0	3	0	2	0	2	0	0	0	0	0	2	0	0	0
15	0	0	0	0	0	0	3	0	3	3	0	0	3	0	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ini			1	1	1			1				1		1	1
fim				1	1		1			1	1		1		1

Figura 15 – Interface da redução do espaço de estados completa
Fonte: o autor (2008).

Por definição, na Figura 15 os valores iguais a dois indicam que a célula está entre as m menores tanto para a linha como para a coluna, o valor um indica que a célula está entre as m menores para a linha ou para a coluna, os valores iguais a zero indicam que a célula não pertence ao espaço de estados reduzido, por fim, os valores iguais a três referem-se ao terceiro procedimento de redução do espaço de estados, ou seja, indicam que a célula jamais foi escolhida por nenhum dos modelos VSP como pertencente à solução ótima. Sendo assim, considerando os três procedimentos de redução do espaço de estados, deve-se levar em conta os valores 1 e 2 na matriz, e considerando apenas os dois primeiros procedimentos de redução, deve-se levar em conta os valores 1, 2 e 3 na matriz. No Apêndice I encontra-se todo o código implementado para desenvolvimento do módulo que executa a redução completa do espaço de estados.

6.2 COMPARAÇÃO DOS RESULTADOS

Essa seção busca comparar as respostas obtidas pelos modelos MDVSP com e sem redução do espaço de estados. Para tanto, devida a complexidade exponencial desses problemas, não foram utilizadas instâncias superiores a 200 viagens e 4 garagens. O algoritmo *Branch-and-Bound* foi utilizado para localizar a solução ótima dos 9 modelos gerados aleatoriamente conforme mostram os resultados da Tabela 3.

Tabela 3 – Comparação dos resultados após redução do espaço estados

amostra				solução sem redução				solução com reduções (procedimentos 1 e 2)				solução com reduções (procedimentos 1, 2 e 3)			
Número de viagens	Número de garagens	Janela de tempo	Número de veículos	Número de variáveis	Tempo (min)	Solução ótima	Número de iterações	Número de variáveis	Tempo requerido	Solução ótima	Número de iterações	Número de variáveis	Tempo requerido	Solução ótima	Número de iterações
200	2	5	32	29.735	2m 23s	22.782	116.659	5.619	0m 6s	22.782	7.201	663	0m 1s	22.862	7.539
200	2	10	20	35.170	4m 15s	19.748	192.821	6.057	0m 7s	19.748	12.129	667	0m 3s	19.748	29.028
200	2	15	17	36.744	7m 58s	19.261	224.316	6.253	0m 8s	19.261	10.297	609	0m 2s	19.284	9.631
200	3	5	32	44.811	2m 25s	22.418	84.707	8.761	0m 9s	22.419	5.279	1.240	0m 4s	22.473	32.450
200	3	10	23	52.437	9m 43s	21.829	302.688	9.427	0m 12s	21.829	18.930	1.240	0m 1s	21.844	12.135
200	3	15	16	54.879	20m 33s	18.774	419.416	9.202	4m 59s	18.774	230.531	1.207	0m 12s	18.774	56.085
200	4	5	32	60.192	3m 3s	22.567	79.442	11.681	0m 13s	22.567	5.587	1.861	0m 1s	22.577	2.441
200	4	10	20	69.960	25m 30s	19.208	466.324	12.285	7m 27s	19.208	257.426	1.781	0m 22s	19.209	94.460
200	4	15	17	73.512	31m 34s	21.754	622.578	12.408	10m 59s	21.754	397.067	2.401	0m 53s	21.754	84.431

Fonte: o autor (2008).

A Tabela 3 apresenta as soluções encontradas considerando três situações distintas: modelo sem redução do espaço de estados, modelos com redução utilizando os procedimentos 1 e 2, e modelo com redução utilizando os procedimentos 1, 2 e 3. Ao analisar os dados, percebe-se que a aplicação dos três procedimentos de redução do espaço de estados diminui significativamente o tempo necessário para encontrar a solução dos problemas. Problemas que demandam aproximadamente trinta minutos para serem solucionados com o espaço de estados completo chegam a apresentar tempos de solução inferiores a 1 minuto quando aplicados os procedimentos de redução. Essa significativa queda no tempo de solução deve-se a redução no número de variáveis com o qual os modelos trabalham. Em média os procedimentos 1 e 2 reduziram o número de variáveis em 82% quando comparados aos modelos com espaço de estados completo, enquanto os procedimentos 1, 2 e 3 reduziram em média 98% das variáveis. A redução no número de variáveis tem efeito direto na diminuição do número de iterações, que apresentou redução média de 87% quando comparados os modelos sem redução e com redução utilizando os procedimentos 1, 2 e 3. O objetivo de uma redução do espaço de estados é manter no espaço reduzido as soluções

ótima ou satisfatória originalmente encontradas no espaço não reduzido. Ao verificar as respostas obtidas na Tabela 3 percebe-se que, quando aplicados os procedimentos de redução 1 e 2, na maioria das vezes as respostas são idênticas às encontradas nos modelos sem redução. Nos casos em que ocorre aumento nos custos ao se aplicarem os procedimentos 1, 2 e 3, verifica-se que em média esse aumento foi de 0,3%. Para evitar distorções nesse percentual, descontou-se das soluções ótimas os custos por uso de veículo. Por exemplo, no segundo item de nossa amostra foi encontrada a solução ótima igual a 19.748 com o uso de 20 veículos. O custo considerando apenas o deslocamento dos veículos foi de 9.748 tendo em vista que cada veículo utilizado contribui com um custo igual a 500 para a solução. Como o percentual de aumento nos custos não apresenta distorções pelo uso de veículos, pode-se afirmar que, apesar da redução de aproximadamente 98% das variáveis, as soluções ótimas ou satisfatórias foram mantidas nos modelos ao aplicar-se os procedimentos de redução do espaço de estados apresentados nesse capítulo, comprovando-se a eficiência dessas reduções.

6.3 SEGUNDA ETAPA - ALGORITMO DE SOLUÇÃO MDVSP

As seções anteriores tiveram como foco principal a redução do espaço de estados. A seção a seguir trata do desenvolvimento de um procedimento capaz de encontrar a solução satisfatória para o problema MDVSP. Como é sabido o MDVSP é um problema NP difícil e por essa razão qualquer forma de encontrar a solução ótima demanda tempo e memória computacional em escala exponencial. Exemplo disso pode ser observado em publicações especializadas, onde as maiores amostras encontradas não superam instâncias maiores que 1500 viagens ou oito garagens. A heurística que é apresentada faz uso do modelo MDVSP com flexibilidade (doravante chamado MDVSPcf).

O MDVSPcf pode ser entendido como um MDVSP que permite que os veículos re-

tornem para garagens distintas daquelas das quais partiram. Com a permissão de retorno dos veículos para garagens distintas das quais partiram é possível eliminar o indexador de garagem da modelagem reduzindo a complexidade dos modelos. Por exemplo, em algumas modelagens, conforme mostrado no referencial teórico, essa flexibilidade permite que o MDVSPcf, assim como o VSP, seja tratado como um problema de designação e desta forma, também possa ser solucionado pelo algoritmo húngaro. Isso ocorre nos modelos onde as respostas para o MDVSP são representadas por caminhos pré-hamiltonianos onde uma das variáveis que compõe o caminho pré-hamiltoniano armazena os custos de deslocamentos até as garagens bem como os custos por uso dos veículos. No modelo MDVSPcf aqui desenvolvido não se fez uso dos caminhos pré-hamiltonianos, mas optou-se pela exclusão do indexador referente à garagem, de modo a manter o modelo o mais próximo possível da modelagem convencional para o problema MDVSP. O MDVSPcf utilizado nessa pesquisa é apresentado a seguir:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}x_{ij0} + \sum_{k=1}^{g}\sum_{i=1}^{n} D_{ik}x_{i0k} + \sum_{k=1}^{g}\sum_{j=1}^{n} E_{jk}x_{0jk} \qquad \forall i,j : M_{ij} \neq 0 \qquad (1)$$

$$\sum_{i=1}^{n} x_{ij0} + \sum_{k=1}^{g} x_{0jk} = 1 \qquad \forall j : M_{ij} \neq 0 \qquad (2)$$

$$\sum_{j=1}^{n} x_{ij0} + \sum_{k=1}^{g} x_{i0k} = 1 \qquad \forall i : M_{ij} \neq 0 \qquad (3)$$

$$\sum_{i=1}^{n} x_{i0k} = \sum_{j=1}^{n} x_{0jk} \qquad \forall k \neq 0 \qquad (4)$$

$$x_{ij0}\{0,1\} \qquad \forall i,j : M_{ij} \neq 0 \qquad (5)$$

Onde:

g: é o número de garagens;
n: é o número de viagens;
x: é a variável de decisão binária;
C: é a matriz de custos entre viagens;
D: é a matriz de custos entre a garagem e a viagem inicia de uma seqüência de viagens;
E: é a matriz de custos entre a garagem e a viagem final de uma seqüência de viagens;
M: é a matriz de folga total n x n;

i: é o indexador da viagem de origem;
j: é o indexador da viagem de destino;
k: é o indexador da garagem.

A função objetivo (1) busca minimização do custo total que é composta por três partes: os custos de deslocamento entre viagens, entre as viagens iniciais e garagens e entre as viagens finais e garagens. A restrição (2) obriga que um deslocamento ocorra até a viagem j, seja originário de uma garagem ou de outra viagem. A restrição (3) obriga que um deslocamento saia da viagem i, seja o destino uma garagem ou outra viagem. A restrição (4) força com que o número de veículos que partem de uma determinada garagem seja idêntico ao número de veículos que retornem a essa mesma garagem, ainda que as garagens de saída e chegada sejam distintas. Apesar do indexador k representar as garagens na modelagem, é possível perceber que quando os indexadores i e j são diferentes de zero , então k assume o valor zero anulando em termos práticos o indexador no que se refere aos deslocamentos entre viagens. O apêndice J apresenta o modelo MDVSPcf para o exemplo que está sendo utilizado nessa tese.

Sabe-se que a resposta ótima de um modelo MDVSP será sempre um valor igual ou menor que a solução ótima do melhor VSP e igual ou maior que a solução ótima do MDVSPcf. Para entender essa afirmação é preciso compreender que o espaço de estados de cada VSP está contido no espaço de estados do MDSVP e esse, por sua vez, está contido no espaço de estados do MDVSPcf (Figura 16).

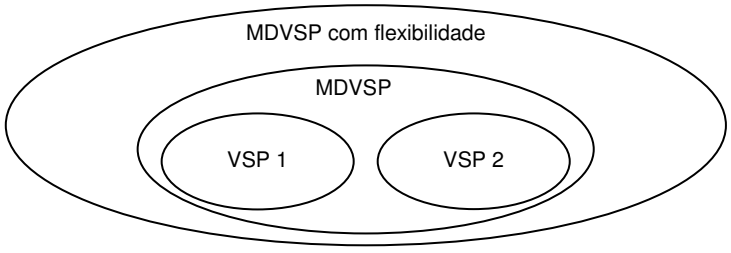


Figura 16 – Intersecção dos espaços de estados MDVSPcf, MDVSP e VSP
Fonte: o autor (2008).

No modelo MDVSPcf adotado nessa tese todos os custos de deslocamentos entre as viagens e garagens são registrados na função objetivo. O registro de todos os custos de

deslocamento até as garagens não permite que o modelo seja resolvido através do algoritmo húngaro, tendo em vista que não é possível adaptar o modelo a uma única matriz. Entretanto a inclusão de todos os possíveis custos de deslocamento no problema, permite que gradativamente encontre-se soluções para o MDVSPcf e se utilize essa informação para definir um limite inferior (*lower bound*) de solução para o MDVSP, pois, conforme constatou-se, a solução MDVSP é sempre maior ou igual à solução MDVSPcf.

O procedimento que descreve-se nessa seção faz uso dessa propriedade para encontrar a solução do MDVSP, em outras palavras, o modelo MDVSPcf é solucionado diversas vezes de modo que iterativamente flexibiliza-se e converge-se as seqüências de viagens encontradas. Flexibilizar significa permitir que um veículo termine suas viagens em uma garagem distinta da inicial, enquanto convergir significa forçar os veículos a retornarem para as mesmas garagens das quais partiram através da adição de restrições no modelo que impeçam uma resposta inadequada ao MDVSP. Sendo assim, esse procedimento consiste na solução progressiva do MDVSPcf, onde aceita-se uma resposta inadequada, mas penaliza-se essa resposta para que uma nova solução seja encontrada. Esse processo executado repetidamente conduz a uma resposta onde os veículos retornam a mesma garagem de onde partiram. Um novo módulo computacional foi desenvolvido para esse procedimento, conforme mostra a Figura 17.

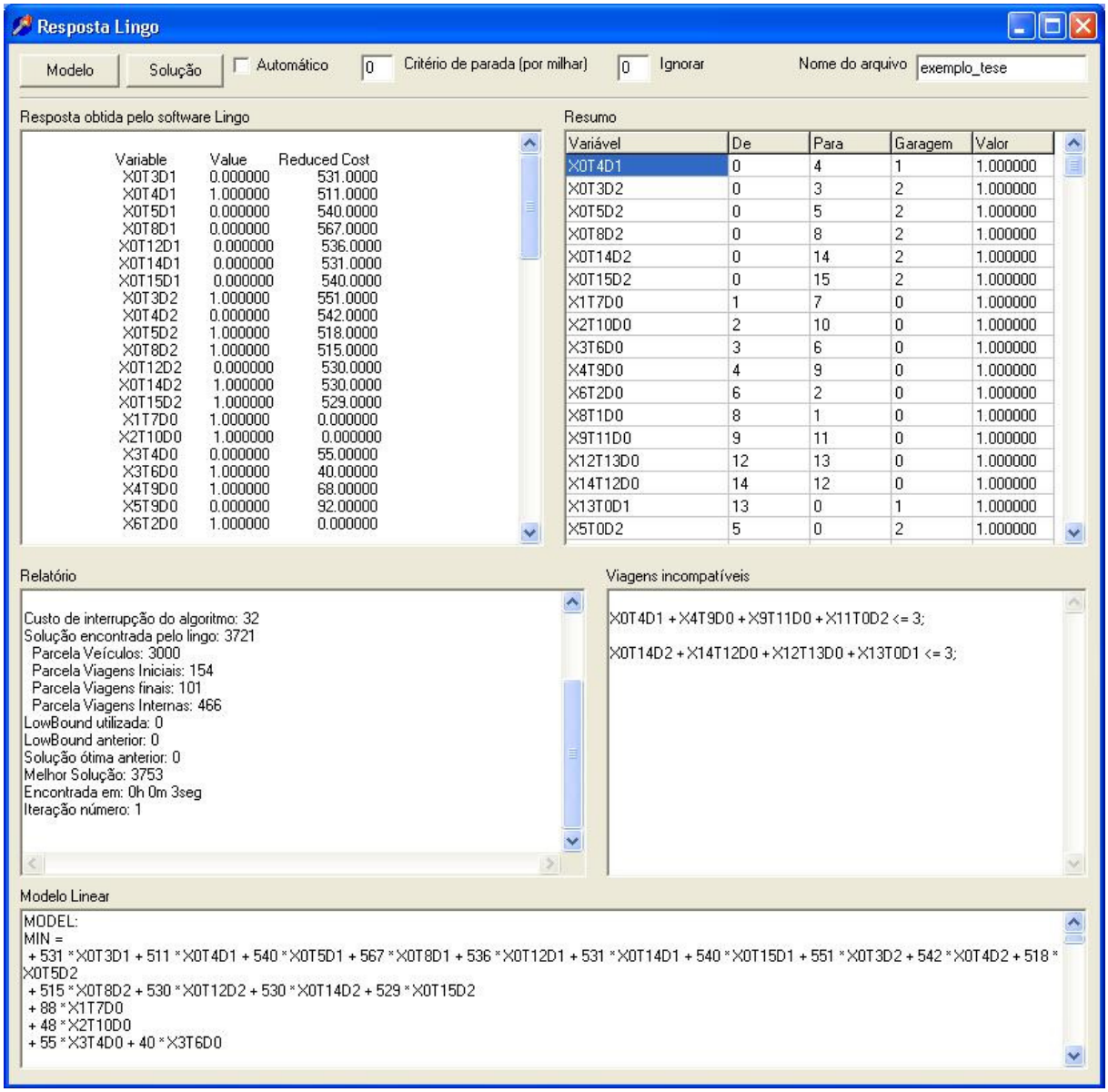


Figura 17 – Interface do módulo de solução MDVSP
Fonte: o autor (2008).

Ao solucionar o modelo MDVSPcf é natural que a flexibilidade de chegada e saída de veículos de garagens distintas gere respostas inadequadas ao modelo MDVSP. Posto isso, é imprescindível que tais soluções sejam restringidas para que o resultado final do procedimento conduza a uma resposta adequada ao MDVSP. Em outras palavras torna-se necessário acrescentar ao modelo linear as restrições de proibição de soluções inadequadas.

Sabe-se que a heurística apresentada trabalha sobre o espaço de estados reduzido e utiliza o modelo MDVSPcf como base para solucionar o MDVSP. Posto isso a resposta

obtida pelo MDVSPcf será sempre o resultado da primeira iteração de nosso procedimento de descoberta da resposta para o MDVSP. Ao analisar a resposta obtida pelo MDVSPcf conforme mostra o Quadro 6, percebe-se que duas viagens são incoerentes para uma resposta considerável adequada ao MDVSP. Sendo assim, o segundo passo desse procedimento é interpretar a resposta obtida pelo MDVSPcf de modo a registrar as viagens inadequadas e acrescentar ao modelo linear as restrições que impeçam que essas seqüências de viagens sejam aceitas na próxima iteração do algoritmo.

Viagem	Número
G2 - 5 - G2	1
G2 - 15 - G2	2
G2 - 8 - 1 - 7 - G2	3
G2 - 14 - 6 - 2 - 10 - G2	4
G2 - 3 - 12 - 13 - G1	5
G1 - 4 - 9 - 11 - G2	6

Quadro 6 – Resposta da primeira iteração do MDVSPcf
Fonte: o autor (2008).

De acordo com exemplo as seqüências 5 e 6 foram consideradas inválidas na primeira iteração do algoritmo: G2 – 3 – 12 – 13 – G1 e G1 – 4 – 9 – 11 – G2. Os deslocamentos “de” e “para” as garagens são representados pelas variáveis $X_{0,3,2}$, $X_{13,0,1}$, $X_{0,4,1}$ e $X_{11,0,2}$, onde o primeiro indexador representa a viagem de origem, o segundo a viagem de destino e o terceiro a garagem. Quando o primeiro ou segundo indicador apresenta valor zero então o ponto de origem e destino refere-se à garagem. Tendo em vista que não é possível permitir que os veículos tenham como origem e destino garagens distintas, então as seguintes restrições devem ser adicionadas ao modelo linear:

$$X_{0,3,2} + X_{3,12,0} + X_{12,13,0} + X_{13,0,1} \leq 3$$

$$X_{0,4,1} + X_{4,9,0} + X_{9,11,0} + X_{11,0,2} \leq 3$$

É possível perceber que estas restrições obrigam que na próxima iteração do algoritmo outra seqüência de viagem seja encontrada ou, no caso de manterem-se as mesmas seqüências, que os veículos retornem para as mesmas garagens das quais partiram. Esse procedimento força a convergência para uma solução adequada para o MDVSP. Contudo

para problemas com muitas viagens o número de restrições acrescentadas ao modelo acaba tornando-se elevado reduzindo sua eficiência. Para corrigir esse problema outro procedimento foi adotado.

Após inserção das restrições de convergência a resposta MDVSP o modelo MDVSPcf é executado novamente. Naturalmente duas possibilidades de respostas podem ser obtidas nesse segundo momento: 1) a solução ótima permanece a mesma, ainda que as seqüências de viagens alterem-se devido à inserção das restrições de convergência ao MDVSP; 2) a solução ótima encontrada é maior que a anterior.

Quando o primeira possibilidade ocorre, então retorna-se ao procedimento de inserção das restrições de convergência a resposta MDVSP até que uma nova solução seja encontrada. Contudo quando a segunda possibilidade ocorre então adota-se os seguintes passos:

1) elimina-se do modelo linear todas as restrições de convergência a resposta MDVSP adicionadas ao modelo até o momento;

2) Adiciona-se ao modelo Restrição de Valor Mínimo para a Resposta MDVSP (*lower bound*).

O objetivo do passo 1 ao eliminar todas as restrições de convergência é a de acelerar a obtenção de uma resposta pelo modelo linear tendo em vista que a acúmulo dessas restrições compromete o desempenho do algoritmo. Contudo a eliminação dessas restrições sem qualquer outro artifício faria com que o modelo voltasse a seu estado inicial e, conseqüentemente, a mesma solução inicial seria sempre encontrada. O artifício desenvolvido foi acrescentar ao modelo uma nova restrição que garantisse que a resposta obtida não retornasse ao estágio inicial. A essa restrição deu-se o nome de “Valor Mínimo para a Resposta MDVSP”. Como apresentado anteriormente, as respostas obtidas por modelos MDVSPcf são sempre inferiores ou iguais àquelas obtidas pelo MDVSP. Sendo assim atribui-se a última resposta obtida pela solução MDVSPcf como restrição de valor mínimo para a res-

posta MDVSP a cada iteração do algoritmo. O modelo a seguir mostra o MDVSPcf acrescido das duas restrições adicionais ("convergência" e "valor mínimo para a resposta MDVSP") explicadas.

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}x_{ij0} + \sum_{k=1}^{g}\sum_{i=1}^{n} D_{ik}x_{i0k} + \sum_{k=1}^{g}\sum_{j=1}^{n} E_{jk}x_{0jk} \qquad \forall i,j : R_{ij} \neq 0 \qquad (1)$$

$$\sum_{i=1}^{n} x_{ij0} + \sum_{k=1}^{g} x_{0jk} = 1 \qquad \forall j : R_{ij} \neq 0 \qquad (2)$$

$$\sum_{j=1}^{n} x_{ij0} + \sum_{k=1}^{g} x_{i0k} = 1 \qquad \forall i : R_{ij} \neq 0 \qquad (3)$$

$$\sum_{i=1}^{n} x_{i0k} = \sum_{j=1}^{n} x_{0jk} \qquad \forall k \neq 0 \qquad (4)$$

$$x_{ij0}\{0,1\} \qquad \forall i,j : R_{ij} \neq 0 \qquad (5)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}x_{ij0} + \sum_{k=1}^{g}\sum_{i=1}^{n} D_{ik}x_{i0k} + \sum_{k=1}^{g}\sum_{j=1}^{n} E_{jk}x_{0jk} \geq b \qquad \forall i,j : R_{ij} \neq 0 \qquad (6)$$

$$x_{0jk} + \sum_{i,j \in S} x_{ij0} + x_{i0l} \leq |S| + 1 \qquad \forall S : R_{ij} \neq 0, k \neq l \qquad (7)$$

Onde:

g: é o número de garagens;
n: é o número de viagens;
x: é a variável de decisão binária;
C: é a matriz de custos entre viagens;
D: é a matriz de custos entre a garagem e a viagem inicial;
E: é a matriz de custos entre a garagem e a viagem final;
R: é a matriz de folga total após os procedimentos de redução, onde 0 indica que a variável foi excluída do espaço de estados e 1 caso contrário;
i: é o indexador da viagem de origem;
j: é o indexador da viagem de destino;
k,l: são os indexadores da garagem;
b: é o valor da última solução MDVSPcf encontrada (*lower bound*);
S: considerando que o MDVSPcf é um grafo G=(N,A), então S é um subgrafo de G, em que |S| representa o número de deslocamentos entre viagens desse subgrafo.

A restrição (6) refere-se ao "Valor Mínimo para a Resposta MDVSP", enquanto a restrição (7) refere-se à convergência à resposta MDVSP. Aqui é necessário ressaltar que o modelo desenvolvido não tem por finalidade encontrar a solução MDVSP diretamente tendo

em vista a complexidade exponencial desses problemas. Mas tentar descobrir uma solução para o MDVSP através de formulações matemáticas menos complexas. Em outras palavras mesmo que a solução encontrada não seja adequada ao modelo MDVSP é possível calcular os custos de correção da solução.

A vantagem ao se calcular o custo de correção da resposta MDSVP é poder utilizar o valor encontrado como critério de parada da heurística. Através da correção dos custos é possível descobrir o quanto uma resposta adequada ao problema MDVSP distancia-se da resposta obtida pelo MDVSPcf. Em outras palavras pode-se dizer que esse procedimento também calcula o custo de interrupção da heurística na iteração na qual ela se encontra. A Tabela 4 mostra os custos de correção para a resposta obtida no exemplo tratado nessa pesquisa:

Tabela 4 – Custo de Interrupção do Algoritmo

Seqüência de viagem: G2 – 3 – 12 – 13 – G1				Seqüência de viagem: G1 – 4 – 9 – 11 – G2			
Custo	início	fim	Total	Custo	início	fim	Total
Atual	$X_{0,3,2}$ 51	$X_{13,0,1}$ 25	76	Atual	$X_{0,4,1}$ 11	$X_{11,0,2}$ 7	18
Corrigido pela garagem1	$X_{0,3,1}$ 31	$X_{13,0,1}$ 25	56	Corrigido pela garagem1	$X_{0,4,1}$ 11	$X_{11,0,1}$ 45	56
Corrigido pela garagem2	$X_{0,3,2}$ 51	$X_{13,0,2}$ 29	80	Corrigido pela garagem2	$X_{0,4,2}$ 42	$X_{11,0,2}$ 7	49

Fonte: o autor (2008).

Os campos em verde mostram os custos atribuídos às seqüências inadequadas encontrados pela solução do MDVSPcf onde o valor total é igual a 94 (76+18). Os campos em amarelo mostram os custos de correção das seqüências, onde a primeira seqüencia deve ser corrigida pela garagem 1 (56≤80) e a segunda seqüência deve ser corrigida pela garagem 2 (49≤56) totalizando um custo de correção igual a 105 (56+49). Sendo assim calcula-se o custo de interrupção do algoritmo pela diferença entre o custo de correção e o custo atual (105-94=11). Em problemas de alta complexidade, onde inúmeras iterações são necessárias, esse recurso torna-se fundamental para uma adequada tomada de decisão, uma vez que atribui ao modelo um critério de parada capaz de informar com precisão o custo de interrupção do algoritmo.

No exemplo trabalhado ao longo dessa tese, foram necessárias cinco iterações da heurística para atingirmos uma solução ótima ao modelo MDVSP. O Quadro 7 mostra passo a passo as iterações ocorridas nesse procedimento para solucionar o MDVSP.

Iteração número: 1	Iteração número: 2	Iteração número: 3	Iteração número: 4	Iteração número: 5
Seqüências encontradas: D1 - 4 - 9 - 11 - D2 D2 - 3 - 6 - 2 - 10 - D2 D2 - 5 - D2 D2 - 8 - 1 - 7 - D2 D2 - 14 - 12 - 13 - D1 D2 - 15 - D2	Seqüências encontradas: D1 - 3 - 4 - 9 - 11 - D2 D2 - 5 - D2 D2 - 8 - 1 - 7 - D2 D2 - 12 - 13 - D1 D2 - 14 - 6 - 2 - 10 - D2 D2 - 15 - D2	Seqüências encontradas: D1 - 3 - 6 - 2 - 10 - D2 D2 - 4 - 9 - 11 - D2 D2 - 5 - D2 D2 - 8 - 1 - 7 - D2 D2 - 14 - 12 - 13 - D1 D2 - 15 - D2	Seqüências encontradas: D1 - 4 - 9 - 11 - D2 D1 - 14 - 12 - 13 - D1 D2 - 3 - 6 - 2 - 10 - D1 D2 - 5 - D2 D2 - 8 - 1 - 7 - D2 D2 - 15 - D2	Seqüências encontradas: D1 - 4 - D1 D1 - 14 - 12 - 13 - D1 D2 - 3 - 6 - 2 - 10 - D2 D2 - 5 - 9 - 11 - D2 D2 - 8 - 1 - 7 - D2 D2 - 15 - D2
Custo de interrupção: 32 Solução MDVSPcf: 3721 LowerBound utilizada: 0 Melhor solução até o momento: 3753 Tempo gasto: 0h 0m 2seg	Custo de interrupção: 24 Solução MDVSPcf: 3729 LowerBound utilizada: 3721 Melhor solução até o momento: 3753 Tempo gasto: 0h 0m 1seg	Custo de interrupção: 21 Solução MDVSPcf: 3732 LowerBound utilizada: 3729 Melhor solução até o momento: 3753 Tempo gasto: 0h 0m 1seg	Custo de interrupção: 10 Solução MDVSPcf: 3743 LowerBound utilizada: 3732 Melhor solução até o momento: 3753 Tempo gasto: 0h 0m 1seg	Custo de interrupção: 0 Solução MDVSPcf: 3744 LowerBound utilizada: 3743 Melhor solução até o momento: 3744 Tempo gasto: 0h 0m 1seg
Restrições de convergência: $X_{0,4,1}+X_{4,9,0}+X_{9,11,0}+X_{11,0,2}\leq 3$ $X_{0,14,2}+X_{14,12,0}+X_{12,13,0}+X_{13,0,1}\leq 3$	Restrições de convergência: $X_{0,3,1}+X_{3,4,0}+X_{4,9,0}+X_{9,11,0}+X_{11,0,2}\leq 4$ $X_{0,12,2}+X_{12,13,0}+X_{13,0,1}\leq 2$	Restrições de convergência: $X_{0,3,1}+X_{3,6,0}+X_{6,2,0}+X_{2,10,0}+X_{10,0,2}\leq 4$ $X_{0,14,2}+X_{14,12,0}+X_{12,13,0}+X_{13,0,1}\leq 3$	Restrições de convergência: $X_{0,4,1}+X_{4,9,0}+X_{9,11,0}+X_{11,0,2}\leq 3$ $X_{0,3,2}+X_{3,6,0}+X_{6,2,0}+X_{2,10,0}+X_{10,0,1}\leq 4$	Fim do algoritmo, solução MDVSP encontrada Tempo total gasto: 0h 0m 6seg

Quadro 7 – Solução do MDVSP a cada iteração.
Fonte: o autor (2008).

As iterações no Quadro 7 ilustram as restrições de convergência e de valor mínimo para a resposta MDVSP, bem como os custos para correção das viagens inadequadas. Esses procedimentos conforme explicado, forçam a heurística a buscar novas soluções, seja pela construção de novas seqüências de viagens ou pela escolha de novas garagens. A implementação computacional desse procedimento encontra-se no apêndice L. Uma vez definidos todos os procedimentos da heurística, tanto no que diz respeito à redução do espaço de estados como na solução propriamente dita do MDVSP prossegue-se na próxima etapa, onde busca-se aplicar o modelo desenvolvido em instâncias superiores as encontrada na literatura.

6.4 APLICAÇÃO DA HEURÍSTICA

Essa seção dedica-se a demonstrar a aplicabilidade da heurística desenvolvida. Para tanto três demonstrações são apresentadas: a primeira compara as respostas obtidas pela heurística em relação à modelagem clássica do MDVSP; a segunda demonstra aplicabilidade da heurística para instâncias superiores àquelas encontradas atualmente na literatura, ou seja, com um mais de 1500 viagens ou oito garagens; por fim, a terceira compara os

resultados obtidos pela heurística em relação aos resultados obtidos por outros autores.

A primeira demonstração busca, ao comparar a heurística com a modelagem clássica do MDVSP, verificar a eficiência do método de solução desenvolvido com uso do MDVSPcf. Obviamente quanto mais próximo da solução MDVSP a heurística apresentar suas respostas maior a sua eficiência. Contudo cabe ressaltar que a complexidade dos problemas MDVSP não permitiu que instâncias superiores a 500 viagens e quatro garagens fossem solucionadas de modo a garantir que a solução exata dos problemas fosse encontrada. Deste modo, as comparações foram realizadas limitando-se o tempo de solução dos problemas em 30 minutos, assim as respostas obtidas na Tabela 5 não representam a solução ótima, mas a melhor solução encontrada pelo algoritmo *branch-and-bound* considerando o tempo estipulado. Ressalta-se ainda que, apesar dos 30 minutos estipulados, dada a complexidade exponencial do MDVSP, não foi possível encontrar uma resposta exata para algumas das amostras geradas.

Posto isso, para melhorar a comparação também optou-se em solucionar o problema clássico MDVSP relaxando as variáveis binárias do modelo de modo que valores decimais pudessem ser aceitos como respostas viáveis. Na prática o relaxamento não demonstra uma solução viável para o problema, mas indica um limite inferior que serve de base de comparação de eficiência tanto para a heurística como a para a resposta MDVSP sem relaxamento.ATabela 5 mostra a comparação entre os resultados da heurística e o MDVSP clássico com e sem relaxamento das variáveis binárias. Como a resposta para o MDVSP está sempre situada entre a resposta MDVSPcf e a melhor resposta para o VSP, então também optou-se em colocar os valores desses dois modelo para fins comparativos.

Tabela 5 – Comparação da heurística com o MDVSP

amostra			MDVSPcf		MDVSP (relaxado)		MDVSP		melhor VSP		heurística		
Número de viagens	Número de garagens	Número de veículos	Tempo requerido	Solução ótima	Tempo requerido	Solução ótima	Tempo requerido	Solução	Tempo requerido	Solução ótima	Tempo requerido	Solução	Número de iterações
500	4	65	0m 3s	45.696	0m 17s	45.701	30m 0s	45.733	0m 1s	47.065	2m 20s	46.126	18
500	4	67	0m 3s	45.250	0m 16s	45.303	30m 0s	45.468	0m 1s	47.137	4m 59s	46.024	28
500	4	69	0m 2s	47.791	0m 20s	47.770	30m 0s	48.227	0m 1s	49.872	1m 12s	48.401	19
500	4	44	0m 2s	43.525	0m 31s	43.525	-	-	0m 1s	44.386	1m 53s	43.916	38
500	4	41	0m 2s	38.404	0m 22s	38.404	-	-	0m 1s	39.265	1m 7s	38.656	29
500	4	43	0m 4s	37.420	0m 27s	37.436	-	-	0m 1s	38.205	1m 44s	37.762	37
500	4	35	0m 2s	37.662	0m 22s	37.662	-	-	0m 1s	38.424	0m 48s	37.920	19
500	4	40	0m 2s	40.860	0m 20s	40.866	-	-	0m 1s	41.783	5m 26s	41.174	31
500	4	33	0m 3s	36.666	0m 31s	36.675	-	-	0m 1s	37.278	6m 32s	36.902	30

Fonte: o autor (2008).

Percebe-se pela Tabela 5 que a heurística desenvolvida consegue solucionar os problemas em menos tempo que a solução clássica do MDVSP, contudo os valores encontrados são em média 3% maiores. Apesar de uma resposta menos eficiente cabe lembrar que o objetivo do modelo é o de solucionar problemas para instâncias superiores a 1500 viagens ou oito garagens que até o momento ainda não foram resolvidos satisfatoriamente na literatura. Sendo assim o principal diferencial do modelo desenvolvido é a capacidade de resolver tais problemas. A Tabela 6 mostra uma gama de problemas solucionados considerando um número de viagens e garagens superior àqueles encontrados na literatura. Salienta-se que a complexidade exponencial dos problemas não permitiu que nem mesmo o modelo clássico MDVSP com relaxamento das variáveis binárias fosse resolvido, considerando o excessivo número de restrições e variáveis que o modelo apresenta. Sendo assim não foi possível comparar para essas instâncias as soluções obtidas pela heurística e pelo modelo clássico MDVSP.

Tabela 6 – Aplicação da heurística para instâncias superiores a da literatura

amostra			MDVSPcf		melhor VSP		heurística		
Número de viagens	Número de garagens	Número de veículos	Tempo requerido	Solução ótima	Tempo requerido	Solução ótima	Tempo requerido	Solução	Número de iterações
1500	6	178	0m 5s	114334	0m 4s	119552	2m 2s	117.242	13
1500	8	172	0m 6s	110484	0m 3s	115735	4m 33s	113.528	21
2000	6	217	0m 7s	144516	0m 8s	147854	4m 25s	146.543	13
2000	8	225	0m 9s	144131	0m 9s	151405	2m 39s	148.005	11
2500	6	269	0m 13s	174192	0m 18s	182397	1m 3s	178.709	5
2500	8	270	0m 21s	171628	0m 18s	179744	5m 7s	176.111	14
3000	6	325	0m 11s	206942	0m 29s	214778	4m 46s	212171	14
3000	8	323	0m 18s	205684	0m 27s	214835	10m 47s	210520	11

Fonte: o autor (2008).

Infelizmente a impossibilidade de verificar a diferença da resposta obtida pela heu-

rística em relação à solução ótima do modelo, não permite afirmar que as respostas produzidas pela heurística também apresentam 3% de variação. Contudo os tempos calculados mostram que a heurística é bastante eficiente nesse critério, tendo em vista que na literatura tais problemas nem conseguem ser solucionados para instâncias reproduzidas nessa tese. Exemplo dessa eficiência pode ser observado na instância com 3000 veículos e oito garagens que foi solucionado em apenas dez minutos e 47 segundos.

Após verificar a viabilidade de aplicação da heurística em problemas com um grande número de viagens e garagens, julgou-se oportuno verificar sua eficiência em relação ao trabalho de outro autor. Nesse caso utilizou-se as amostras geradas por Huisman (2004) que foram coletadas de sua página na *internet* para fins comparativos. Atualmente Huisman apresenta os melhores resultados para solução do MDVSP considerando amostras geradas aleatoriamente. A Tabela 7 compara os resultados obtidos.

Tabela 7 – Comparação da heurística com os resultados de Huisman

amostra		solução Huisman			solução heurística com proc. de redução 1, 2 e 3				
Número de viagens	Número de garagens	Número de Veículos	Tempo requerido	Solução	Número de Veículos	Tempo requerido	Solução	diferença % tempo	diferença % solução
500	4	123	77s	1.289.114	123	3s	1.293.520	-96,1%	0,34%
500	4	118	77s	1.241.618	118	3s	1.247.197	-96,1%	0,45%
500	4	123	77s	1.283.811	123	4s	1.287.524	-94,8%	0,29%
500	4	120	77s	1.258.634	120	4s	1.262.130	-94,8%	0,28%
500	4	126	77s	1.317.077	126	3s	1.318.833	-96,1%	0,13%
500	8	124	119s	1.292.411	124	10s	1.297.168	-91,6%	0,37%
500	8	123	119s	1.276.919	123	9s	1.280.376	-92,4%	0,27%
500	8	126	119s	1.304.251	126	18s	1.309.011	-84,9%	0,36%
500	8	123	119s	1.277.838	123	9s	1.282.800	-92,4%	0,39%
500	8	123	119s	1.276.010	123	9s	1.279.000	-92,4%	0,23%
1000	4	241	1287s	2.516.247	241	14s	2.528.126	-98,9%	0,47%
1000	4	229	1287s	2.413.393	229	10s	2.422.836	-99,2%	0,39%
1000	4	233	1287s	2.452.905	233	11s	2.455.638	-99,1%	0,11%
1000	4	237	1287s	2.490.812	237	11s	2.492.627	-99,1%	0,07%
1000	4	238	1287s	2.519.191	238	11s	2.510.126	-99,1%	-0,36%
1000	8	232	6206s	2.422.112	232	50s	2.433.226	-99,2%	0,46%
1000	8	244	6206s	2.524.293	244	58s	2.531.996	-99,1%	0,31%
1000	8	247	6206s	2.556.313	247	29s	2.565.015	-99,5%	0,34%
1000	8	237	6206s	2.478.393	237	52s	2.482.682	-99,2%	0,17%
1000	8	240	6206s	2.498.388	241	47s	2.512.434	-99,2%	0,56%
1500	4	368	4149s	3.830.912	368	34s	3.838.227	-99,2%	0,19%
1500	4	338	4149s	3.559.176	338	24s	3.561.268	-99,4%	0,06%
1500	4	350	4149s	3.649.757	350	23s	3.655.774	-99,4%	0,16%
1500	4	326	4149s	3.406.815	326	31s	3.419.648	-99,3%	0,38%
1500	4	343	4149s	3.567.122	343	33s	3.577.977	-99,2%	0,30%

Fonte: o autor (2008).

Para fins registro, os valores na coluna tempo apresentados na solução Huisman aparecem repetidos pois na página do autor só é informado o tempo médio de solução por instância. Percebe-se pelos dados na Tabela 7 que a heurística apresentou um resultado médio 0,27% pior que as soluções encontradas por Huisman. Em contra partida as soluções encontradas foram em média 97% mais rápidas. Isso significa que as reduções do espaço de estado utilizadas não comprometem a solução satisfatória de problemas MDVSP e garantem que resultados eficientes sejam encontrados com o algoritmo desenvolvido. Além disso, acredita-se que o custo adicional encontrado pela heurística pode ser tolerado diante do ganho em redução no tempo de solução dos problemas.

7 CONSIDERAÇÕES FINAIS

Esta pesquisa apresentou o desenvolvimento de uma heurística para a solução do problema MDVSP. Como citado durante a pesquisa, este é um problema de complexidade NP difícil, exigindo excessivo esforço computacional. Deste modo, a heurística desenvolvida deveria apresentar características que permitissem a tratabilidade deste tipo de problema, ou seja, necessidade de pouca memória computacional e a capacidade de solucionar o problema em pouco tempo.

Inicialmente tratou-se da redução do espaço de estados como uma forma de reduzir a instância de problemas sem comprometer a solução dos mesmos. Três procedimentos de redução do espaço de estados foram desenvolvidos e quando combinados mostraram resultados satisfatórios tendo em vista que atingiram uma redução média de 98% das variáveis dos modelos sem comprometer os resultados atingidos que em média ficaram prejudicados 0,3%. A redução do espaço de estados utilizada foi desenvolvida de forma exata mapeando e buscando pistas das melhores alternativas de solução através da resolução de problemas de complexidade polinomial, tais como o VSP. Entretanto a redução do espaço de estados não conduz por si própria a solução do problema. Deste modo num segundo momento foi desenvolvido um procedimento de busca da solução MDVSP.

O procedimento desenvolvido para solução do MDVSP teve como base o MDVSP com flexibilidade. Sendo assim a proposta da heurística consistia em flexibilizar o problema e depois forçar sua correção. Para efetuar a correção, duas restrições foram necessários: a primeira efetuava a inserção de restrições que corrigissem as seqüências inadequadas de viagens à solução MDVSP, enquanto a segunda cuidava da definição do valor mínimo para a solução MDVSP. Esse procedimento funciona de modo iterativo até que uma solução

adequada ao MDVSP seja encontrada.

Essa abordagem permite que pouca memória computacional seja utilizada e torna desnecessária a utilização de processamento paralelo em diversas máquinas ou o uso de supercomputadores na solução de problemas com elevado número de viagens e garagens. A Figura 18 resume a heurística desenvolvida, onde as etapas A até J referem-se à redução do espaço de estados e as etapas L até R a solução propriamente dita do problema MDVSP.

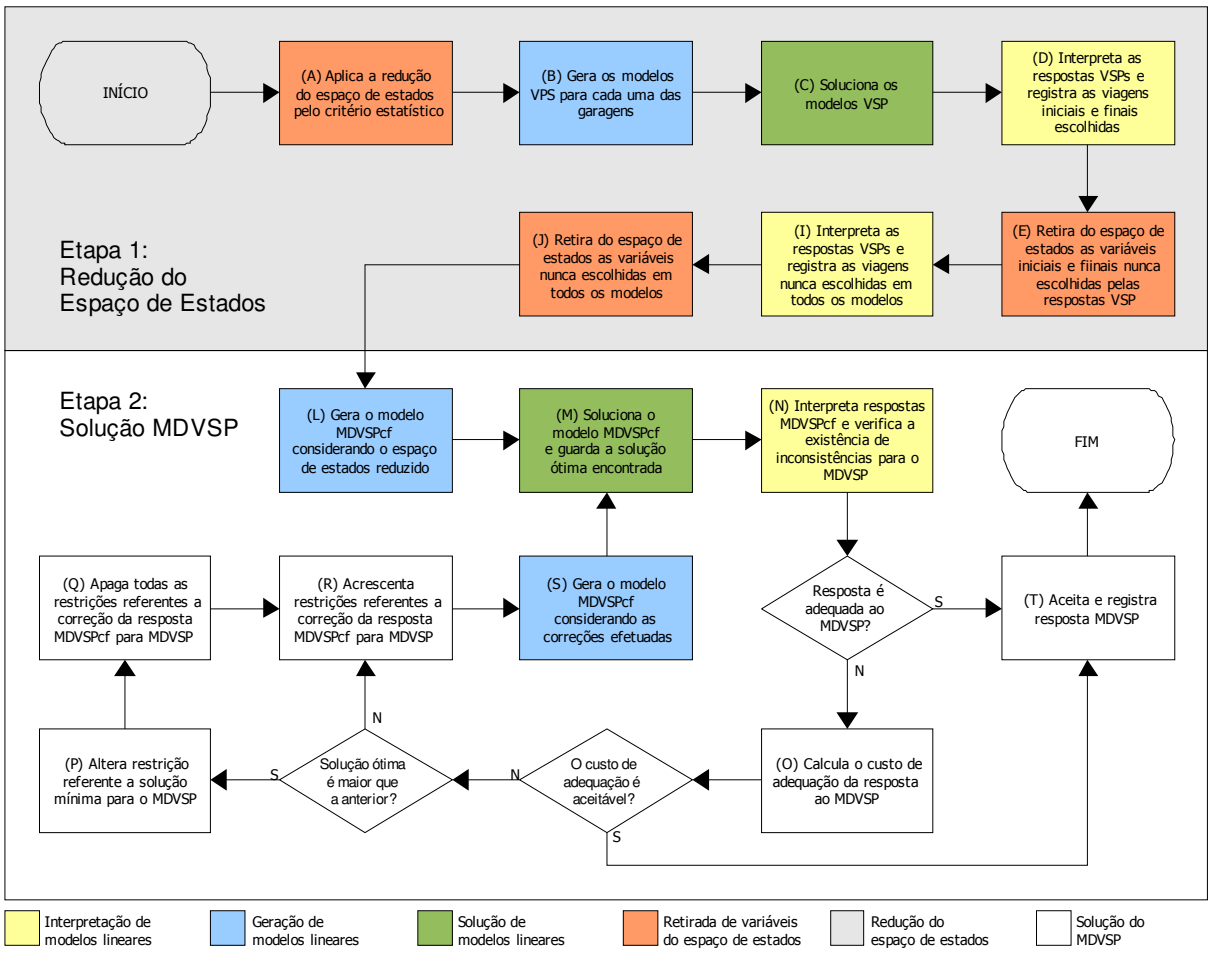


Figura 18 – Fluxograma da heurística
Fonte: o autor (2008).

Todos os modelos lineares gerados (etapas B, L e S) apresentam complexidade assintótica na ordem (n^2) posto que a função objetivo exige a leitura de todas as variáveis do modelo cujos custos estão expressos na matriz de folga total n x n. Analogamente, todas as etapas de interpretação dos modelos lineares (etapas D, I e N) apresentam complexidade

assintótica na ordem (n^2) tendo em vista que o número de variáveis que necessita de interpretação também pode ser definido através de n x n.

As etapas de retirada das variáveis do espaço de estados (A, E e J) apresentam complexidade assintótica na ordem (n^2), tendo em vista que o registro do espaço de estados reduzido ocorre na matriz de redução que possui ordem n x n. Analogamente a etapa Q apresenta complexidade n x n pois o registro é efetuado na matriz de escolhas.

As etapas O e R dependem do número de seqüências inadequadas que necessitam de correção a cada iteração. Como o número de seqüências inadequadas não ultrapassa o número de veículos, e o número de veículos não pode ultrapassar o número de viagens, então para estas etapas tem-se complexidade assintótica na ordem (n).

A etapa C apresenta complexidade assintótica O(n^2 log n) por se tratar de um modelo de designação resolvido pelo algoritmo *auction*. Contudo no que se refere à etapa M, após a execução da etapa R, o modelo MDVSPcf perde sua característica de modelo de designação pela adição das restrições de correção das seqüências inadequadas. Isso permite que algumas reflexões sobre a etapa M sejam elaboradas.

É preciso salientar que conforme aumenta-se o número de seqüências inadequadas no modelo, maior o tempo necessário para solucioná-lo. Observou-se que a partir de aproximadamente dez iterações o número de restrições acrescidas ao modelo original começa a impactar no tempo de solução do algoritmo. Isso significa que a cada iteração o modelo começa, em média, a demorar mais tempo para ser solucionado. A conseqüência disso é que em termos práticos a utilização do algoritmo *branch-and-bound* limita a utilização da heurística, considerando a peculiaridade desse algoritmo de buscar sempre uma solução ótima para os problemas. Apesar de dez iterações serem suficientes para se atingir uma boa conversão para uma solução satisfatória ao MDVSP é necessário ressaltar que outros algoritmos podem ser testados e apresentarem um resultado melhor que os obtidos nesse estudo.

Em outras palavras, a heurística desenvolvida nessa tese pode ser integrada sem prejuízo com outras heurísticas e algoritmos encontrados na literatura. Para tanto bastaria substituir o método de solução *branch-and-bound* utilizado na etapa M por outros métodos mais eficientes e rápidos. Optou-se pela utilização do *branch-and-bound* pois o mesmo conduz, apesar de sua complexidade, sempre a resposta ótima. Outros métodos de solução, como por exemplo *column generation* e *tabu-search*, podem ser testados conjuntamente com a heurística aqui desenvolvida. Pode-se afirmar que a facilidade de integração entre os modelos abre possibilidades para pesquisas futuras e melhoria nas soluções encontradas nessa tese. Apresentar nessa tese as inúmeras alternativas de integração seria inviável, tendo em vista que para cada uma delas consistiria de um novo estudo exigindo o desenvolvimento de novos modelos lineares, novos interpretadores de soluções e da própria heurística que seria integrada. Sendo assim, como mencionado anteriormente optou-se pelo uso do algoritmo *branch-and-bound*, deixando a inúmeras integrações possíveis como sugestão de pesquisa futura.

Estes benefícios são conseqüências de uma redução do espaço de estados abordada de maneira simples e compacta, a qual permite uma eficiente implementação para análise computacional. A heurística desenvolvida ainda pode ser facilmente integrada com outros algoritmos, métodos e heurísticas direcionados às análises e avaliações mais complexas de sistemas logísticos. Esta possibilidade de integração demonstra claramente o benefício da pesquisa desenvolvida em relação a outros estudos encontrados na literatura.

Apesar desta pesquisa focar o MDVSP, acredita-se que as reduções de espaço de estados aqui desenvolvidas podem ser aplicadas em diversas áreas do conhecimento. Assim, outros problemas, que também trabalhem com alguma forma de escalonamento de veículos, tais como MDVRSP, VSP, MDVCSP, CSP, VCSP entre outros, poderiam fazer uso dos procedimentos desenvolvidos, beneficiando-se das mesmas vantagens atingidas na solução do MDVSP.

Além das vantagens computacionais proporcionadas pela redução do espaço de estados, foi possível perceber as vantagens de sua aplicação em modelos lineares. Os modelo lineares, desenvolvidos até então, utilizavam uma grande quantidade de variáveis e restrições para poder tratar o problema de MDVSP. Geralmente, nestes modelos cada variável é composta por três indexadores, um indicando a viagem de saída, outro a viagem de chegada e um terceiro indexador para a garagem. Em alguns casos o aumento no número de restrições e variáveis dado a excessiva quantidade de indexadores inviabiliza o desenvolvimento de modelos linear. O MDVSPcf utilizado de modo iterativo com os procedimentos desenvolvidos apresenta apenas dois indexadores uma vez que o indexador relativo à garagem é suprimido. Isso implica na capacidade de tratamento do MDVSP com programação linear de forma mais eficiente.

Finalizando, acredita-se que a simplicidade dos procedimentos gerados nesta pesquisa permitirá que um computador central, responsável pelo planejamento logístico, prediga em menos tempo o comportamento do sistema sob diferentes configurações e múltiplas garagens. Conseqüentemente, a heurística poderá auxiliar na administração diária dos veículos, permitindo a aplicação de diversos critérios na seleção da melhor alternativa, de acordo com os parâmetros adotados em cada sistema (custo, tempo de transporte, tempo ocioso, prioridade, etc.). Deste modo, espera-se que este estudo sirva de base para construção de modelos de mensuração da performance logística e auxilie no uso mais eficiente dos recursos.

REFERÊNCIAS

ACKOFF, R L; SASIENI, M W. **Pesquisa Operacional**. Rio de Janeiro: Livros Técnicos e Científicos, 1977.

ALBUQUERQUE, F. **Programando em Linguagem C, C++ e turbo C++**. Rio de Janeiro: Berkeley, 1991.

AMICO D; FISCHETTI M; TOTH P. Heuristic algorithms for the multiple depot vehicle scheduling problem. **Management Science**, 39, p115-125, 1993.

ANDRADE, E L de. **Introdução a Pesquisa Operacional – Métodos e modelos para a análise de decisão**. Rio de Janeiro: LTC, 1998.

BAITA F; PESENTI R; UKOVICH W; FAVARETTO D. A comparison of different solution approaches to the vehicle scheduling problem in a practical case. **Computers & Operations Research**, 27, p1249-1269, 2000.

BALL M; BODIN L. A matching based heuristic for scheduling mass transit crews and vehicles. **Transportation Science**, 17, p4-31, 1983.

BASNET C; FOULDS L; IGBARIA M. FleetManager: a microcomputer-based decision support system for vehicle routing. **Decision Support systems**, 16, p195–207, 1996.

BERTALANFFY, L V. **Teoria Geral dos Sistemas**. Editora: Vozes, Rio de Janeiro, 1975.

BERTOSSI A A; CARRARESI P; GALLO G. On some matching problems arising in vehicle scheduling models. **Network**, 17, p271-281, 1987.

BODIN L; GOLDEN B. Classification in vehicle routing and scheduling. **Networks**, 11, p97-108, 1981.

BODIN L; GOLDEN B; ASSAD A; BALL M. Routing and scheduling of vehicles and crew - the state of the art. **Computers and Operations Research**, 10, p63-212, 1983.

BODIN, L; ROSENfiELD, D; KYDES, A. UCOST: A Micro Approach to a Transit Planning Problem. **Journal of Urban Analysis** 5, p47–69, 1978.

BOFFEY, T B. **Graph Theory in Operations Research**. Hong Kong: Macmillan, 1984.

BOKINGE U; HASSELSTROM D. Improved vehicle scheduling in public transport through systematic changes in the time-table. **European Journal of Operational Research**, 5, p388-395, 1980.

CAMPELLO R E; MACULAN N. **Algoritmos e Heurísticas - Desenvolvimento e Avaliação de Performance**. Niterói: EDUFF, 1994.

CARLSON P M. Exploiting the opportunities of collaborative decision making: a model and efficient solution algorithm for airline use. **Transportation Science**, 34, p381-393, 2000.

CARPANETO, G; DELL'AMICO, M; FISCHETTI, M; TOTH P. A branch and bound algorithm for the multiple depot vehicle scheduling problem. **Networks**, no 19, p531-548, 1989.

CARRARESI, P; GALLO G. Network models for vehicle and crew scheduling. **European Journal of Operations Research**, 16, p139-151, 1984.

CORDEAU J F; GENDREAU M; LAPORTE G; POTVIN J Y; SEMET F. A guide to vehicle routing heuristics. **Journal of the Operational Research Society**, 53, p512–522, 2002.

DADUNA J R; PAIXÃO J M. Vehicle scheduling for public mass transit: an overview. **Computer-Aided Transit Scheduling**: apresentado em: Sixth International Workshop, p76-90, Boston, MA, 1995.

DANTZIG G B; RAMSER J. The truck Dispatching Problem. **Management Science**, 6, p81-91, 1959.

DELL'AMICO, M; FISCHETTI, M; TOTH P. Heuristic Algorithms for the Multiple Depot Vehicle Scheduling Problem. **Management Science** 39, p115–125, 1993.

DESAULNIERS, G; HICKMAN, M. Public Transit. Em: LAPORTE, G. e BARNHART, C. (editores), Transportation, **Handbooks in Operations Research and Management Science**, Elsevier Science, Amsterdam. (aguardando publicação), 2007.

DESROCHERS M; LENSTRA J K; SAVELSBERGH M W; STOUGIE L. Towards a model and algorithm management system for vehicle routing and scheduling problems. **Decision Support Systems**, 25, p109–113, 1999.

DESROSIERS, J; DUMAS, Y; SOLOMON, M M; SOUMIS, F. Time Constrained Routing and Scheduling. Em: BALL, M O; MAGNANTI, T L; MONMA, C L; NEMHAUSER, G L (editores), Network Routing, **Handbooks in Operations Research and Management Science** 8, Elsevier Science, Amsterdam, p35–139, 1995.

FLOOD, R; CARSON, E. **Dealing with Complexity**. New York: Plenum Press, 1993.

FRELING R; WAGELMANS A P; PAIXÃO J M. Models and algorithms for single-depot vehicle scheduling. **Transportation Science**, 3, 2, p165–180, 2001.

GAVISH B; SHIFLER E. An approach for solving a class of transportation scheduling problems. **European Journal of Operations Research**, 3, p12-134 ,1978.

GENDREAU M; POTVIN J-Y. Dynamic vehicle routing and dispatching. CRAINIC T; LAPORTE, G (editores). **Fleet Management and Logistics**. Kluwer, New York, p115-126, 1998.

GHIANI G; GUERRIERO F; LAPORTE G; MUSMANNO R. Real-time vehicle routing: solution concepts, algorithms and parallel computing strategies. **European Journal of Operational Research**, 151(1), p1-11, 2003.

GOLDBARG M C; LUNA H P. **Otimização Combinatória e Programação Linear - Modelos e Algoritmos**. Rio de Janeiro: Campus, 2000.

HADJAR, A; MARCOTTE, O; SOUMIS, F. A Branch-and-Cut Algorithm for the Multiple Depot Vehicle Scheduling Problem. **Operations Research** 54, p130–149, 2006.

HAGHANI, A; BANIHASHEMI, M. Heuristic approaches for solving large-scale bus transit vehicle scheduling problem with route time constraints. **Transport Research Part A**, 36, p309-333, 2002.

HUISMAN, D; FRELING, R; WAGELMANS, A. A robust solution approach to the dynamic vehicle scheduling problem. **Transportation Science**, v38, 4, p447-458, 2004.

HUISMAN, D. **Integrated and Dynamic Vehicle and Crew Scheduling**. Tese de doutorado, Erasmus Univesiteit Rotterdam, 2004.

HWANG, C; GUO, T Y; SHIEH L S A canonical state-space representation for systems using Multipoint. **Journal of the Franklin Institute**, vol 328, no 2/3, p207-216, 1991.

ICHOUA S; GENDREAU M; POTVIN J-Y. Diversion issues in real-rime vehicle dispatching. **Transportation Science**, 34, p426-438, 2000.

JONKER R; VOLGENANT A. Improving the Hungarian assignment problem. **Operations Research Letters**, 5, p71-176, 1986.

KLIEWER, N; MELLOULI, T; SUHL, L. A Time-Space Network Based Exact Optimization Model for Multi-Depot Bus Scheduling. **European Journal of Operational Research**. (aguardando publicação), 2006.

LAMATSCH, A. An Approach to Vehicle Scheduling with Depot Capacity Constraints. Em: DESROCHERS M. e ROUSSEAU J. M. (editores). **Computer Aided Transit Scheduling, Lecture Notes in Economics and Mathematical Systems** 386, Springer-Verlag, Berlin, p181–195, 1992.

LANDRY, M; MALOUIN, J L; ORAL, M. Model validation in Operations Research. **European Journal of Operational Research**, vol 14, no 3, 1983.

LAPORTE G; LOUVEAUX F V. Solving stochastic routing problems with integer l-shaped method. Crainic T e Laporte G, eds., **Fleet Management and Logistics**. Kluwer, New York, p159-167, 1998.

LETTOVSKY L. **Airline operations recovery: an optimization approach**. Tese de doutorado, Georgia Institute of Technology, Estados Unidos, 1997.

LI J Q. **Models and Algorithms of real-time vehicle rescheduling problem under disruption**. Tese de doutorado, Departament of Systems and industrial Engineering, University of Arizona, 2006.

LI J Q; BORENSTEIN D; MIRCHANDANI P B. A decision support system for the single-depot vehicle rescheduling problem. **Computers and Operations Research**, submetido em, 2005.

LI J Q; MIRCHANDANI P B; BORENSTEIN D. Parallel auction algorithm for bus rescheduling. Apresentado em: **9th International Conference on Computer-Aided Scheduling of Public Transport**. San Diego, California, USA, 2004.

LÖBEL, A. **Optimal Vehicle Scheduling in Public Transit**. Tese de Ph.D, Technische Universität Berlin, Berlin, Germany, 1997.

LÖBEL, A. Vehicle Scheduling in Public Transit and Lagrangian Pricing. **Management Science** 44, p1637–1649, 1998.

MARTINS, G A. **Estatística Geral e Aplicada**. São Paulo: Atlas, 2002.

MESQUITA, M; PAIXÃO, J. Multiple Depot Vehicle Scheduling Problem: A New Heuristic Based on Quasi-Assignment Algorithms. Em: DESROCHERS M. e ROUSSEAU J. M. (editores), **Computer-Aided Transit Scheduling, Lecture Notes in Economics and Mathematical Systems** 386, Springer-Verlag, Berlin, p167–180, 1993.

MORGAN, E T; RAZOUK, R R. Interactive State-Space Analysis of Concurrent Systems.

IEEE Transactions on Software Engineering, 13, 10, p1080-1091, 1987.

NUSSBAUM M; SEVULPEDA M; COBIAN A; GAETE J; PARRA E; CRUZ J. A fuel distribution knowledge-based decision support system. **Omega**, 25, p225–234, 1997.

ODONI, A R; ROUSSEAU, J M; WILSON, N H M. Models in Urban and Air Transportation. Em: POLLOCK, S. M; ROTHKOPF, M. H. e BARNETT, A. (editores), **Operations Research and the Public Sector, Handbooks in Operations Research and Management Science** 6, North-Holland, Amsterdam, p107–150, 1994.

PAIXÃO J M; BRANCO I. A quasi-assignment algorithm for bus scheduling. **Networks**, 17, p249-269, 1987.

PIDD, M. **Modelagem Empresarial – ferramentas para tomada de decisão**. Porto Alegre: Bookman, 1998.

POOT A; KANT G; WAGELMANS A P M. A savings based method for real-life vehicle routing problems. **Journal of the Operational Research Society**, 53, p57–68, 2002.

PSARAFTIS H N. Dynamic vehicle routing: Status and prospects. **Annals of Operations Research**, 61, p143-164, 1995.

REISDORPH, K. **Aprenda em 21 dias Delphi 4**. Rio de Janeiro: Campus, 1999.

RIBEIRO, C; SOUMIS F. A Column Generation Approach to the Multiple Depot Vehicle Scheduling Problem. **Operations Research** 42, p41–52, 1994.

ROSENBERGER J M; JOHNSON E L; NEMHAUSER, J L. Rerouting aircraft for airline recovery, **Transportation Science**, 37, p408-421, 1987.

ROTHENBERG, J. The Nature of Modeling. Em: WILDMAN, L E; LOPARO, K A; NIELSEN, N R (editores). **Artificial Intelligence, Simulation and Modeling**, Wiley, p75-92, 1989.

RUIZ R; MAROTO C; ALCARAZ J. A decision support system for a real vehicle routing problem. **European Journal of Operational Research**, 153, p593–606, 2004.

SHAMBLIN, J E; STEVENS JR, G T **Pesquisa Operacional – uma abordagem básica**. São Paulo: Atlas, 1979.

SIMON, H. A. Prediction and Prescription in Systems Modeling. **OR Forum**, p7-13, 1989.

SONG T; ZHOU L. A new algorithm for the quasi-assignment problem. **Annals of Operations Research**, 37, p205-223, 1990.

TEODOROVIC D; STOJKOVIC G. Model to reduce airline schedule disturbances. **Jornal of Transportation Engineering**, 121, p324-331, 1995.

TOSCANI L V; VELOSO P A. **Complexidade de algoritmos**. Porto Alegre: Instituto de Informática da UFRGS: Sagra Luzzatto, 2001.

YANG J; JAILLET P; MAHMASSANI H. Real-time multivehicle truckload pickup and delivery problems. **Transportation Science**, 38, p135-148, 2004.

APÊNDICE A – Módulo gerador de viagens aleatórias (GVA)

UNIDADE A

```
unit dout_gav;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs,
  StdCtrls, Grids, ExtCtrls;

type
  TFormDoutGav = class(TForm)
    EditFileName: TEdit;
    EditDistY: TEdit;
    StringGridQH: TStringGrid;
    EditM: TEdit;
    EditN: TEdit;
    Label3: TLabel;
    EditDistX: TEdit;
    ButtonSave: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label4: TLabel;
    Bevel1: TBevel;
    Bevel2: TBevel;
    Label6: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label5: TLabel;
    EditNumV: TEdit;
    Label7: TLabel;
    EditJT: TEdit;
    Label8: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    CheckBox1: TCheckBox;
    ButtonGerarHorar: TButton;
    ButtonAux1: TButton;
    ButtonVisMat: TButton;
    Bevel4: TBevel;
    Label13: TLabel;
    EditCustVeic: TEdit;
    ButtonAux2: TButton;
    ButtonAux3: TButton;
    ButtonAux4: TButton;
    EditNumGarag: TEdit;
    Label14: TLabel;
    StringGridGarag: TStringGrid;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    ButtonOpen: TButton;
    procedure EditDistXChange(Sender: TObject);
    procedure EditDistYChange(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
    procedure ButtonGerarHorarClick(Sender: TObject);
    procedure ButtonAux1Click(Sender: TObject);
    procedure EditNChange(Sender: TObject);
    procedure EditMChange(Sender: TObject);
    procedure ButtonVisMatClick(Sender: TObject);
    procedure EditFileNameChange(Sender: TObject);
    procedure EditCustVeicChange(Sender: TObject);
    procedure ButtonSaveClick(Sender: TObject);
    procedure ButtonOpenClick(Sender: TObject);
    procedure ButtonAux2Click(Sender: TObject);
    procedure ButtonAux3Click(Sender: TObject);
    procedure ButtonAux4Click(Sender: TObject);
    procedure EditNumGaragChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormDoutGav: TFormDoutGav;
  aux1, aux2, maxviagem, numcol, numrow, NumViagens,
CustoVeic, NumGarag: integer;
  Nomedoarquivo, s: string;
  matrizhorar, matrizgarag, matrizViagExpGaragIni, matrizVia-
gExpGaragFim, matrizviagexp, matrizocios, matrizft, matrizo-
cios_vrsp, matrizft_vrsp: array of array of integer;
  matrizaux_mdvsp: array of array of array of integer;
  arquivo: file of integer;

implementation

uses dout_mat;

{$R *.DFM}

procedure TFormDoutGav.FormCreate(Sender: TObject);
begin
stringgridQH.cells[1,0]:='DIx';
stringgridQH.cells[2,0]:='DIy';
stringgridQH.cells[3,0]:='DFx';
stringgridQH.cells[4,0]:='DFy';
stringgridQH.cells[5,0]:='HINI';
stringgridQH.cells[6,0]:='TV';
stringgridQH.cells[7,0]:='HTER';
stringgridGarag.cells[1,0]:='Dx';
stringgridGarag.cells[2,0]:='Dy';
nomedoarquivo:=EditFileName.text;
CustoVeic:=strtoint(EditCustVeic.Text);
NumGarag:=strtoint(EditNumGarag.Text);
end;

procedure TFormDoutGav.ButtonVisMatClick(Sender: TObject);
begin
FormMatriz.Showmodal;
end;

procedure TFormDoutGav.EditFileNameChange(Sender: TObject);
begin
nomedoarquivo:=EditFileName.text;
end;

procedure TFormDoutGav.EditCustVeicChange(Sender: TObject);
begin
CustoVeic:=strtoint(EditCustVeic.Text);
end;

procedure TFormDoutGav.EditNumGaragChange(Sender:
TObject);
begin
NumGarag:=strtoint(EditNumGarag.Text);
end;

procedure TFormDoutGav.EditNChange(Sender: TObject);
begin
ButtonAux1.Click;
end;

procedure TFormDoutGav.EditMChange(Sender: TObject);
begin
ButtonAux1.Click;
end;

procedure TFormDoutGav.EditDistXChange(Sender: TObject);
begin
if CheckBox1.Checked then ButtonAux1.Click;
end;

procedure TFormDoutGav.EditDistYChange(Sender: TObject);
begin
if CheckBox1.Checked then ButtonAux1.Click;
end;

procedure TFormDoutGav.CheckBox1Click(Sender: TObject);
begin
if Checkbox1.checked then
  begin
  EditJT.Enabled:=false;
```

```
  EditNumV.Enabled:=false;
  EditN.Enabled:=true;
  EditM.Enabled:=true;
  ButtonAux1.Click;
  end
    else
  begin
  EditJT.Enabled:=true;
  EditNumV.Enabled:=true;
  EditN.Enabled:=false;
  EditM.Enabled:=false;
  end
end;

procedure TFormDoutGav.ButtonGerarHorarClick(Sender:
TObject);
var
j: integer;
begin
ButtonAux2.Click;
{sorteia inicios e fins das viagens e calcula suas duracoes}
for j:=1 to NumViagens do
  begin
  matrizhorar[0,j-1]:=random(strtoint(EditDistX.Text)+1);
  matrizhorar[1,j-1]:=random(strtoint(EditDistY.Text)+1);
  matrizhorar[2,j-1]:=random(strtoint(EditDistX.Text)+1);
  matrizhorar[3,j-1]:=random(strtoint(EditDistY.Text)+1);
  matrizhorar[4,j-1]:=random(strtoint(EditJT.Text)+1);
  aux1:=matrizhorar[0,j-1]-matrizhorar[2,j-1];
  aux1:=aux1*aux1;
  aux2:=matrizhorar[1,j-1]-matrizhorar[3,j-1];
  aux2:=aux2*aux2;
  aux1:=trunc(sqrt(aux1+aux2));
  matrizhorar[5,j-1]:=aux1;
  matrizhorar[6,j-1]:=matrizhorar[4,j-1]+aux1;
  end;
{sorteia posicoes das garagens}
for j:=1 to NumGarag do
  begin
  matrizgarag[0,j-1]:=random(strtoint(EditDistX.Text)+1);
  matrizgarag[1,j-1]:=random(strtoint(EditDistY.Text)+1);
  end;
ButtonAux3.click;
ButtonAux4.click;
ButtonVisMat.Enabled:=true;
ButtonSave.Enabled:=true;
end;

{salvando arquivo com grade de horários}
procedure TFormDoutGav.ButtonSaveClick(Sender: TObject);
var
j: integer;
begin
s:=EditFileName.text+'.gva';
AssignFile(arquivo,s);
rewrite(arquivo);
reset(arquivo);
{salva N}
aux1:=strtoint(EditN.text);
write(arquivo,aux1);
{salva M}
aux1:=strtoint(EditM.text);
write(arquivo,aux1);
{salva DistX}
aux1:=strtoint(EditDistX.text);
write(arquivo,aux1);
{salva DistY}
aux1:=strtoint(EditDistY.text);
write(arquivo,aux1);
{salva JT}
aux1:=strtoint(EditJT.text);
write(arquivo,aux1);
{salva numero de viagens}
aux1:=NumViagens;
write(arquivo,aux1);
{salva custo dos veiculos}
aux1:=CustoVeic;
write(arquivo,aux1);
{salva numero de garagens}
aux1:=NumGarag;
write(arquivo,aux1);
{salva grade de horários}
for j:=1 to NumViagens do
  begin
  aux1:=matrizhorar[0,j-1];
  write(arquivo,aux1);
  aux1:=matrizhorar[1,j-1];
  write(arquivo,aux1);
  aux1:=matrizhorar[2,j-1];
  write(arquivo,aux1);
  aux1:=matrizhorar[3,j-1];
  write(arquivo,aux1);
  aux1:=matrizhorar[4,j-1];
  write(arquivo,aux1);
  aux1:=matrizhorar[5,j-1];
  write(arquivo,aux1);
  aux1:=matrizhorar[6,j-1];
  write(arquivo,aux1);
  end;
{salva posição das garagens}
for j:=1 to NumGarag do
  begin
  aux1:=matrizgarag[0,j-1];
  write(arquivo,aux1);
  aux1:=matrizgarag[1,j-1];
  write(arquivo,aux1);
  end;
closefile(arquivo);
ButtonSave.Enabled:=false;
end;

{abrindo arquivo com grade de horários}
procedure TFormDoutGav.ButtonOpenClick(Sender: TObject);
var
j: integer;
begin
s:=EditFileName.text+'.gva';
AssignFile(arquivo,s);
reset(arquivo);
{abre  N}
read(arquivo,aux1);
EditN.text:=inttostr(aux1);
{abre  M}
read(arquivo,aux1);
EditM.text:=inttostr(aux1);
{abre DistX}
read(arquivo,aux1);
EditDistX.text:=inttostr(aux1);
{abre DistY}
read(arquivo,aux1);
EditDistY.text:=inttostr(aux1);
{abre JT}
read(arquivo,aux1);
EditJT.text:=inttostr(aux1);
{abre numero de viagens}
read(arquivo,aux1);
EditNumV.text:=inttostr(aux1);
{abre custo veiculos}
read(arquivo,aux1);
EditCustVeic.text:=inttostr(aux1);
{abre numero de garagens}
read(arquivo,aux1);
EditNumGarag.text:=inttostr(aux1);
ButtonAux2.Click;
{abre grade de horários}
for j:=1 to NumViagens do
  begin
  read(arquivo,aux1);
  matrizhorar[0,j-1]:=aux1;
  read(arquivo,aux1);
  matrizhorar[1,j-1]:=aux1;
  read(arquivo,aux1);
  matrizhorar[2,j-1]:=aux1;
  read(arquivo,aux1);
  matrizhorar[3,j-1]:=aux1;
  read(arquivo,aux1);
  matrizhorar[4,j-1]:=aux1;
  read(arquivo,aux1);
  matrizhorar[5,j-1]:=aux1;
  read(arquivo,aux1);
  matrizhorar[6,j-1]:=aux1;
  end;
{abre posição das garagens}
for j:=1 to Numgarag do
  begin
  read(arquivo,aux1);
  matrizgarag[0,j-1]:=aux1;
  read(arquivo,aux1);
  matrizgarag[1,j-1]:=aux1;
  end;
ButtonAux3.Click;
ButtonAux4.Click;
```

```
closefile(arquivo);
ButtonVisMat.Enabled:=true;
end;

{Calcula janela de tempo e numero de veículos automaticamente}
procedure TFormDoutGav.ButtonAux1Click(Sender: TObject);
begin
aux1:=strtoint(EditDistX.Text)*strtoint(EditDistX.Text);
aux2:=strtoint(EditDistY.Text)*strtoint(EditDistY.Text);
aux2:=aux2+aux1;
aux1:=trunc(sqrt(aux2));
editJT.text:=inttostr(strtoint(EditN.text)*aux1);
edit-
NumV.text:=inttostr(trunc(strtoint(EditJT.text)*strtoint(EditM.text)/60
));
end;

{Define tamanho das matrizes dinâmicas}
procedure TFormDoutGav.ButtonAux2Click(Sender: TObject);
begin
NumViagens:=strtoint(EditNumV.Text);
setlength(matrizhorar,7,NumViagens);
setlength(matrizviagexp,NumViagens,NumViagens);
setlength(matrizviagexpgaragini,NumViagens,NumGarag);
setlength(matrizviagexpgaragfim,NumViagens,NumGarag);
setlength(matrizocios,NumViagens,NumViagens);
setlength(matrizft,NumViagens,NumViagens);
setlength(matrizgarag,2,NumGarag);
setleng-
th(matrizaux_mdvsp,NumViagens,NumViagens,NumGarag);
end;

{mostra valores sorteados nas grades de horários}
procedure TFormDoutGav.ButtonAux3Click(Sender: TObject);
var
j: integer;
begin
stringgridQH.Rowcount:=NumViagens+1;
for j:=1 to NumViagens do
  begin
  stringgridQH.cells[0,j]:=inttostr(j);
  stringgridQH.cells[1,j]:=inttostr(matrizhorar[0,j-1]);
  stringgridQH.cells[2,j]:=inttostr(matrizhorar[1,j-1]);
  stringgridQH.cells[3,j]:=inttostr(matrizhorar[2,j-1]);
  stringgridQH.cells[4,j]:=inttostr(matrizhorar[3,j-1]);
  stringgridQH.cells[5,j]:=inttostr(matrizhorar[4,j-1]);
  stringgridQH.cells[6,j]:=inttostr(matrizhorar[5,j-1]);
  stringgridQH.cells[7,j]:=inttostr(matrizhorar[6,j-1]);
  end;
stringgridGarag.Rowcount:=NumGarag+1;
for j:=1 to NumGarag do
  begin
  stringgridGarag.cells[0,j]:=inttostr(j);
  stringgridGarag.cells[1,j]:=inttostr(matrizgarag[0,j-1]);
  stringgridGarag.cells[2,j]:=inttostr(matrizgarag[1,j-1]);
  end;
end;

{calcula matrizes de custo e tempo de viagens}
procedure TFormDoutGav.ButtonAux4Click(Sender: TObject);
var
i,j: integer;
begin
{calcula matriz de tempos de viagem expressa entre viagens}
for j:=1 to NumViagens do
  for i:=1 to NumViagens do
    begin
    aux1:=matrizhorar[2,i-1]-matrizhorar[0,j-1];
    aux1:=aux1*aux1;
    aux2:=matrizhorar[3,i-1]-matrizhorar[1,j-1];
    aux2:=aux2*aux2;
    matrizviagexp[j-1,i-1]:=trunc(sqrt(aux1+aux2));
    end;
{calcula matriz de tempos de ociosidade}
for j:=1 to NumViagens do
  for i:=1 to NumViagens do
    begin
    aux1:=matrizhorar[4,i-1];
    aux2:=matrizhorar[6,j-1];
    matrizocios[j-1,i-1]:=aux1-aux2-matrizviagexp[j-1,i-1];
    end;
{calcula matriz de tempos de folga total}
for j:=1 to NumViagens do
  for i:=1 to NumViagens do
    if (matrizocios[j-1,i-1]>=0) then
      matrizft[j-1,i-1]:=matrizocios[j-1,i-1]+matrizviagexp[j-1,i-1]
      else matrizft[j-1,i-1]:=0;
{calcula matriz de tempos de viagem expressa do depot até
viagem}
for i:=1 to NumGarag do
  for j:=1 to NumViagens do
    begin
    aux1:=matrizhorar[0,j-1]-matrizgarag[0,i-1];
    aux1:=aux1*aux1;
    aux2:=matrizhorar[1,j-1]-matrizgarag[1,i-1];
    aux2:=aux2*aux2;
    matrizViagExpGaragIni[j-1,i-1]:=trunc(sqrt(aux1+aux2));
    aux1:=matrizhorar[2,j-1]-matrizgarag[0,i-1];
    aux1:=aux1*aux1;
    aux2:=matrizhorar[3,j-1]-matrizgarag[1,i-1];
    aux2:=aux2*aux2;
    matrizViagExpGaragFim[j-1,i-1]:=trunc(sqrt(aux1+aux2));
    end;

end;

end.
```

UNIDADE B

```
unit dout_mat;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs,
  StdCtrls, Grids, ExtCtrls;

type
  TFormMatriz = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    StringGrid1: TStringGrid;
    Button1: TButton;
    ComboBox1: TComboBox;
    Label3: TLabel;
    Bevel1: TBevel;
    StringGridGaragIni: TStringGrid;
    Label4: TLabel;
    Label5: TLabel;
    StringGridGaragFim: TStringGrid;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormMatriz: TFormMatriz;

implementation

uses dout_gav;

{$R *.DFM}

procedure TFormMatriz.Button1Click(Sender: TObject);
var
i,j:integer;
begin
stringgrid1.colcount:=NumViagens+1;
stringgrid1.rowcount:=NumViagens+1;
stringgridGaragIni.colcount:=NumViagens+1;
stringgridGaragFim.colcount:=NumViagens+1;
stringgridGaragIni.rowcount:=Numgarag+1;
stringgridGaragFim.rowcount:=NumGarag+1;
for i:=1 to NumGarag do
  begin
  stringgridGaragIni.cells[0,i]:=inttostr(i);
  stringgridGaragFim.cells[0,i]:=inttostr(i);
  end;
aux1:=0;
for i:=1 to NumViagens do
  begin
  stringgrid1.cells[0,i]:=inttostr(i);
  Stringgrid1.cells[i,0]:=inttostr(i);
```

```
  stringgridGaragIni.cells[i,0]:=inttostr(i);
  stringgridGaragFim.cells[i,0]:=inttostr(i);
  end;
if combobox1.text='ociosidade' then
  begin
  for j:=1 to NumViagens do
    for i:=1 to NumViagens do string-
grid1.cells[i,j]:=inttostr(matrizocios[j-1,i-1]);
  label3.caption:='Matriz de Ociosidade';
  stringgridGaragIni.visible:=false;
  stringgridGaragFim.visible:=false;
  label4.visible:=false;
  label6.visible:=false;
  label7.visible:=false;
  label8.visible:=false;
  label5.caption:='';
  end
    else
  if combobox1.text='folga total' then
    begin
    for j:=1 to NumViagens do
      for i:=1 to NumViagens do
        if (matrizocios[j-1,i-1]>=0) then
          begin
          stringgrid1.cells[i,j]:=inttostr(matrizft[j-1,i-1]);
          inc(aux1);
          end
            else stringgrid1.cells[i,j]:='';
    label3.caption:='Matriz de Folga Total';
    stringgridGaragIni.visible:=false;
    stringgridGaragFim.visible:=false;
    label4.visible:=false;
    label6.visible:=false;
    label7.visible:=false;
    label8.visible:=false;
    label5.caption:='Total de variáveis: '+inttostr(aux1);
    end
      else
    begin
    combobox1.text:='viagem expressa';
    for j:=1 to NumViagens do
      for i:=1 to NumViagens do string-
grid1.cells[i,j]:=inttostr(matrizviagexp[j-1,i-1]);
    for j:=1 to NumViagens do
      for i:=1 to Numgarag do
        begin
        stringgridGaragI-
ni.cells[j,i]:=inttostr(matrizViagExpGaragIni[j-1,i-1]);
        stringgridGarag-
Fim.cells[j,i]:=inttostr(matrizViagExpGaragFim[j-1,i-1]);
        end;
    label3.caption:='Matriz de Viagem Expressa';
    stringgridGaragIni.visible:=true;
    stringgridGaragFim.visible:=true;
    label4.visible:=true;
    label6.visible:=true;
    label7.visible:=true;
    label8.visible:=true;
    label5.caption:='';
    end;
end;

procedure TFormMatriz.FormActivate(Sender: TObject);
begin
Button1.click;
end;

end.
```

APÊNDICE B – Módulo de redução

```
unit dout_vsp;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs,
  StdCtrls, Grids, ExtCtrls;

type
  TFormDoutVsp = class(TForm)
    EditFileName: TEdit;
    EditDistY: TEdit;
    EditM: TEdit;
    EditN: TEdit;
    EditDistX: TEdit;
    ButtonOpen: TButton;
    Label2: TLabel;
    Label4: TLabel;
    Bevel2: TBevel;
    Label6: TLabel;
    Label10: TLabel;
    Label5: TLabel;
    EditNumV: TEdit;
    Label7: TLabel;
    EditJT: TEdit;
    Label8: TLabel;
    Label11: TLabel;
    ButtonGerarLMo: TButton;
    ButtonEspEst: TButton;
    Label13: TLabel;
    EditCustVeic: TEdit;
    ButtonRspLingo: TButton;
    ButtonAux2: TButton;
    ButtonAux4: TButton;
    EditNumGarag: TEdit;
    Label14: TLabel;
    Label17: TLabel;
    ButtonViagCom: TButton;
    procedure FormCreate(Sender: TObject);
    procedure ButtonGerarLMoClick(Sender: TObject);
    procedure EditFileNameChange(Sender: TObject);
    procedure ButtonRspLingoClick(Sender: TObject);
    procedure ButtonOpenClick(Sender: TObject);
    procedure ButtonAux2Click(Sender: TObject);
    procedure ButtonAux4Click(Sender: TObject);
    procedure ButtonEspEstClick(Sender: TObject);
    procedure ButtonViagComClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormDoutVsp: TFormDoutVsp;
  aux1, aux2, maxviagem, NumViagens, CustoVeic, NumGarag:
integer;
  Nomedoarquivo, s: string;
  matrizhorar, matrizgarag, matrizViagExpGaragIni, matrizVia-
gExpGaragFim, matrizviagexp, matrizocios, matrizft, matrizgara-
gesc: array of array of integer;
  matrizaux_mdvsp: array of array of array of integer;
  arquivo: file of integer;

implementation

uses dout_lmo_vsp, dout_rsp_vsp, dout_red, dout_viagcom;

{$R *.DFM}

procedure TFormDoutVsp.FormCreate(Sender: TObject);
begin
nomedoarquivo:=EditFileName.text;
end;

procedure TFormDoutVsp.ButtonGerarLMoClick(Sender: TObject);
begin
FormLinearModelVsp.Showmodal;
end;

procedure TFormDoutVsp.ButtonRspLingoClick(Sender: TObject);
begin
FormRspLingo.showmodal;
end;

procedure TFormDoutVsp.ButtonEspEstClick(Sender: TObject);
begin
FormRed.showmodal;
end;

procedure TFormDoutVsp.ButtonViagComClick(Sender: TObject);
begin
FormViagCom.showmodal;
end;

procedure TFormDoutVsp.EditFileNameChange(Sender: TObject);
begin
nomedoarquivo:=EditFileName.text;
end;

{abrindo arquivo com grade de horários}
procedure TFormDoutVsp.ButtonOpenClick(Sender: TObject);
var
j: integer;
begin
s:=nomedoarquivo+'.gva';
AssignFile(arquivo,s);
reset(arquivo);
{abre  N}
read(arquivo,aux1);
EditN.text:=inttostr(aux1);
{abre  M}
read(arquivo,aux1);
EditM.text:=inttostr(aux1);
{abre DistX}
read(arquivo,aux1);
EditDistX.text:=inttostr(aux1);
{abre DistY}
read(arquivo,aux1);
EditDistY.text:=inttostr(aux1);
{abre JT}
read(arquivo,aux1);
EditJT.text:=inttostr(aux1);
{abre numero de viagens}
read(arquivo,aux1);
EditNumV.text:=inttostr(aux1);
{abre Custo de Veiculos}
read(arquivo,aux1);
EditCustVeic.text:=inttostr(aux1);
{abre numero de garagens}
read(arquivo,aux1);
EditNumGarag.text:=inttostr(aux1);
ButtonAux2.Click;
{abre grade de horários}
for j:=1 to NumViagens do
  begin
  read(arquivo,aux1);
  matrizhorar[0,j-1]:=aux1;
  read(arquivo,aux1);
  matrizhorar[1,j-1]:=aux1;
  read(arquivo,aux1);
  matrizhorar[2,j-1]:=aux1;
  read(arquivo,aux1);
  matrizhorar[3,j-1]:=aux1;
  read(arquivo,aux1);
  matrizhorar[4,j-1]:=aux1;
  read(arquivo,aux1);
  matrizhorar[5,j-1]:=aux1;
  read(arquivo,aux1);
  matrizhorar[6,j-1]:=aux1;
  end;
{abre posição das garagens}
for j:=1 to Numgarag do
  begin
```

```
  read(arquivo,aux1);
  matrizgarag[0,j-1]:=aux1;
  read(arquivo,aux1);
  matrizgarag[1,j-1]:=aux1;
  end;
closefile(arquivo);
if NumGarag<=1 then EditFileName.text:='arquivo com apenas 1
garagem'
  else
  begin
  ButtonAux4.Click;
  ButtonEspEst.Enabled:=true;
  ButtonGerarLMo.Enabled:=true;
  ButtonRspLingo.Enabled:=true;
  ButtonViagCom.Enabled:=true;
  end;
end;

{Define tamanho das matrizes dinâmicas}
procedure TFormDoutVsp.ButtonAux2Click(Sender: TObject);
begin
NumViagens:=strtoint(EditNumV.Text);
NumGarag:=strtoint(EditNumGarag.Text);
CustoVeic:=strtoint(EditCustVeic.Text);
setlength(matrizhorar,7,NumViagens);
setlength(matrizviagexp,NumViagens,NumViagens);
setlength(matrizviagexpgaragini,NumViagens,NumGarag);
setlength(matrizviagexpgaragfim,NumViagens,NumGarag);
setlength(matrizocios,NumViagens,NumViagens);
setlength(matrizft,NumViagens,NumViagens);
setlength(matrizgarag,2,NumGarag);
setleng-
th(matrizaux_mdvsp,NumViagens,NumViagens,NumGarag);
setlength(matrizgaragesc,NumViagens,NumViagens);
end;

{calcula matrizes de custo e tempo de viagens}
procedure TFormDoutVsp.ButtonAux4Click(Sender: TObject);
var
i,j,l: integer;
begin
{calcula matriz de tempos de viagem expressa entre viagens}
for j:=1 to NumViagens do
  for i:=1 to NumViagens do
    begin
    aux1:=matrizhorar[2,i-1]-matrizhorar[0,j-1];
    aux1:=aux1*aux1;
    aux2:=matrizhorar[3,i-1]-matrizhorar[1,j-1];
    aux2:=aux2*aux2;
    matrizviagexp[j-1,i-1]:=trunc(sqrt(aux1+aux2));
    end;

{calcula matriz de tempos de ociosidade}
for j:=1 to NumViagens do
  for i:=1 to NumViagens do
    begin
    aux1:=matrizhorar[4,i-1];
    aux2:=matrizhorar[6,j-1];
    matrizocios[j-1,i-1]:=aux1-aux2-matrizviagexp[j-1,i-1];
    end;

{calcula matriz de tempos de folga total}
for j:=1 to NumViagens do
  for i:=1 to NumViagens do
    if (matrizocios[j-1,i-1]>=0) then
      matrizft[j-1,i-1]:=matrizocios[j-1,i-1]+matrizviagexp[j-1,i-1]
        else matrizft[j-1,i-1]:=0;

{calcula matriz de tempos de viagem expressa do depot até
viagem}
for i:=1 to NumGarag do
  for j:=1 to NumViagens do
    begin
    aux1:=matrizhorar[0,j-1]-matrizgarag[0,i-1];
    aux1:=aux1*aux1;
    aux2:=matrizhorar[1,j-1]-matrizgarag[1,i-1];
    aux2:=aux2*aux2;
    matrizViagExpGaragIni[j-1,i-1]:=trunc(sqrt(aux1+aux2));
    aux1:=matrizhorar[2,j-1]-matrizgarag[0,i-1];
    aux1:=aux1*aux1;
    aux2:=matrizhorar[3,j-1]-matrizgarag[1,i-1];
    aux2:=aux2*aux2;
    matrizViagExpGaragFim[j-1,i-1]:=trunc(sqrt(aux1+aux2));
    end;

{calcula tempo de ida e retorno das garagens para o MDVSP -
Hamilton}
for l:=1 to NumGarag do
  for i:=1 to NumViagens do
    for j:=1 to NumViagens do
      if (i=j) or (matrizft[j-1,i-1]>0) then
        matrizaux_mdvsp[i-1,j-1,l-1]:=matrizViagExpGaragFim[i-1,l-
1]+matrizViagExpGaragIni[j-1,l-1]+CustoVeic
        else matrizaux_mdvsp[i-1,j-1,l-1]:=0;

{verifica qual retorno hamiltoniano tem duracao mais curta e
guarda informação na matriz de escolha da garagem}
for i:=1 to NumViagens do
  for j:=1 to NumViagens do
    if (matrizaux_mdvsp[i-1,j-1,0]>0) then
      for l:=1 to NumGarag-1 do
        if (matrizaux_mdvsp[i-1,j-1,l-1]<=matrizaux_mdvsp[i-1,j-
1,l]) then
          matrizgaragesc[i-1,j-1]:=l else matrizgaragesc[i-1,j-
1]:=l+1;

end;

end.
```

APÊNDICE C – Procedimento de redução 1

```
unit dout_red;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs,
  StdCtrls, Grids;

type
  TFormRed = class(TForm)
    Button1: TButton;
    Label2: TLabel;
    StringGrid1: TStringGrid;
    ButtonAux1: TButton;
    Label1: TLabel;
    EditFR: TEdit;
    Label3: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure ButtonAux1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormRed: TFormRed;
  matrizordem, matrizreg1, matrizreg2: array of array of integer;
  reduce, indice: array of integer;
  raiz, fr, n_reduc, aux3: integer;

implementation

uses dout_vsp;

{$R *.DFM}

procedure TFormRed.Button1Click(Sender: TObject);
var
i,j: integer;
begin
stringgrid1.colcount:=NumViagens+1;
stringgrid1.rowcount:=NumViagens+1;
raiz:=trunc(sqrt(NumViagens))+1;
fr:=strtoint(EditFR.text);
n_reduc:=raiz*fr;
setlength(indice,n_reduc);
setlength(reduce,n_reduc);
setlength(matrizordem,NumViagens,NumViagens);
setlength(matrizreg1,NumViagens,NumViagens);
setlength(matrizreg2,NumViagens,NumViagens);

{inicia operacao de reducao das linhas da matriz}
{reinicia matrizes que serao trabalhadas}
aux3:=0;
for i:=1 to NumViagens do
  begin
  stringgrid1.cells[0,i]:=inttostr(i);
  Stringgrid1.cells[i,0]:=inttostr(i);
  end;
for j:=1 to n_reduc do
  begin
  indice[j-1]:=99999;
  reduce[j-1]:=99999;
  end;
for j:=1 to NumViagens do
  for i:=1 to NumViagens do
    begin
    matrizordem[j-1,i-1]:=matrizft[j-1,i-1];
    matrizreg1[j-1,i-1]:=0;
    matrizreg2[j-1,i-1]:=0;
    end;

ButtonAux1.Click;

{Numera matriz final}
for i:=1 to NumViagens do
  begin
  stringgrid1.cells[0,i]:=inttostr(i);
  Stringgrid1.cells[i,0]:=inttostr(i);
  end;

{Exibe valores da matriz final}
aux1:=0;
for j:=1 to NumViagens do
  for i:=1 to NumViagens do
    begin
    matrizreg1[j-1,i-1]:=matrizreg1[j-1,i-1]+matrizreg2[i-1,j-1];
    stringgrid1.cells[i,j]:=inttostr(matrizreg1[j-1,i-1]);
    if matrizreg1[j-1,i-1]>=1 then inc(aux1);
    end;
label1.caption:='Número de variáveis: '+inttostr(aux1);
label2.caption:='Raiz do número de viagens: '+inttostr(raiz);
end;

procedure TFormRed.ButtonAux1Click(Sender: TObject);
Var
i,j,l: integer;
begin
{calcula e faz a reducao nas linhas da matriz}
for j:=1 to NumViagens do
  begin
  for i:=1 to NumViagens do
    begin
    if ((matrizordem[j-1,i-1]<>0) and (matrizordem[j-1,i-
1]<=reduce[n_reduc-1])) then
      begin
      {verifica qual a posicao onde o novo valor sera inserido}
      for l:=1 to n_reduc do if matrizordem[j-1,i-
1]<=reduce[n_reduc-l] then aux3:=n_reduc-l;
      {faz com que todos os demais valores pulem uma casa}
      for l:=1 to n_reduc-aux3-1 do reduce[n_reduc-
l]:=reduce[n_reduc-l-1];
      {posicao de insercao assume valor novo}
      reduce[aux3]:=matrizordem[j-1,i-1];
      {faz com que todos os indices pulem uma casa}
      for l:=1 to n_reduc-aux3-1 do indice[n_reduc-
l]:=indice[n_reduc-l-1];
      {posicao de insercao assume valor novo}
      indice[aux3]:=strtoint(stringgrid1.cells[0,i]);
      end;
    end;
  for l:=1 to n_reduc do
    if indice[l-1]<>99999 then matrizreg1[j-1,indice[l-1]-1]:=1;
  for l:=1 to n_reduc do
    begin
    indice[l-1]:=99999;
    reduce[l-1]:=99999;
    end;
  end;

aux3:=0;
{calcula e faz a reducao nas colunas da matriz}
for i:=1 to NumViagens do
  begin
  for j:=1 to NumViagens do
    begin
    if ((matrizordem[j-1,i-1]<>0) and (matrizordem[j-1,i-
1]<=reduce[n_reduc-1])) then
      begin
      {verifica qual a posicao onde o novo valor sera inserido}
      for l:=1 to n_reduc do if matrizordem[j-1,i-
1]<=reduce[n_reduc-l] then aux3:=n_reduc-l;
      {faz com que todos os demais valores pulem uma casa}
      for l:=1 to n_reduc-aux3-1 do reduce[n_reduc-
l]:=reduce[n_reduc-l-1];
      {posicao de insercao assume valor novo}
      reduce[aux3]:=matrizordem[j-1,i-1];
      {faz com que todos os indices pulem uma casa}
      for l:=1 to n_reduc-aux3-1 do indice[n_reduc-
l]:=indice[n_reduc-l-1];
      {posicao de insercao assume valor novo}
      indice[aux3]:=strtoint(stringgrid1.cells[j,0]);
      end;
```

```
      end;
   for l:=1 to n_reduc do
      if indice[l-1]<>99999 then matrizreg2[i-1,indice[l-1]-1]:=1;
   for l:=1 to n_reduc do
      begin
      indice[l-1]:=99999;
      reduce[l-1]:=99999;
      end;
   end;

end;

end.
```

APÊNDICE D – Procedimento de redução 2

```
unit dout_rsp_vsp;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs,
  StdCtrls, Grids, ExtCtrls;

type
  TFormRspLingo = class(TForm)
    Memo1: TMemo;
    Button1: TButton;
    StringGrid1: TStringGrid;
    Label1: TLabel;
    Bevel1: TBevel;
    Label2: TLabel;
    Memo2: TMemo;
    Label3: TLabel;
    Edit1: TEdit;
    Label4: TLabel;
    buttonauxiliar: TButton;
    ButtonTodas: TButton;
    Memo3: TMemo;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure buttonauxiliarClick(Sender: TObject);
    procedure ButtonTodasClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormRspLingo: TFormRspLingo;
  s, t, u, de, para, g, viag_fim, viag_ini, viagens_ini, viagens_fim:
string;
  matriz_rsp_ini, matriz_rsp_fim, matriz_rsp_centro: array of array
of integer;
  pronto: boolean;
  seq_para: integer;

implementation

uses dout_vsp;

{$R *.DFM}

procedure TFormRspLingo.Button1Click(Sender: TObject);
var
i,j:integer;

begin
for i:=0 to 4 do
   for j:=1 to stringgrid1.rowcount-1 do stringgrid1.cells[i,j]:='';
stringgrid1.rowcount:=2;
viag_ini:='';
viag_fim:='';
viagens_ini:='';
viagens_fim:='';
memo2.lines.clear;
nomedoarquivo:=Edit1.text;
Memo1.Lines.LoadFromFile(nomedoarquivo+'.lgr');
buttonauxiliar.click;
memo2.lines.add(viagens_ini+viagens_fim);
memo3.lines.clear;
memo3.lines.add(viagens_ini+viagens_fim);
memo3.lines.SaveToFile(nomedoarquivo+'.vif');
end;

procedure TFormRspLingo.FormCreate(Sender: TObject);
begin
stringgrid1.cells[0,0]:='Variável';
stringgrid1.cells[1,0]:='De';
stringgrid1.cells[2,0]:='Para';
stringgrid1.cells[3,0]:='Garagem';
stringgrid1.cells[4,0]:='Valor';
end;

procedure TFormRspLingo.buttonauxiliarClick(Sender: TObject);
var
i,j,l: integer;

begin
{construindo a solucao na grade}
stringgrid1.Visible:=false;
i:=5;
repeat
t:=''; {t guarda a variavel}
u:=''; {u guarda o valor}
s:=memo1.lines[i];
for j:=1 to length(s)-20 do
  begin
  if (j<=35) and (s[j]<>' ') then t:=t+s[j];
  if (j>35) and (s[j]<>' ') then u:=u+s[j];
  end;
if t='V' then inc(i) else
begin
if length(u)<>0 then
  begin
  if u[1]='1' then
    begin
    stringgrid1.cells[0,stringgrid1.rowcount-1]:=t;
    stringgrid1.cells[4,stringgrid1.rowcount-1]:=u;
    l:=1;
    de:='';
    para:='';
    repeat
      if (t[l]<>'X') then de:=de+t[l];
      inc(l);
    until t[l]='T';
    repeat
      if (t[l]<>'T') then para:=para+t[l];
      inc(l);
    until t[l]='D';
    stringgrid1.cells[1,stringgrid1.rowcount-1]:=de;
    stringgrid1.cells[2,stringgrid1.rowcount-1]:=para;
    stringgrid1.cells[3,stringgrid1.rowcount-1]:=t[l+1];
    stringgrid1.rowcount:=stringgrid1.rowcount+1;
    end;
  inc(i);
  end;
end;
until (length(u)=0) {or (length(t)<=3)};
stringgrid1.rowcount:=stringgrid1.rowcount-1;

{passando a solucao da grade para as matrizes de respostas}
aux1:=1;
for i:=1 to stringgrid1.rowcount-1 do
  if stringgrid1.cells[1,i]='0' then
    begin
    setlength(matriz_rsp_ini,3,aux1);
    for j:=1 to 3 do matriz_rsp_ini[j-1,aux1-
1]:=strtoint(stringgrid1.cells[j,i]);
    viag_ini:=viag_ini+stringgrid1.cells[2,i]+' ';
    inc(aux1);
    end;

aux2:=stringgrid1.rowcount-aux1;
setlength(matriz_rsp_centro,3,aux2);
for i:=1 to stringgrid1.rowcount-1 do
  if (stringgrid1.cells[1,i]<>'0') then
    begin
    aux2:=strtoint(stringgrid1.cells[1,i]);
    for j:=1 to 3 do matriz_rsp_centro[j-1,aux2-
1]:=strtoint(stringgrid1.cells[j,i]);
    if (stringgrid1.cells[2,i]='0') then vi-
ag_fim:=viag_fim+stringgrid1.cells[1,i]+' ';
    end;

viagens_ini:=viagens_ini+viag_ini;
viagens_fim:=viagens_fim+viag_fim;
```

```
{Montando sequencias das viagens}
for i:=1 to aux1-1 do
  begin
  pronto:=false;
  seq_para:=matriz_rsp_ini[1,i-1];
  s:='';
  s:='D'+inttostr(matriz_rsp_ini[2,i-1])+' - '+inttostr(seq_para);
  repeat
  if matriz_rsp_centro[1,seq_para-1]=0 then
    begin
    pronto:=true;
    s:=s+' - D'+inttostr(matriz_rsp_centro[2,seq_para-1]);
    end
      else
    begin
    s:=s+' - '+inttostr(matriz_rsp_centro[1,seq_para-1]);
    seq_para:=matriz_rsp_centro[1,seq_para-1];
    end;
  until pronto;
  memo2.lines.add(s);
  end;

stringgrid1.Visible:=true;
end;

procedure TFormRspLingo.ButtonTodasClick(Sender: TObject);
var
i,j,l: integer;

begin
memo2.lines.clear;
viagens_ini:='';
viagens_fim:='';
for l:=1 to NumGarag do
  begin
  for i:=0 to 4 do
    for j:=1 to stringgrid1.rowcount-1 do stringgrid1.cells[i,j]:='';
  stringgrid1.rowcount:=2;
  viag_ini:='';
  viag_fim:='';
  nomedoarquivo:=Edit1.text;
  Me-
mo1.Lines.LoadFromFile(nomedoarquivo+'_sd'+inttostr(l)+'_red.lgr'
);
  buttonauxiliar.click;
  end;
memo2.lines.add(viagens_ini+viagens_fim);
memo3.lines.clear;
memo3.lines.add(viagens_ini);
memo3.lines.SaveToFile(nomedoarquivo+'.vgi');
memo3.lines.clear;
memo3.lines.add(viagens_fim);
memo3.lines.SaveToFile(nomedoarquivo+'.vgf');
end;

procedure TFormRspLingo.FormActivate(Sender: TObject);
begin
edit1.text:=nomedoarquivo;
end;

end.
```

APÊNDICE E – Procedimento de redução 3

```
unit dout_viagcom;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs,
  StdCtrls, Grids, ExtCtrls;

type
  TFormViagCom = class(TForm)
    Memo1: TMemo;
    StringGrid1: TStringGrid;
    Label1: TLabel;
    Bevel1: TBevel;
    Label2: TLabel;
    Memo2: TMemo;
    Label3: TLabel;
    Edit1: TEdit;
    Label4: TLabel;
    buttonauxiliar: TButton;
    ButtonTodas: TButton;
    Memo3: TMemo;
    procedure FormCreate(Sender: TObject);
    procedure buttonauxiliarClick(Sender: TObject);
    procedure ButtonTodasClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormViagCom: TFormViagCom;
  s, t, u, de, para: string;
  viagenscomuns: array of array of integer;
  matriz_rsp: array of array of array of integer;
  count: array of integer;

implementation

uses dout_vsp;

{$R *.DFM}

procedure TFormViagCom.FormCreate(Sender: TObject);
begin
stringgrid1.cells[0,0]:='Var';
end;

procedure TFormViagCom.FormActivate(Sender: TObject);
begin
stringgrid1.colcount:=4+NumGarag;
edit1.text:=nomedoarquivo;
end;

procedure TFormViagCom.buttonauxiliarClick(Sender: TObject);
var
j,l: integer;

begin
{construindo a solucao e armazenado na matriz}
aux1:=0;
repeat
t:=''; {t guarda a variavel}
u:=''; {u guarda o valor}
s:=memo1.lines[aux1+5];
for j:=1 to length(s)-20 do
   begin
   if (j<=35) and (s[j]<>' ') then t:=t+s[j];
   if (j>35) and (s[j]<>' ') then u:=u+s[j];
   end;
if (length(u)<>0) and (t<>'V') then
   begin
   inc(aux1);
   l:=1;
   de:='';
   para:='';
   repeat
      if (t[l]<>'X') then de:=de+t[l];
      inc(l);
   until t[l]='T';
   repeat
      if (t[l]<>'T') then para:=para+t[l];
      inc(l);
   until t[l]='D';
   if (de<>'0') and (para<>'0') then matriz_rsp[strtoint(de)-
1,strtoint(para)-1,aux2-1]:=strtoint(u[1]);
   end;
until (length(u)=0) or (length(t)<=3);
end;

procedure TFormViagCom.ButtonTodasClick(Sender: TObject);
var
i,j,l,k,m: integer;

begin
memo2.lines.clear;
for i:=0 to 2+NumGarag do
   for j:=1 to stringgrid1.rowcount-1 do stringgrid1.cells[i,j]:='';
stringgrid1.rowcount:=2;
setlength(matriz_rsp,numviagens,numviagens,numgarag+1);{+1 =
para o somatorio}
setlength(viagenscomuns,numviagens,numviagens);
setlength(count,numgarag+1);

{reinicializando variaveis da matriz de resposta}
for i:=1 to numviagens do
   for j:=1 to numviagens do
      for l:=1 to numgarag+2 do
         matriz_rsp[i-1,j-1,l-1]:=0;

{le resposta de cada vsp}
for l:=1 to NumGarag do
   begin
   Memo1.Lines.clear;
   Me-
mo1.Lines.LoadFromFile(nomedoarquivo+'_sd'+inttostr(l)+'_red.lgr'
);
   aux2:=l;
   buttonauxiliar.click;
   end;

{le resposta do hamiltoniano}
{Memo1.Lines.clear;
Memo1.Lines.LoadFromFile(Nomedoarquivo+'_md_cf_red.lgr');
aux2:=Numgarag+1;
buttonauxiliar.click;}

memo2.visible:=false;
{calcula o somatorio das respostas}
k:=0;
m:=0;
for i:=1 to Numviagens do
   for j:=1 to Numviagens do
      begin
      aux2:=0;
      for l:=1 to Numgarag do
         if matrizocios[i-1,j-1]>=0 then aux2:=aux2+matriz_rsp[i-1,j-1,l-
1];
      matriz_rsp[i-1,j-1,Numgarag]:=aux2;
      if  matrizocios[i-1,j-1]>=0 then inc(count[aux2]);
      if  (aux2=0) and (matrizocios[i-1,j-1]>=0) then
         begin
         viagenscomuns[i-1,j-1]:=1;
         inc(k);
         end;
      if  (aux2=Numgarag) and (matrizocios[i-1,j-1]>=0) then
         begin
         viagenscomuns[i-1,j-1]:=2;
         inc(k);
         inc(m);
         end;
      end;
```

```
aux1:=0;
for i:=1 to Numviagens do
  for j:=1 to Numviagens do
    if matrizocios[i-1,j-1]>=0 then inc(aux1);

memo2.lines.add('total de variáveis: '+inttostr(aux1));
memo2.lines.add('total de variáveis com valor igual a zero ou
(garagens+1): '+inttostr(k));
memo2.lines.add('total de viagens cobertas: '+inttostr(m));

for i:=0 to Numgarag do
  memo2.lines.add('viagens '+inttostr(i)+': '+inttostr(count[i]));

u:=nomedoarquivo+'.vco';
AssignFile(arquivo,u);
rewrite(arquivo);
reset(arquivo);
for i:=1 to numviagens do
  for j:=1 to numviagens do
    begin
    if viagenscomuns[i-1,j-1]=1 then k:=1 else
      if viagenscomuns[i-1,j-1]=2 then k:=2 else k:=0;
    write(arquivo,k);
    end;
closefile(arquivo);
Memo2.Lines.add('arquivo gravado');

stringgrid1.Visible:=false;
stringgrid1.rowcount:=Numviagens+1;
stringgrid1.colcount:=Numviagens+1;
aux1:=0;
for i:=1 to numviagens do
  for j:=1 to numviagens do
    if matrizocios[i-1,j-1]>0 then
      begin
      stringgrid1.cells[j,i]:=inttostr(matriz_rsp[i-1,j-1,Numgarag]);
      inc(aux1);
      end
        else
      begin
      stringgrid1.cells[j,i]:='';
      inc(aux1);
      end;

for i:=1 to numviagens do
  begin
  stringgrid1.cells[0,i]:=inttostr(i);
  stringgrid1.cells[i,0]:=inttostr(i);
  end;

memo2.visible:=true;
stringgrid1.Visible:=true;

end;

end.
```

APÊNDICE F – Modelo linear VSP

```
unit dout_lmo_vsp;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs,
  StdCtrls, ExtCtrls;

type
  TFormLinearModelVsp = class(TForm)
    Memo2: TMemo;
    Label2: TLabel;
    ButtonLinearModel: TButton;
    Bevel1: TBevel;
    ButtonSalvarLMo: TButton;
    CheckBoxPI: TCheckBox;
    ComboBoxLinModel: TComboBox;
    Edit1: TEdit;
    Label1: TLabel;
    procedure ButtonLinearModelClick(Sender: TObject);
    procedure ButtonSalvarLMoClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormLinearModelVsp: TFormLinearModelVsp;
  s,t,p1,p2,p3,p4,p5,p6,p7:string;

implementation

uses dout_vsp, dout_red;

{$R *.DFM}

procedure TFormLinearModelVsp.ButtonLinearModelClick(Sender:
TObject);
var
i,j,aux3: integer;

procedure quebradelinha;
begin
aux3:=aux3+1;
if aux3>=10 then
   begin
   memo2.lines.add(s);
   aux3:=0;
   s:='';
   end;
end;

begin
{Função objetivo}
s:='';
aux1:=0;
aux2:=0;
aux3:=0;
memo2.Visible:=false;
memo2.lines.clear;
ButtonSalvarLMo.Enabled:=true;
memo2.lines.add('MIN =');
aux2:=strtoint(edit1.text);
if (aux2<=0) or (aux2>NumGarag) then
   begin
   aux2:=1;
   edit1.text:='1';
   end;

{VSP SEM REDUÇÃO}
if ComboBoxLinModel.text='VSP' then
begin
{viagem expressa originárias da garagem}
for i:=1 to NumViagens do
  begin
  s:=s+' + '+inttostr(matrizViagExpGaragIni[i-1,aux2-1])+' *
X0T'+inttostr(i)+'D'+inttostr(aux2);
  aux1:=aux1+1;
  quebradelinha;
  end;
if aux3>=1 then memo2.lines.add(s);
s:='';

{custo total (FT) entre viagens}
for j:=1 to NumViagens do
   begin
   s:='';
   aux3:=0;
   for i:=1 to NumViagens do
      if (matrizocios[j-1,i-1]>=0) then
         begin
         s:=s+' + '+inttostr(matrizft[j-1,i-1])+' *
X'+inttostr(j)+'T'+inttostr(i)+'D'+inttostr(aux2);
         aux1:=aux1+1;
         quebradelinha;
         end;
   if s<>'' then memo2.lines.add(s);
   end;
s:='';

{viagem expressa retorno para a garagem}
aux3:=0;
for i:=1 to NumViagens do
  begin
  s:=s+' + '+inttostr(matrizViagExpGaragFim[i-1,aux2-1])+' *
X'+inttostr(i)+'T0D'+inttostr(aux2);
  aux1:=aux1+1;
  quebradelinha;
  end;
if s<>'' then memo2.lines.add(s);
memo2.lines.add(' + '+inttostr(CustoVeic)+' * V;');
s:='';

{Restrição de chegada}
for i:=1 to NumViagens do
   begin
   s:='';
   aux3:=0;
   for j:=1 to NumViagens do
      if (matrizocios[j-1,i-1]>=0) then
         begin
         s:=s+' + X'+inttostr(j)+'T'+inttostr(i)+'D'+inttostr(aux2);
         quebradelinha;
         end;
   s:=s+' + X0T'+inttostr(i)+'D'+inttostr(aux2)+' = 1;';
   memo2.lines.add(s);
   end;
s:='';

{Restrição de saída}
for j:=1 to NumViagens do
   begin
   s:='';
   aux3:=0;
   for i:=1 to NumViagens do
      if (matrizocios[j-1,i-1]>=0) then
         begin
         s:=s+' + X'+inttostr(j)+'T'+inttostr(i)+'D'+inttostr(aux2);
         quebradelinha;
         end;
   s:=s+' + X'+inttostr(j)+'T0D'+inttostr(aux2)+' = 1;';
   memo2.lines.add(s);
   end;
s:='';

{Restrição de saída da garagem}
s:='';
aux3:=0;
for i:=1 to NumViagens do
   begin
   s:=s+' + X0T'+inttostr(i)+'D'+inttostr(aux2);
   quebradelinha;
   end;
```

```
memo2.lines.add(s+' - V = 0;');
s:='';

aux3:=0;
for i:=1 to NumViagens do
  begin
  s:=s+' + X'+inttostr(i)+'T0D'+inttostr(aux2);
  quebradelinha;
  end;
memo2.lines.add(s+' - V = 0;');
s:='';

{insere variáveis binárias}
if checkboxpi.checked then
begin
for i:=1 to NumViagens do
  for j:=1 to NumViagens do
    if (matrizocios[i-1,j-1]>=0) then
      me-
mo2.lines.add('@BIN(X'+inttostr(j)+'T'+inttostr(i)+'D'+inttostr(aux2)
+');');
memo2.lines.add('!numero de variaveis='+inttostr(aux1+1));
s:='';
end;

end;

{VSP COM REDUÇÃO}
if ComboBoxLinModel.text='VSP - REDUZIDO' then
begin
{viagem expressa originárias da garagem}
for i:=1 to NumViagens do
 begin
 s:=s+' + '+inttostr(matrizViagExpGaragIni[i-1,aux2-1])+' *
X0T'+inttostr(i)+'D'+inttostr(aux2);
 aux1:=aux1+1;
 quebradelinha;
 end;
if aux3>=1 then memo2.lines.add(s);
s:='';

{custo total (FT) entre viagens}
for j:=1 to NumViagens do
  begin
  s:='';
  aux3:=0;
  for i:=1 to NumViagens do
    if (matrizreg1[j-1,i-1]>=1) then
      begin
      s:=s+' + '+inttostr(matrizft[j-1,i-1])+' *
X'+inttostr(j)+'T'+inttostr(i)+'D'+inttostr(aux2);
      aux1:=aux1+1;
      quebradelinha;
      end;
  if s<>'' then memo2.lines.add(s);
  end;
s:='';

{viagem expressa retorno para a garagem}
aux3:=0;
for i:=1 to NumViagens do
 begin
 s:=s+' + '+inttostr(matrizViagExpGaragFim[i-1,aux2-1])+' *
X'+inttostr(i)+'T0D'+inttostr(aux2);
 aux1:=aux1+1;
 quebradelinha;
 end;
if s<>'' then memo2.lines.add(s);
memo2.lines.add(' + '+inttostr(CustoVeic)+' * V;');
s:='';

{Restrição de chegada}
for i:=1 to NumViagens do
  begin
  s:='';
  aux3:=0;
  for j:=1 to NumViagens do
    if (matrizreg1[j-1,i-1]>=1) then
      begin
      s:=s+' + X'+inttostr(j)+'T'+inttostr(i)+'D'+inttostr(aux2);
      quebradelinha;
      end;
  s:=s+' + X0T'+inttostr(i)+'D'+inttostr(aux2)+' = 1;';
  memo2.lines.add(s);
  end;
s:='';

{Restrição de saída}
for j:=1 to NumViagens do
  begin
  s:='';
  aux3:=0;
  for i:=1 to NumViagens do
    if (matrizreg1[j-1,i-1]>=1) then
      begin
      s:=s+' + X'+inttostr(j)+'T'+inttostr(i)+'D'+inttostr(aux2);
      quebradelinha;
      end;
  s:=s+' + X'+inttostr(j)+'T0D'+inttostr(aux2)+' = 1;';
  memo2.lines.add(s);
  end;
s:='';

{Restrição de saída da garagem}
s:='';
aux3:=0;
for i:=1 to NumViagens do
  begin
  s:=s+' + X0T'+inttostr(i)+'D'+inttostr(aux2);
  quebradelinha;
  end;
memo2.lines.add(s+' - V = 0;');
s:='';

aux3:=0;
for i:=1 to NumViagens do
  begin
  s:=s+' + X'+inttostr(i)+'T0D'+inttostr(aux2);
  quebradelinha;
  end;
memo2.lines.add(s+' - V = 0;');
s:='';

{insere variáveis binárias}
if checkboxpi.checked then
begin
for i:=1 to NumViagens do
  for j:=1 to NumViagens do
    if (matrizreg1[j-1,i-1]>=1) then
      me-
mo2.lines.add('@BIN(X'+inttostr(j)+'T'+inttostr(i)+'D'+inttostr(aux2)
+');');
memo2.lines.add('!numero de variaveis='+inttostr(aux1+1));
s:='';
end;

end;

memo2.Visible:=true;
end;

procedure TFormLinearModelVsp.ButtonSalvarLMoClick(Sender:
TObject);
begin
if ComboBoxLinModel.text='VSP' then me-
mo2.Lines.SaveToFile(nomedoarquivo+'_sd'+inttostr(aux2)+'.lg4');
if ComboBoxLinModel.text='VSP - REDUZIDO' then me-
mo2.Lines.SaveToFile(nomedoarquivo+'_sd'+inttostr(aux2)+'_red.l
g4');
ButtonSalvarLMo.Enabled:=false;
end;

end.
```

APÊNDICE G – Modelo linear vsp para a garagem 1

MIN = + 63 * X0T1D1 + 57 * X0T2D1 + 31 * X0T3D1 + 11 * X0T4D1 + 40 * X0T5D1 + 34 * X0T6D1 + 60 * X0T7D1 + 67 * X0T8D1 + 55 * X0T9D1 + 14 * X0T10D1 + 59 * X0T11D1 + 36 * X0T12D1 + 10 * X0T13D1 + 31 * X0T14D1 + 40 * X0T15D1 + 88 * X1T7D1 + 143 * X1T11D1 + 67 * X2T7D1 + 48 * X2T10D1 + 122 * X2T11D1 + 85 * X3T2D1 + 55 * X3T4D1 + 40 * X3T6D1 + 46 * X3T12D1 + 122 * X4T7D1 + 68 * X4T9D1 + 103 * X4T10D1 + 43 * X4T13D1 + 146 * X5T7D1 + 92 * X5T9D1 + 127 * X5T10D1 + 67 * X5T13D1 + 19 * X6T2D1 + 69 * X6T9D1 + 104 * X6T10D1 + 44 * X6T13D1 + 51 * X8T1D1 + 97 * X8T2D1 + 67 * X8T4D1 + 58 * X8T12D1 + 28 * X8T15D1 + 82 * X9T11D1 + 106 * X12T7D1 + 87 * X12T10D1 + 161 * X12T11D1 + 27 * X12T13D1 + 104 * X13T11D1 + 82 * X14T2D1 + 52 * X14T4D1 + 37 * X14T6D1 + 43 * X14T12D1 + 155 * X15T7D1 + 101 * X15T9D1 + 136 * X15T10D1 + 76 * X15T13D1 + 72 * X1T0D1 + 39 * X2T0D1 + 7 * X3T0D1 + 18 * X4T0D1 + 59 * X5T0D1 + 9 * X6T0D1 + 68 * X7T0D1 + 53 * X8T0D1 + 69 * X9T0D1 + 37 * X10T0D1 + 45 * X11T0D1 + 18 * X12T0D1 + 25 * X13T0D1 + 56 * X14T0D1 + 46 * X15T0D1 + 500 * V;
+ X8T1D1 + X0T1D1 = 1;
+ X3T2D1 + X6T2D1 + X8T2D1 + X14T2D1 + X0T2D1 = 1;
+ X0T3D1 = 1;
+ X3T4D1 + X8T4D1 + X14T4D1 + X0T4D1 = 1;
+ X0T5D1 = 1;
+ X3T6D1 + X14T6D1 + X0T6D1 = 1;
+ X1T7D1 + X2T7D1 + X4T7D1 + X5T7D1 + X12T7D1 + X15T7D1 + X0T7D1 = 1;
+ X0T8D1 = 1;
+ X4T9D1 + X5T9D1 + X6T9D1 + X15T9D1 + X0T9D1 = 1;
+ X2T10D1 + X4T10D1 + X5T10D1 + X6T10D1 + X12T10D1 + X15T10D1 + X0T10D1 = 1;
+ X1T11D1 + X2T11D1 + X9T11D1 + X12T11D1 + X13T11D1 + X0T11D1 = 1;
+ X3T12D1 + X8T12D1 + X14T12D1 + X0T12D1 = 1;
+ X4T13D1 + X5T13D1 + X6T13D1 + X12T13D1 + X15T13D1 + X0T13D1 = 1;
+ X0T14D1 = 1;
+ X8T15D1 + X0T15D1 = 1;
+ X1T7D1 + X1T11D1 + X1T0D1 = 1;
+ X2T7D1 + X2T10D1 + X2T11D1 + X2T0D1 = 1;
+ X3T2D1 + X3T4D1 + X3T6D1 + X3T12D1 + X3T0D1 = 1;
+ X4T7D1 + X4T9D1 + X4T10D1 + X4T13D1 + X4T0D1 = 1;
+ X5T7D1 + X5T9D1 + X5T10D1 + X5T13D1 + X5T0D1 = 1;
+ X6T2D1 + X6T9D1 + X6T10D1 + X6T13D1 + X6T0D1 = 1;
+ X7T0D1 = 1;
+ X8T1D1 + X8T2D1 + X8T4D1 + X8T12D1 + X8T15D1 + X8T0D1 = 1;
+ X9T11D1 + X9T0D1 = 1;
+ X10T0D1 = 1;
+ X11T0D1 = 1;
+ X12T7D1 + X12T10D1 + X12T11D1 + X12T13D1 + X12T0D1 = 1;
+ X13T11D1 + X13T0D1 = 1;
+ X14T2D1 + X14T4D1 + X14T6D1 + X14T12D1 + X14T0D1 = 1;
+ X15T7D1 + X15T9D1 + X15T10D1 + X15T13D1 + X15T0D1 = 1;
+ X0T1D1 + X0T2D1 + X0T3D1 + X0T4D1 + X0T5D1 + X0T6D1 + X0T7D1 + X0T8D1 + X0T9D1 + X0T10D1 + X0T11D1 + X0T12D1 + X0T13D1 + X0T14D1 + X0T15D1 - V = 0;
+ X1T0D1 + X2T0D1 + X3T0D1 + X4T0D1 + X5T0D1 + X6T0D1 + X7T0D1 + X8T0D1 + X9T0D1 + X10T0D1 + X11T0D1 + X12T0D1 + X13T0D1 + X14T0D1 + X15T0D1 - V = 0;

APÊNDICE H – Modelo linear vsp para a garagem 2

MIN = + 68 * X0T1D2 + 21 * X0T2D2 + 51 * X0T3D2 + 42 * X0T4D2 + 18 * X0T5D2 + 34 * X0T6D2 + 38 * X0T7D2 + 15 * X0T8D2 + 56 * X0T9D2 + 57 * X0T10D2 + 32 * X0T11D2 + 30 * X0T12D2 + 51 * X0T13D2 + 30 * X0T14D2 + 29 * X0T15D2 + 88 * X1T7D2 + 143 * X1T11D2 + 67 * X2T7D2 + 48 * X2T10D2 + 122 * X2T11D2 + 85 * X3T2D2 + 55 * X3T4D2 + 40 * X3T6D2 + 46 * X3T12D2 + 122 * X4T7D2 + 68 * X4T9D2 + 103 * X4T10D2 + 43 * X4T13D2 + 146 * X5T7D2 + 92 * X5T9D2 + 127 * X5T10D2 + 67 * X5T13D2 + 19 * X6T2D2 + 69 * X6T9D2 + 104 * X6T10D2 + 44 * X6T13D2 + 51 * X8T1D2 + 97 * X8T2D2 + 67 * X8T4D2 + 58 * X8T12D2 + 28 * X8T15D2 + 82 * X9T11D2 + 106 * X12T7D2 + 87 * X12T10D2 + 161 * X12T11D2 + 27 * X12T13D2 + 104 * X13T11D2 + 82 * X14T2D2 + 52 * X14T4D2 + 37 * X14T6D2 + 43 * X14T12D2 + 155 * X15T7D2 + 101 * X15T9D2 + 136 * X15T10D2 + 76 * X15T13D2 + 28 * X1T0D2 + 47 * X2T0D2 + 54 * X3T0D2 + 32 * X4T0D2 + 20 * X5T0D2 + 42 * X6T0D2 + 22 * X7T0D2 + 18 * X8T0D2 + 45 * X9T0D2 + 16 * X10T0D2 + 7 * X11T0D2 + 39 * X12T0D2 + 29 * X13T0D2 + 19 * X14T0D2 + 11 * X15T0D2 + 500 * V;
+ X8T1D2 + X0T1D2 = 1;
+ X3T2D2 + X6T2D2 + X8T2D2 + X14T2D2 + X0T2D2 = 1;
+ X0T3D2 = 1;
+ X3T4D2 + X8T4D2 + X14T4D2 + X0T4D2 = 1;
+ X0T5D2 = 1;
+ X3T6D2 + X14T6D2 + X0T6D2 = 1;
+ X1T7D2 + X2T7D2 + X4T7D2 + X5T7D2 + X12T7D2 + X15T7D2 + X0T7D2 = 1;
+ X0T8D2 = 1;
+ X4T9D2 + X5T9D2 + X6T9D2 + X15T9D2 + X0T9D2 = 1;
+ X2T10D2 + X4T10D2 + X5T10D2 + X6T10D2 + X12T10D2 + X15T10D2 + X0T10D2 = 1;
+ X1T11D2 + X2T11D2 + X9T11D2 + X12T11D2 + X13T11D2 + X0T11D2 = 1;
+ X3T12D2 + X8T12D2 + X14T12D2 + X0T12D2 = 1;
+ X4T13D2 + X5T13D2 + X6T13D2 + X12T13D2 + X15T13D2 + X0T13D2 = 1;
+ X0T14D2 = 1;
+ X8T15D2 + X0T15D2 = 1;
+ X1T7D2 + X1T11D2 + X1T0D2 = 1;
+ X2T7D2 + X2T10D2 + X2T11D2 + X2T0D2 = 1;
+ X3T2D2 + X3T4D2 + X3T6D2 + X3T12D2 + X3T0D2 = 1;
+ X4T7D2 + X4T9D2 + X4T10D2 + X4T13D2 + X4T0D2 = 1;
+ X5T7D2 + X5T9D2 + X5T10D2 + X5T13D2 + X5T0D2 = 1;
+ X6T2D2 + X6T9D2 + X6T10D2 + X6T13D2 + X6T0D2 = 1;
+ X7T0D2 = 1;
+ X8T1D2 + X8T2D2 + X8T4D2 + X8T12D2 + X8T15D2 + X8T0D2 = 1;
+ X9T11D2 + X9T0D2 = 1;
+ X10T0D2 = 1;
+ X11T0D2 = 1;
+ X12T7D2 + X12T10D2 + X12T11D2 + X12T13D2 + X12T0D2 = 1;
+ X13T11D2 + X13T0D2 = 1;
+ X14T2D2 + X14T4D2 + X14T6D2 + X14T12D2 + X14T0D2 = 1;
+ X15T7D2 + X15T9D2 + X15T10D2 + X15T13D2 + X15T0D2 = 1;
+ X0T1D2 + X0T2D2 + X0T3D2 + X0T4D2 + X0T5D2 + X0T6D2 + X0T7D2 + X0T8D2 + X0T9D2 + X0T10D2 + X0T11D2 + X0T12D2 + X0T13D2 + X0T14D2 + X0T15D2 - V = 0;
+ X1T0D2 + X2T0D2 + X3T0D2 + X4T0D2 + X5T0D2 + X6T0D2 + X7T0D2 + X8T0D2 + X9T0D2 + X10T0D2 + X11T0D2 + X12T0D2 + X13T0D2 + X14T0D2 + X15T0D2 - V = 0;

APÊNDICE I – Modelo de redução – procedimentos 1, 2 e 3

```
unit dout_red;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs,
  StdCtrls, Grids;

type
  TFormRed = class(TForm)
    Button1: TButton;
    Label2: TLabel;
    StringGrid1: TStringGrid;
    ButtonAux1: TButton;
    Label1: TLabel;
    EditFR: TEdit;
    Label3: TLabel;
    StringGrid2: TStringGrid;
    Label4: TLabel;
    Memo1: TMemo;
    procedure Button1Click(Sender: TObject);
    procedure ButtonAux1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormRed: TFormRed;
  matrizordem, matrizreg1, matrizreg2: array of array of integer;
  reduce, indice, vgi, vgf: array of integer;
  raiz, fr, n_reduc, aux3: integer;
  u, t: string;

implementation

uses dout_md_padrao;

{$R *.DFM}

procedure TFormRed.Button1Click(Sender: TObject);
var
i,j: integer;
begin
stringgrid1.colcount:=NumViagens+1;
stringgrid1.rowcount:=NumViagens+1;
stringgrid2.colcount:=NumViagens+1;
raiz:=trunc(sqrt(NumViagens))+1;
fr:=strtoint(EditFR.text);
n_reduc:=raiz*fr;
setlength(indice,n_reduc);
setlength(reduce,n_reduc);
setlength(vgi,NumViagens);
setlength(vgf,NumViagens);
setlength(matrizordem,NumViagens,NumViagens);
setlength(matrizreg1,NumViagens,NumViagens);
setlength(matrizreg2,NumViagens,NumViagens);

{inicia operacao de reducao das linhas da matriz}
{reinicia matrizes que serao trabalhadas}
aux3:=0;
for i:=1 to NumViagens do
  begin
  stringgrid1.cells[0,i]:=inttostr(i);
  Stringgrid1.cells[i,0]:=inttostr(i);
  end;
for j:=1 to n_reduc do
  begin
  indice[j-1]:=99999;
  reduce[j-1]:=99999;
  end;
for j:=1 to NumViagens do
  for i:=1 to NumViagens do
    begin
    matrizordem[j-1,i-1]:=matrizft[j-1,i-1];
    matrizreg1[j-1,i-1]:=0;
    matrizreg2[j-1,i-1]:=0;
    end;

ButtonAux1.Click;

{Numera matriz final}
for i:=1 to NumViagens do
  begin
  stringgrid1.cells[0,i]:=inttostr(i);
  Stringgrid1.cells[i,0]:=inttostr(i);
  end;

{Une matrizreg1 e matrizreg2 em uma unica matriz - mtrizreg1}
aux1:=0;
for j:=1 to NumViagens do
  for i:=1 to NumViagens do
    begin
    matrizreg1[j-1,i-1]:=matrizreg1[j-1,i-1]+matrizreg2[i-1,j-1];
    if matrizreg1[j-1,i-1]>=1 then inc(aux1);
    end;
label1.caption:='Número de variáveis: '+inttostr(aux1);
label2.caption:='Raiz do número de viagens: '+inttostr(raiz);

{abre arquivo com viagens comuns e registra em matrizreg1 como
numero 3}
s:=nomedoarquivo+'.vco';
AssignFile(arquivo,s);
reset(arquivo);
{abre  N}
for j:=1 to NumViagens do
  for i:=1 to NumViagens do
    begin
    read(arquivo,aux1);
    if (aux1=1) and (matrizreg1[j-1,i-1]>=1) then matrizreg1[j-1,i-
1]:=3;
    end;
closefile(arquivo);

{Exibe valores da matriz final}
for j:=1 to NumViagens do
  for i:=1 to NumViagens do
    stringgrid1.cells[i,j]:=inttostr(matrizreg1[j-1,i-1]);

end;

procedure TFormRed.ButtonAux1Click(Sender: TObject);
Var
i,j,l: integer;
begin
{calcula e faz a reducao nas linhas da matriz}
for j:=1 to NumViagens do
  begin
  for i:=1 to NumViagens do
    begin
    if ((matrizordem[j-1,i-1]<>0) and (matrizordem[j-1,i-
1]<=reduce[n_reduc-1])) then
      begin
      {verifica qual a posicao onde o novo valor sera inserido}
      for l:=1 to n_reduc do if matrizordem[j-1,i-
1]<=reduce[n_reduc-l] then aux3:=n_reduc-l;
      {faz com que todos os demais valores pulem uma casa}
      for l:=1 to n_reduc-aux3-1 do reduce[n_reduc-
l]:=reduce[n_reduc-l-1];
      {posicao de insercao assume valor novo}
      reduce[aux3]:=matrizordem[j-1,i-1];
      {faz com que todos os indices pulem uma casa}
      for l:=1 to n_reduc-aux3-1 do indice[n_reduc-
l]:=indice[n_reduc-l-1];
      {posicao de insercao assume valor novo}
      indice[aux3]:=strtoint(stringgrid1.cells[0,i]);
      end;
    end;
  for l:=1 to n_reduc do
    if indice[l-1]<>99999 then matrizreg1[j-1,indice[l-1]-1]:=1;
  for l:=1 to n_reduc do
    begin
    indice[l-1]:=99999;
```

```
      reduce[l-1]:=99999;
      end;
   end;

aux3:=0;
{calcula e faz a reducao nas colunas da matriz}
for i:=1 to NumViagens do
   begin
   for j:=1 to NumViagens do
      begin
      if ((matrizordem[j-1,i-1]<>0) and (matrizordem[j-1,i-
1]<=reduce[n_reduc-1])) then
         begin
         {verifica qual a posicao onde o novo valor sera inserido}
         for l:=1 to n_reduc do if matrizordem[j-1,i-
1]<=reduce[n_reduc-l] then aux3:=n_reduc-l;
         {faz com que todos os demais valores pulem uma casa}
         for l:=1 to n_reduc-aux3-1 do reduce[n_reduc-
l]:=reduce[n_reduc-l-1];
         {posicao de insercao assume valor novo}
         reduce[aux3]:=matrizordem[j-1,i-1];
         {faz com que todos os indices pulem uma casa}
         for l:=1 to n_reduc-aux3-1 do indice[n_reduc-
l]:=indice[n_reduc-l-1];
         {posicao de insercao assume valor novo}
         indice[aux3]:=strtoint(stringgrid1.cells[j,0]);
         end;
      end;
   for l:=1 to n_reduc do
      if indice[l-1]<>99999 then matrizreg2[i-1,indice[l-1]-1]:=1;
   for l:=1 to n_reduc do
      begin
      indice[l-1]:=99999;
      reduce[l-1]:=99999;
      end;
   end;

{encontra viagens iniciais no arquivo vgi}
{marca com o valor 1 as viagens iniciais e fianis que devem ser
utilizadas}
stringgrid2.cells[0,1]:='ini';
for i:=1 to NumViagens do stringgrid2.cells[i,0]:=inttostr(i);
for i:=1 to NumViagens do stringgrid2.cells[i,1]:='';
for i:=1 to NumViagens do vgi[i-1]:=0;
memo1.Lines.clear;
memo1.Lines.loadfromfile(nomedoarquivo+'.vgi');
aux3:=-1;
   repeat
   inc(aux3);
   t:=memo1.lines[aux3];
   u:='';
   for i:=1 to length(t) do
      if (t[i]<>' ') then u:=u+t[i] else
         begin
         vgi[strtoint(u)-1]:=1;
         u:='';
         end;
   until length(t)=0;

for i:=1 to NumViagens do
    if vgi[i-1]=1 then
       stringgrid2.cells[i,1]:='1';

{encontra viagens finais no arquivo vgf}
{marca com o valor 1 as viagens iniciais e fianis que devem ser
utilizadas}
stringgrid2.cells[0,2]:='fim';
for i:=1 to NumViagens do stringgrid2.cells[i,2]:='';
for i:=1 to NumViagens do vgf[i-1]:=0;
memo1.Lines.clear;
memo1.Lines.loadfromfile(nomedoarquivo+'.vgf');
aux3:=-1;
   repeat
   inc(aux3);
   t:=memo1.lines[aux3];
   u:='';
   for i:=1 to length(t) do
      if (t[i]<>' ') then u:=u+t[i] else
         begin
         vgf[strtoint(u)-1]:=1;
         u:='';
         end;
   until length(t)=0;

for i:=1 to NumViagens do
    if vgf[i-1]=1 then
       stringgrid2.cells[i,2]:='1';

end;

end.
```

APÊNDICE J – Modelo linear MDVSPcf

```
MIN = + 531 * X0T3D1 + 511 * X0T4D1 + 540 * X0T5D1 + 567 * X0T8D1 + 536 * X0T12D1 + 531 * X0T14D1 + 540 * X0T15D1 + 551 * X0T3D2 +
542 * X0T4D2 + 518 * X0T5D2 + 515 * X0T8D2 + 530 * X0T12D2 + 530 * X0T14D2 + 529 * X0T15D2 + 88 * X1T7D0 + 48 * X2T10D0 + 55 *
X3T4D0 + 40 * X3T6D0 + 68 * X4T9D0 + 92 * X5T9D0 + 19 * X6T2D0 + 51 * X8T1D0 + 82 * X9T11D0 + 27 * X12T13D0 + 37 * X14T6D0 + 43 *
X14T12D0 + 18 * X4T0D1 + 59 * X5T0D1 + 68 * X7T0D1 + 37 * X10T0D1 + 45 * X11T0D1 + 25 * X13T0D1 + 46 * X15T0D1 + 32 * X4T0D2 + 20
* X5T0D2 + 22 * X7T0D2 + 16 * X10T0D2 + 7 * X11T0D2 + 29 * X13T0D2 + 11 * X15T0D2
;
 + X0T3D1 + X0T4D1 + X0T5D1 + X0T8D1 + X0T12D1 + X0T14D1 + X0T15D1 =  + X4T0D1 + X5T0D1 + X7T0D1 + X10T0D1 + X11T0D1 +
X13T0D1 + X15T0D1;
+ X0T3D2 + X0T4D2 + X0T5D2 + X0T8D2 + X0T12D2 + X0T14D2 + X0T15D2 =  + X4T0D2 + X5T0D2 + X7T0D2 + X10T0D2 + X11T0D2 +
X13T0D2 + X15T0D2;
 + X8T1D0 = 1;
+ X6T2D0 = 1;
+ X0T3D1 + X0T3D2 = 1;
+ X3T4D0 + X0T4D1 + X0T4D2 = 1;
+ X0T5D1 + X0T5D2 = 1;
+ X3T6D0 + X14T6D0 = 1;
+ X1T7D0 = 1;
+ X0T8D1 + X0T8D2 = 1;
+ X4T9D0 + X5T9D0 = 1;
+ X2T10D0 = 1;
+ X9T11D0 = 1;
+ X14T12D0 + X0T12D1 + X0T12D2 = 1;
+ X12T13D0 = 1;
+ X0T14D1 + X0T14D2 = 1;
+ X0T15D1 + X0T15D2 = 1;
 + X1T7D0 = 1;
+ X2T10D0 = 1;
+ X3T4D0 + X3T6D0 = 1;
+ X4T9D0 + X4T0D1 + X4T0D2 = 1;
+ X5T9D0 + X5T0D1 + X5T0D2 = 1;
+ X6T2D0 = 1;
+ X7T0D1 + X7T0D2 = 1;
+ X8T1D0 = 1;
+ X9T11D0 = 1;
+ X10T0D1 + X10T0D2 = 1;
+ X11T0D1 + X11T0D2 = 1;
+ X12T13D0 = 1;
+ X13T0D1 + X13T0D2 = 1;
+ X14T6D0 + X14T12D0 = 1;
+ X15T0D1 + X15T0D2 = 1;
 + 531 * X0T3D1 + 511 * X0T4D1 + 540 * X0T5D1 + 567 * X0T8D1 + 536 * X0T12D1 + 531 * X0T14D1 + 540 * X0T15D1 + 551 * X0T3D2 + 542 *
X0T4D2 + 518 * X0T5D2 + 515 * X0T8D2 + 530 * X0T12D2 + 530 * X0T14D2 + 529 * X0T15D2 + 88 * X1T7D0 + 48 * X2T10D0 + 55 * X3T4D0 +
40 * X3T6D0 + 68 * X4T9D0 + 92 * X5T9D0 + 19 * X6T2D0 + 51 * X8T1D0 + 82 * X9T11D0 + 27 * X12T13D0 + 37 * X14T6D0 + 43 * X14T12D0
+ 18 * X4T0D1 + 59 * X5T0D1 + 68 * X7T0D1 + 37 * X10T0D1 + 45 * X11T0D1 + 25 * X13T0D1 + 46 * X15T0D1 + 32 * X4T0D2 + 20 * X5T0D2 +
22 * X7T0D2 + 16 * X10T0D2 + 7 * X11T0D2 + 29 * X13T0D2 + 11 * X15T0D2
>= 0;
@BIN(X0T3D1); @BIN(X0T3D2); @BIN(X0T4D1); @BIN(X0T4D2); @BIN(X0T5D1); @BIN(X0T5D2); @BIN(X0T8D1); @BIN(X0T8D2);
@BIN(X0T12D1); @BIN(X0T12D2); @BIN(X0T14D1); @BIN(X0T14D2); @BIN(X0T15D1); @BIN(X0T15D2); @BIN(X4T0D1); @BIN(X4T0D2);
@BIN(X5T0D1); @BIN(X5T0D2); @BIN(X7T0D1); @BIN(X7T0D2); @BIN(X10T0D1); @BIN(X10T0D2); @BIN(X11T0D1); @BIN(X11T0D2);
@BIN(X13T0D1); @BIN(X13T0D2); @BIN(X15T0D1); @BIN(X15T0D2); @BIN(X1T7D0); @BIN(X2T10D0); @BIN(X3T4D0); @BIN(X3T6D0);
@BIN(X4T9D0); @BIN(X5T9D0); @BIN(X6T2D0); @BIN(X8T1D0); @BIN(X9T11D0); @BIN(X12T13D0); @BIN(X14T6D0); @BIN(X14T12D0);

END
```

APÊNDICE L – Procedimento 4 para de solução do MDVSP

```
unit dout_rsp;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs,
  StdCtrls, Grids, ExtCtrls, ShellApi;

type
  TFormRspLingo = class(TForm)
    Memo1: TMemo;
    StringGrid1: TStringGrid;
    Label1: TLabel;
    Bevel1: TBevel;
    Label2: TLabel;
    Memo2: TMemo;
    Label3: TLabel;
    Edit1: TEdit;
    Label4: TLabel;
    ButtonSolucao: TButton;
    Memo3: TMemo;
    MemoQUEBRAS_ITER: TMemo;
    Label5: TLabel;
    StringGrid2: TStringGrid;
    ButtonModeloLinear: TButton;
    Memo5: TMemo;
    Label6: TLabel;
    ButtonOpenLingo: TButton;
    MemoFINAL: TMemo;
    MemoQUEBRAS_ACUM: TMemo;
    CheckBox1: TCheckBox;
    Edit2: TEdit;
    Label7: TLabel;
    Edit3: TEdit;
    Label8: TLabel;
    Memo4: TMemo;
    procedure FormCreate(Sender: TObject);
    procedure ButtonSolucaoClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure ButtonModeloLinearClick(Sender: TObject);
    procedure ButtonOpenLingoClick(Sender: TObject);
    procedure Edit2Change(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormRspLingo: TFormRspLingo;
  Parada, lowbound, lowboundanterior, melhor_solucao, SolucaoO-
tima, solucaootimaanterior, Iter_Cont: integer;
  FIM: boolean;
  tempo1, tempo2, tempoi, tempo_iter: TDateTime;
  hora1,min1,seg1,mseg1,hora2,min2,seg2,mseg2: word;

implementation
uses dout_md_padrao, dout_red;
{$R *.DFM}

procedure TFormRspLingo.FormCreate(Sender: TObject);
begin
stringgrid1.cells[0,0]:='Variável';
stringgrid1.cells[1,0]:='De';
stringgrid1.cells[2,0]:='Para';
stringgrid1.cells[3,0]:='Garagem';
stringgrid1.cells[4,0]:='Valor';
stringgrid2.cells[0,0]:='Garag';
stringgrid2.cells[1,0]:='C_esc';
stringgrid2.cells[2,0]:='C_atu';
stringgrid2.cells[3,0]:='Difer';
end;

procedure TFormRspLingo.ButtonSolucaoClick(Sender: TObject);
var
i, j, l, seq_para, custo_atual, garagem, diferenca, cont_seq,
cont_quebra: Integer;
Parcela_Cust_Veic, Parcela_Cust_ViagIni, Parcela_Cust_ViagFim,
Parcela_Cust_ViagInt, Iter_ignorar: integer;
custo_escolhida: array of integer;
pronto: boolean;
matriz_rsp_ini, matriz_rsp_centro: array of array of integer;
s, t, u, v, de, para, gar, viag_fim, viag_ini, viagens_ini, viagens_fim:
string;

begin
Iter_ignorar:=strtoint(Edit3.text);
{antes da interpretaçao ele chama o lingo e resolve o modelo}
tempo_iter:=now;
ButtonOpenLingo.Click;

{após resolver o modelo ele interpreta o resultado obtido}
for i:=0 to 4 do
  for j:=1 to stringgrid1.rowcount-1 do stringgrid1.cells[i,j]:='';
stringgrid1.rowcount:=2;
viag_ini:='';
viag_fim:='';
viagens_ini:='';
viagens_fim:='';
memo2.lines.clear;
nomedoarquivo:=Edit1.text;
Memo1.Lines.LoadFromFile(nomedoarquivo+'_pad2_resp.txt');

{construindo a solucao na grade}
stringgrid1.Visible:=false;
i:=2;
repeat
t:=''; {t guarda a variavel}
u:=''; {u guarda o valor}
s:=memo1.lines[i];
for j:=1 to length(s)-20 do
  begin
  if (j<=35) and (s[j]<>' ') then t:=t+s[j];
  if (j>35) and (s[j]<>' ') then u:=u+s[j];
  end;
if length(u)<>0 then
  begin
  if u[1]='1' then
    begin
    stringgrid1.cells[0,stringgrid1.rowcount-1]:=t;
    stringgrid1.cells[4,stringgrid1.rowcount-1]:=u;
    l:=1;
    de:='';
    para:='';
    repeat
      if (t[l]<>'X') then de:=de+t[l];
      inc(l);
    until t[l]='T';
    repeat
      if (t[l]<>'T') then para:=para+t[l];
      inc(l);
    until (t[l]='D') or (l=length(t));
    stringgrid1.cells[1,stringgrid1.rowcount-1]:=de;
    stringgrid1.cells[2,stringgrid1.rowcount-1]:=para;
    if t[l]='D' then stringgrid1.cells[3,stringgrid1.rowcount-1]:=t[l+1]
      else stringgrid1.cells[3,stringgrid1.rowcount-1]:='0';
    stringgrid1.rowcount:=stringgrid1.rowcount+1;
    end;
  inc(i);
  end;
until (length(u)=0) or (length(t)<=3);
stringgrid1.rowcount:=stringgrid1.rowcount-1;

{passando a solucao da grade para as matrizes de respostas}
aux1:=1;
for i:=1 to stringgrid1.rowcount-1 do
  if stringgrid1.cells[1,i]='0' then
    begin
    setlength(matriz_rsp_ini,3,aux1);
    for j:=1 to 3 do matriz_rsp_ini[j-1,aux1-
1]:=strtoint(stringgrid1.cells[j,i]);
    viag_ini:=viag_ini+stringgrid1.cells[2,i]+' ';
    inc(aux1);
    end;
aux2:=stringgrid1.rowcount-aux1;
```

```
setlength(matriz_rsp_centro,3,aux2);
for i:=1 to stringgrid1.rowcount-1 do
  if (stringgrid1.cells[1,i]<>'0') then
    begin
    aux2:=strtoint(stringgrid1.cells[1,i]);
    for j:=1 to 3 do matriz_rsp_centro[j-1,aux2-
1]:=strtoint(stringgrid1.cells[j,i]);
    if (stringgrid1.cells[2,i]='0') then
viag_fim:=viag_fim+stringgrid1.cells[1,i]+' ';
    end;
viagens_ini:=viagens_ini+viag_ini;
viagens_fim:=viagens_fim+viag_fim;

{Montando sequencias das viagens}
memoQUEBRAS_ITER.Lines.clear;
setlength(custo_escolhida,NumGarag);
aux2:=0;
for i:=1 to aux1-1 do
  begin
  v:='';
  pronto:=false;
  cont_seq:=1;
  cont_quebra:=1;
  seq_para:=matriz_rsp_ini[1,i-1];
  s:='';
  s:='D'+inttostr(matriz_rsp_ini[2,i-1])+' - '+inttostr(seq_para);
  u:='X0T'+inttostr(seq_para)+'D'+inttostr(matriz_rsp_ini[2,i-1]);
  t:='X0T'+inttostr(seq_para)+'D'+inttostr(matriz_rsp_ini[2,i-1]);
  {calcula correcao do modelo}
  garagem:=matriz_rsp_ini[2,i-1];
  custo_atual:=matrizViagExpGaragIni[seq_para-1,garagem-1];
  for j:=1 to NumGarag do custo_escolhida[j-
1]:=matrizViagExpGaragIni[seq_para-1,j-1];
  {fim da 1a etapa de calculo da correcao do modelo}
  repeat
  if matriz_rsp_centro[1,seq_para-1]=0 then
    begin
    pronto:=true;
    s:=s+' - D'+inttostr(matriz_rsp_centro[2,seq_para-1]);
    if t[length(t)]<>inttostr(matriz_rsp_centro[2,seq_para-1]) then
      begin
      u:=u+' +
X'+inttostr(seq_para)+'T0D'+inttostr(matriz_rsp_centro[2,seq_para-
1])+' <= '+inttostr(cont_seq)+';';
      memoQUEBRAS_ITER.lines.add(v);
      memoQUEBRAS_ACUM.lines.add(v);
      memoQUEBRAS_ITER.lines.add(u);
      memoQUEBRAS_ACUM.lines.add(u);
       v:='';
      {calcula correcao do modelo}
      inc(aux2);
      garagem:=matriz_rsp_centro[2,seq_para-1];
      cus-
to_atual:=custo_atual+matrizViagExpGaragFim[seq_para-
1,garagem-1];
      for j:=1 to NumGarag do custo_escolhida[j-
1]:=custo_escolhida[j-1]+matrizViagExpGaragFim[seq_para-1,j-1];
      {seleciona a garagem com menor custo para fazer a
correcao}
      garagem:=0;
      for j:=1 to NumGarag-1 do if cus-
to_escolhida[garagem]>=custo_escolhida[j] then garagem:=j;
      stringgrid2.RowCount:=aux2+1;
      stringgrid2.cells[0,aux2]:=inttostr(garagem+1);
      stringgrid2.cells[1,aux2]:=inttostr(custo_escolhida[garagem]);
      stringgrid2.cells[2,aux2]:=inttostr(custo_atual);
      stringgrid2.cells[3,aux2]:=inttostr(custo_escolhida[garagem]-
custo_atual);
      {fim da 2a etapa de calculo da correcao do modelo}
       end;
    end
      else
    begin
    s:=s+' - '+inttostr(matriz_rsp_centro[1,seq_para-1]);
    u:=u+' + X'+inttostr(matriz_rsp_centro[0,seq_para-
1])+'T'+inttostr(matriz_rsp_centro[1,seq_para-1])+'D0';
    seq_para:=matriz_rsp_centro[1,seq_para-1];
    inc(cont_quebra);
    inc(cont_seq);
    if cont_quebra>=15 then
      begin
      cont_quebra:=0;
      if v='' then v:=u else v:=v+#13+u;
      {memoQUEBRAS_ITER.lines.add(u);
      memoQUEBRAS_ACUM.lines.add(u);}
      u:='';
      end;
    end;
  until pronto;
  memo2.lines.add(s);
  end;
stringgrid1.Visible:=true;

diferenca:=0;
for j:=1 to aux2 do diferen-
ca:=diferenca+strtoint(stringgrid2.cells[3,j]);
memo2.lines.add('');
memo2.lines.add('Custo de interrupção do algoritmo:
'+inttostr(diferenca));
memo4.lines.add('Custo de interrupção do algoritmo:
'+inttostr(diferenca));

{Calculando solucao otima encontrada pelo lingo}
SolucaoOtimaAnterior:=SolucaoOtima;
SolucaoOtima:=0;
Parcela_Cust_veic:=0;
Parcela_Cust_ViagIni:=0;
Parcela_Cust_ViagFim:=0;
Parcela_Cust_ViagInt:=0;
for i:=1 to stringgrid1.rowcount-1 do
  begin
  de:=stringgrid1.cells[1,i];
  para:=stringgrid1.cells[2,i];
  gar:=stringgrid1.cells[3,i];
  aux1:=strtoint(de);
  aux2:=strtoint(para);
  aux3:=strtoint(gar);
  if aux1=0 then
    begin
    Parcela_Cust_veic:=Parcela_Cust_veic+CustoVeic;
    Parce-
la_Cust_ViagIni:=Parcela_Cust_ViagIni+matrizviagexpgaragini[aux
2-1,aux3-1];
    end
      else
        if aux2=0 then
          Parce-
la_Cust_ViagFim:=Parcela_Cust_ViagFim+matrizviagexpgaragfim[
aux1-1,aux3-1]
          else Parce-
la_Cust_ViagInt:=Parcela_Cust_ViagInt+matrizft[aux1-1,aux2-1];
  end;
SolucaoOti-
ma:=Parcela_Cust_veic+Parcela_Cust_ViagIni+Parcela_Cust_Via
gFim+Parcela_Cust_ViagInt;
memo2.lines.add('Solução encontrada pelo lingo:
'+inttostr(SolucaoOtima));
memo2.lines.add('  Parcela Veículos:
'+inttostr(Parcela_Cust_veic));
memo2.lines.add('  Parcela Viagens Iniciais:
'+inttostr(Parcela_Cust_ViagIni));
memo2.lines.add('  Parcela Viagens finais:
'+inttostr(Parcela_Cust_ViagFim));
memo2.lines.add('  Parcela Viagens Internas:
'+inttostr(Parcela_Cust_ViagInt));
memo2.lines.add('LowBound utilizada: '+inttostr(lowbound));
memo2.lines.add('LowBound anterior:
'+inttostr(SolucaoOtimaAnterior));
memo2.lines.add('Solução ótima anterior:
'+inttostr(SolucaoOtimaAnterior));

memo4.lines.add('Solução encontrada pelo lingo:
'+inttostr(SolucaoOtima));
memo4.lines.add('  Parcela Veículos:
'+inttostr(Parcela_Cust_veic));
memo4.lines.add('  Parcela Viagens Iniciais:
'+inttostr(Parcela_Cust_ViagIni));
memo4.lines.add('  Parcela Viagens finais:
'+inttostr(Parcela_Cust_ViagFim));
memo4.lines.add('  Parcela Viagens Internas:
'+inttostr(Parcela_Cust_ViagInt));
memo4.lines.add('LowBound utilizada: '+inttostr(lowbound));

{calcula melhor solucao ate o momento}
if ((SolucaoOtima+diferenca)<=melhor_solucao) then Me-
lhor_solucao:=SolucaoOtima+diferenca;
memo2.lines.add('Melhor Solução: '+inttostr(melhor_solucao));
memo4.lines.add('Melhor Solução: '+inttostr(melhor_solucao));

if ((Melhor_solucao-solucaoOtima)<=(parada*SolucaoOtima/1000))
then FIM:=true;
if FIM then
```

```
  begin
  {calcula tempo total para achara solucao}
  memo2.lines.add('Melhor solução até o momento:
'+inttostr(melhor_solucao));
  tempo2:=now;
  tempo2:=tempo2-tempoi;
  decodetime(tempo2,hora2,min2,seg2,mseg2);
  memo2.lines.add('Encontrada em: '+inttostr(hora2)+'h
'+inttostr(min2)+'m '+inttostr(seg2)+'seg ');
  memo4.lines.add('Encontrada em: '+inttostr(hora2)+'h
'+inttostr(min2)+'m '+inttostr(seg2)+'seg ');
  buttonSolucao.Enabled:=false;
  end
    else
  {sistema da convergencia do MDVSP}
  begin
  lowboundAnterior:=LowBound;
  if (SolucaoOtima>LowBound) or (Iter_cont<=Iter_ignorar) then
    begin
    LowBound:=SolucaoOtima;
    MemoQUEBRAS_ACUM.Clear;
    end;

  {juntar as partes e depois salva-las}
  memoFINAL.Lines.Clear;
  memo5.selectall;
  memo5.CopyToClipboard;
  memoFINAL.pastefromClipboard;
  memoFINAL.Lines.add('>= '+inttostr(lowbound)+';');
  if (Iter_cont<=Iter_ignorar) then
    begin
    memoQUEBRAS_Iter.SelectAll;
    memoQUEBRAS_Iter.CopyToClipboard;
    MemoQUEBRAS_Acum.pastefromClipboard;
    end;

  if (SolucaoOtima=SolucaoOtimaAnterior) then
    begin
    memoQUEBRAS_ACUM.selectall;
    memoQUEBRAS_ACUM.CopyToClipboard;
    memoFINAL.pastefromClipboard;
    end
      else
    begin
    memoQUEBRAS_ITER.selectall;
    memoQUEBRAS_ITER.CopyToClipboard;
    memoFINAL.pastefromClipboard;
    end;
  memo3.selectall;
  memo3.CopyToClipboard;
  memoFINAL.pastefromClipboard;
  memoFI-
NAL.Lines.SaveToFile(nomedoarquivo+'_md_pad2_red3.ltf');

  {calcula tempo da iteracao}
  tempo2:=now;
  tempo2:=tempo2-tempo_iter;
  decodetime(tempo2,hora2,min2,seg2,mseg2);
  memo2.lines.add('Encontrada em: '+inttostr(hora2)+'h
'+inttostr(min2)+'m '+inttostr(seg2)+'seg ');
  memo4.lines.add('Encontrada em: '+inttostr(hora2)+'h
'+inttostr(min2)+'m '+inttostr(seg2)+'seg ');

  inc(iter_cont);
  memo2.lines.add('Iteração número: '+inttostr(iter_cont));
  memo4.lines.add('Iteração número: '+inttostr(iter_cont));
  memo4.lines.add('');
  me-
mo4.Lines.SaveToFile(nomedoarquivo+'_md_pad2_resumo.txt');
  if checkbox1.checked then ButtonSolucao.Click;
  end;

end;

procedure TFormRspLingo.ButtonModeloLinearClick(Sender:
TObject);
var
g,i,j: integer;

procedure quebradelinha;
begin
aux3:=aux3+1;
if aux3>=10 then
  begin
  memo5.lines.add(s);
  aux3:=0;
  s:='';
  end;
end;

begin
{Função objetivo}
s:='';
aux1:=0;
aux2:=0;
aux3:=0;
LowBound:=0;
Melhor_solucao:=99999999;
FIM:=false;
Parada:=strtoint(Edit2.Text);
ButtonSolucao.Enabled:=true;
memo5.lines.clear;
memo5.lines.add('MODEL:');
memo5.lines.add('MIN =');
{MDVSP COM REDUÇÃO - MDVSP - PAD - REDUZIDO 3}

{viagem expressa originárias da garagem}
aux3:=0;
s:='';
for g:=1 to NumGarag do
  for i:=1 to NumViagens do if vgi[i-1]=1 then
    begin
    s:=s+' + '+inttostr(matrizViagExpGaragIni[i-1,g-1]+CustoVeic)+'
* X0T'+inttostr(i)+'D'+inttostr(g);
    aux1:=aux1+1;
    quebradelinha;
    end;
if aux3>=1 then memo5.lines.add(s);

{custo total (FT) entre viagens}
for j:=1 to NumViagens do
  begin
  s:='';
  aux3:=0;
  for i:=1 to NumViagens do
    if (matrizreg1[j-1,i-1]>=1) and (matrizreg1[j-1,i-1]<=2) then
      begin
      s:=s+' + '+inttostr(matrizft[j-1,i-1])+' *
X'+inttostr(j)+'T'+inttostr(i)+'D0';
      aux1:=aux1+1;
      quebradelinha;
      end;
  if s<>'' then memo5.lines.add(s);
  end;

{viagem expressa retorno para a garagem}
aux3:=0;
s:='';
for g:=1 to NumGarag do
  for i:=1 to NumViagens do if vgf[i-1]=1 then
    begin
    s:=s+' + '+inttostr(matrizViagExpGaragFim[i-1,g-1])+' *
X'+inttostr(i)+'T0D'+inttostr(g);
    aux1:=aux1+1;
    quebradelinha;
    end;
if aux3>=1 then memo5.lines.add(s);
memo5.lines.add(';');
memo5.lines.add('');

{Restrição onde o mesmo numero de veiculos deve chegar e sair
de uma garagem}
for g:=1 to NumGarag do
  begin
  s:='';
  aux3:=0;
   for i:=1 to NumViagens do
    if vgi[i-1]=1 then
      begin
      s:=s+' + X0T'+inttostr(i)+'D'+inttostr(g);
      quebradelinha;
      end;
  s:=s+' = ';
  memo5.lines.add(s);
  s:='';
  for i:=1 to NumViagens do
    if vgf[i-1]=1 then
      begin
      s:=s+' + X'+inttostr(i)+'T0D'+inttostr(g);
      quebradelinha;
      end;
  s:=s+';';
```

```
  memo5.lines.add(s);
  end;

{obrigatoriedade de cumprir as viagens - escolha pelo menos uma
linha}
memo5.lines.add('');
for i:=1 to NumViagens do
  begin
  s:='';
  aux3:=0;
  for j:=1 to NumViagens do
      if (matrizreg1[j-1,i-1]>=1) and (matrizreg1[j-1,i-1]<=2) then
        begin
        s:=s+' + X'+inttostr(j)+'T'+inttostr(i)+'D0';
        quebradelinha;
        end;
  for g:=1 to NumGarag do if vgi[i-1]=1 then
    s:=s+' + X0T'+inttostr(i)+'D'+inttostr(g);
  s:=s+' = 1;';
  memo5.lines.add(s);
  end;

{obrigatoriedade de cumprir as viagens - escolha pelo menos uma
coluna}
memo5.lines.add('');
for i:=1 to NumViagens do
  begin
  s:='';
  aux3:=0;
  for j:=1 to NumViagens do
      if (matrizreg1[i-1,j-1]>=1) and (matrizreg1[i-1,j-1]<=2) then
        begin
        s:=s+' + X'+inttostr(i)+'T'+inttostr(j)+'D0';
        quebradelinha;
        end;
  for g:=1 to NumGarag do if vgf[i-1]=1 then
    s:=s+' + X'+inttostr(i)+'T0D'+inttostr(g);
  s:=s+' = 1;';
  memo5.lines.add(s);
  end;

{insere restricao de custo minimo para acelerar o modelo}
memo5.lines.add('');
aux3:=0;
s:='';
for g:=1 to NumGarag do
  for i:=1 to NumViagens do if vgi[i-1]=1 then
    begin
    s:=s+' + '+inttostr(matrizViagExpGaragIni[i-1,g-1]+CustoVeic)+'
* X0T'+inttostr(i)+'D'+inttostr(g);
    aux1:=aux1+1;
    quebradelinha;
    end;
if aux3>=1 then memo5.lines.add(s);
for j:=1 to NumViagens do
  begin
  s:='';
  aux3:=0;
  for i:=1 to NumViagens do
    if (matrizreg1[j-1,i-1]>=1) and (matrizreg1[j-1,i-1]<=2) then
      begin
      s:=s+' + '+inttostr(matrizft[j-1,i-1])+' *
X'+inttostr(j)+'T'+inttostr(i)+'D0';
      aux1:=aux1+1;
      quebradelinha;
      end;
  if s<>'' then memo5.lines.add(s);
  end;
aux3:=0;
s:='';
for g:=1 to NumGarag do
  for i:=1 to NumViagens do if vgf[i-1]=1 then
    begin
    s:=s+' + '+inttostr(matrizViagExpGaragFim[i-1,g-1])+' *
X'+inttostr(i)+'T0D'+inttostr(g);
    aux1:=aux1+1;
    quebradelinha;
    end;
if aux3>=1 then memo5.lines.add(s);

{insere variáveis binárias}
for i:=1 to NumViagens do
  for g:=1 to Numgarag do
    if vgi[i-1]=1 then
memo3.lines.add('@BIN(X0T'+inttostr(i)+'D'+inttostr(g)+');');
for i:=1 to NumViagens do
  for g:=1 to Numgarag do
    if vgf[i-1]=1 then
memo3.lines.add('@BIN(X'+inttostr(i)+'T0D'+inttostr(g)+');');
for j:=1 to NumViagens do
  for i:=1 to NumViagens do
    if (matrizreg1[j-1,i-1]>=1) and (matrizreg1[j-1,i-1]<=2) then
      memo3.lines.add('@BIN(X'+inttostr(j)+'T'+inttostr(i)+'D0);');

memo3.lines.add('');
memo3.lines.add('!numero de variaveis='+inttostr(aux1+1));
memo3.lines.add('');
memo3.lines.add('END');
memo3.lines.add('SET TERSEO 1');
memo3.lines.add('GO');
memo3.lines.add('DIVERT '+nomedoarquivo+'_pad2_resp.txt');
memo3.lines.add('SOLUTION');
memo3.lines.add('RVRT');
memo3.lines.add('QUIT');

{salva a primeira versao do modelo}
memoFINAL.Lines.Clear;
memo5.selectall;
memo5.CopyToClipboard;
memoFINAL.pastefromClipboard;
memoFINAL.Lines.add('>= 0;');
memo3.selectall;
memo3.CopyToClipboard;
memoFINAL.pastefromClipboard;
memoFI-
NAL.Lines.SaveToFile(nomedoarquivo+'_md_pad2_red3.ltf');

tempoi:=now;
Iter_Cont:=0;

if checkbox1.checked then ButtonSolucao.Click;

end;

procedure TFormRspLingo.ButtonOpenLingoClick(Sender:
TObject);
var
SEInfo: TShellExecuteInfo;
ExitCode: DWORD;
ExecuteFile, ParamString: string;

begin
ExecuteFile:='c:\programas\lingo8\lingo80.exe';
ParamString:='/t'+nomedoarquivo+'_md_pad2_red3.ltf';
FillChar(SEInfo, SizeOf(SEInfo), 0) ;
SEInfo.cbSize := SizeOf(TShellExecuteInfo) ;
with SEInfo do
  begin
  fMask := SEE_MASK_NOCLOSEPROCESS;
  Wnd := Application.Handle;
  lpFile := PChar(ExecuteFile) ;
  lpParameters := PChar(ParamString);
  // lpDirectory := PChar(StartInString);
  nShow := SW_SHOWNORMAL;
  end;
if ShellExecuteEx(@SEInfo) then
  begin
  repeat
  Application.ProcessMessages;
  GetExitCodeProcess(SEInfo.hProcess, ExitCode);
  until (ExitCode <> STILL_ACTIVE) or Application.Terminated;
  {ShowMessage('Calculator terminated');}
  tempo1:=now;
  decodetime(tempo1,hora1,min1,seg1,mseg1);
    repeat
    tempo2:=now;
    decodetime(tempo2,hora2,min2,seg2,mseg2);
    until seg2-seg1>=2;
  end else ShowMessage('Error starting Lingo!');
end;

procedure TFormRspLingo.FormActivate(Sender: TObject);
begin
edit1.text:=nomedoarquivo;
end;

procedure TFormRspLingo.Edit2Change(Sender: TObject);
begin
parada:=strtoint(edit2.text);
end;

end.
```

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ADMINISTRAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ADMINISTRAÇÃO
DOUTORADO EM SISTEMAS DE INFORMAÇÃO E APOIO À DECISÃO

DESENVOLVIMENTO DE HEURÍSTICA PARA SOLUÇÃO DO PROBLEMA DE ESCALONAMENTO DE VEÍCULOS COM MÚLTIPLAS GARAGENS

Autor: Leonardo Rosa Rohde

Orientador: Denis Borenstein, PhD

Tese apresentada como requisito para obtenção do título de Doutor pelo Programa de Pós-Graduação da Faculdade de Administração da Universidade Federal do Rio Grande do Sul.

Porto Alegre
2008

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas