Universidade Federal de Minas Gerais

NARI LOUISE TENKLEY

ORDER PICKING: MODELOS E ALGORITMOS DE ROTEAMENTO

Belo Horizonte
2008

Programa de Pós-Graduação em Engenharia de Produção
Escola de Engenharia
Universidade Federal de Minas Gerais

ORDER PICKING: MODELOS E ALGORITMOS DE ROTEAMENTO

POR:
NARI LOUISE TENKLEY

Dissertação de mestrado submetida à banca examinadora aprovada pelo Colegiado do Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Minas Gerais como requisito para obtenção do título de Mestre em Engenharia.

Orientador: Professor Gilberto de Miranda Júnior
Co-orientador: Professor Luiz Ricardo Pinto

Belo Horizonte
Março de 2008

AGRADECIMENTOS

Ao Professor Gilberto Miranda Jr., pela orientação, paciência e otimismo durante este trabalho.

Ao Professor Luiz Ricardo Pinto pelo apoio e confiança.

Ao Bruno, pela dedicação, companheirismo e por todas as contribuições a este trabalho.

Ao meu marido Guilherme pelo amor e carinho, tornando todos os meus dias mais felizes.

À minha família que sempre me apoiou e incentivou e que, mesmo quando estou longe deles, estão sempre ao meu lado.

Aos meus sogros e cunhado que estiveram presentes em todos os momentos com muito amor e alegria.

A todos os professores do PPGEP por todos os conhecimentos que eles me passaram ao longo deste curso.

Aos meus amigos e colegas das disciplinas e do Ladesp pelo companheirismo, ajuda, amizade e compreensão.

À CAPES pela bolsa concedida, que foi fundamental para a realização deste trabalho.

“Everything is vague to a degree you do not realize till you have tried to make it precise.”

-Bertrand Russel

RESUMO

Este trabalho trata o problema de roteamento de veículos para o processo de recolhimento de itens dentro de armazéns. O objetivo é minimizar o número de veículos necessários para atender toda a demanda e voltar para o doca dentro de um dado tempo máximo, que representa o nível de serviço do armazém. Os veículos podem atender toda a demanda de um item qualquer ou somente uma fração da demanda, deixando uma parte da demanda a ser recolhida por outro veículo. O problema pode ser classificado como o problema de roteamento de veículos com coleta fracionada e um prazo de entrega considerado como um tipo de janela de tempo (*vehicle routing problem with time windows and split pick-ups* – VRPTWSP).

Dois novos modelos de otimização para este problema são desenvolvidos: um modelo estático e um modelo *on-line*. O modelo estático considere o caso no qual todos os veículos encontram-se estacionados na doca. O modelo *on-line* considera o caso no qual os veículos já estão em andamento e as rotas originais podem ser alteradas para acomodar demandas recém-chegadas. Resultados dos dois modelos para problemas teste são dados. Além das duas formulações novas, também são introduzidas uma heurística para geração de limites superiores e novas desigualdades válidas para um limite inferior mais justo.

Palavras-Chave: Problema de roteamento de veículos com janelas de tempo e coleta fracionada, prazo de entrega, *on-line*, armazém, *picking*.

ABSTRACT

This paper examines the vehicle routing problem for order picking within warehouses. The objective is to minimize the number of vehicles needed to pick all of the demand and return to the depot within a given amount of time, which represents the warehouse's service level. Each vehicle can pick all of the demand in a given location or a fraction of the total demand, leaving part of the demand to be picked by another vehicle. The problem can be classified as the vehicle routing problem with split pick-ups and service level constraints considered as a form of time windows (VRPTWSP).

Two new optimization models are presented for this problem: a static model and an on-line model. The static model considers the case in which all of the vehicles are at the depot. The on-line model considers the case in which the vehicles have already left the depot and can alter their original routes to pick newly received items on the same route. Computational results for the two models are given. In addition to these two new formulations, a heuristic for generating upper bounds and a new class of valid inequalities which tightens the lower bounds are presented.

Key Words: Vehicle Routing Problem with Time Windows and Split Pick-ups, service level constraints, on-line, warehouse, picking.

SUMÁRIO

AGRADECIMENTOS.....i

RESUMO.....iii

ABSTRACT.....iv

SUMÁRIO.....v

LISTA DE FIGURAS.....vii

LISTA DE TABELAS.....viii

Capítulo 1 – INTRODUÇÃO 1

1.1 A Importância da Logística e dos Problemas de Roteamento de Veículos.....1

1.2 Caracterização dos problemas de roteirização.....2

1.3 Objetivo.....4

Capítulo 2 - REVISÃO DA LITERATURA 6

2.1 Métodos Clássicos de Resolução dos Problemas de Roteirização de Veículos.....6

2.1.1 Método de Economia de Clarke e Wright (Savings Algorithm)

2.1.2 Método de Partição do Roteiro Gigante

2.1.3 Algoritmo de Varredura (Sweep Algorithm)

2.1.4 Algoritmo de Fisher e Jaikumar

2.1.5 Pós-Otimização

2.2 Métodos Modernos de Resolução.....11

2.2.1 Simulated Annealing

2.2.2 Busca Tabu

2.2.3 Taburoute

2.2.4 Algoritmos de Busca Genética

2.2.7 O Procedimento de Memória Adaptativa de Rochat e Taillard

2.3 Problemas de Roterização de Veículos na Literatura.....20

2.3.1 Problema Clássico de Roteirização de Veículos (VRP e CVRP)

2.3.2 Problema de Roteirização de Veículos com Janelas de Tempo (VRPTW)

2.3.3 Problema de Roteirização de Veículos com Entrega Fracionada (VRPSD)

2.3.4 Problema de Roteirização de Veículos com Janelas de Tempo e Entrega Fracionada (VRPTWSD)

2.4 Roteirização de Veículos em Armazéns.....27

2.4.1 Problemas de Roterização de Veículos em Armazéns na Literatura

Capítulo 3 - DEFINIÇÃO DO PROBLEMA ESTÀTICO 29
3.1 Modelo Matemático – Formulação Estático.....30
3.2 Problemas Teste.....32
3.3 Resultados do Modelo Estático.....33

Capítulo4 - DEFINIÇÃO DO PROBLEMA *ON-LINE*.....*36*
4.1 Modelo Matemático – Formulação *On-Line*.....37
4.2 Problemas Teste.....40
4.3 Resultados – Modelo *On-Line*.....42

Capítulo 5 - NOVA CLASSE DE DESIGULADADES VALIDAS.....44
5.1 Exemplo Ilustrativo - Desigualdades Válidas.....44
5.2 Algoritmo para Desigualdades Válidas.....48
5.3 Resultados Exemplo Ilustrativo – Desigualdades Válidas.....50

Capítulo 6 - HEURÍSTICA PARA O PROBLEMA *ON-LINE*.....51

Capítulo 7 - CONCLUSÕES.....54

Referências Bibliográficas.....56

Lista de Figuras

Figura 1. Formação de rota gigante 10

Figura 2. Pseudocódigo para o algoritmo de Simulated Annealing.....13

Figura 3: (A) A solução VRP, (B) a solução SDVRP.....24

Figura 4. Armazém retangular de 20 pontos.....33

Figura 5. Rotas e Demanda Recolhido pelos Veículos 1 e 2.....34

Figura 6. Rotas e Demanda Recolhido pelos Veículos 3 e 4.....34

Figura 7. Rotas e Demanda Recolhido por Veículo 5.....35

Figura 8. Rota do Veículo 5.....41

Figura 9. Rota do Veículo 5 criada pelo modelo on-line e demanda nova recolhido.....42

Figura 10. Rota Original do Veículo 1 45

Figura 11. Rota Original do Veículo 2 45

Figura 12. Exemplo de Rota – Heurística.....51

Lista de Tabelas

Tabela 1. Caracterização dos problemas de roteirização.....3

Tabela 2: Demanda presente em cada nó do Conjunto A.....32

Tabela 3. Resultados do modelo estático.....35

Tabela 4: Demanda nova presente em cada nó.....40

Tabela 5. Resultados do modelo on-line.....42

Tabela 6. Demanda Original e Demanda Nova para exemplo de 10 nós.....44

Capítulo 1

Introdução

1.1 A Importância da Logística e dos Problemas de Roteamento de Veículos

Nas últimas décadas, com o barateamento dos custos de transportes e comunicação, tem-se percebido uma expansão de comércio caracterizado pelo aumento de distância entre a produção e consumo de produtos. Com maiores canais de distribuição, mais competição internacional, e clientes que exigem cada vez mais rapidez na entrega de produtos e serviços, a logística mudou de ser uma atividade necessária para facilitar a produção e venda de produtos para ser uma atividade que adicione valor aos produtos e serviços. Para uma empresa que trabalha em um mercado substancialmente maior do que sua área de produção, o gerenciamento dos sistemas de logística torna-se uma atividade essencial (BALLOU, 1999).

A grande maioria de sistemas de logística depende de uma frota de veículos para fornecer bens e/ou serviços para seus clientes, varejistas, armazéns, ou fábricas. Com o objetivo de controlar os custos destas frotas, a empresa precisa decidir como carregar os veículos e quais serão os destinos de cada um (SIMCHI-LEVI et al., 2005). A decisão de como formar e utilizar as frotas pode ter grande impacto nos custos da empresa já que, de acordo com Ballou (1999), os custos de transporte geralmente correspondem a entre um e dois terços do total de custos na área de logística.

Dentro da área de otimização combinatória, o problema de roteirização de veículos (*vehicle routing problem*, VRP) é um problema difícil e de muito apelo prático. O problema tem como objetivo identificar conjuntos de rotas para uma frota de veículos que serve um grupo definido de clientes, minimizando os custos e satisfazendo todas as demandas. Este problema é interessante teoricamente por ser *NP*-difícil e é interessante também por ter aplicação na vida real, com aplicações vistas tanto em movimentação de

produtos quanto na distribuição de serviços. O problema foi introduzido há mais de 40 anos e continua sendo estudado e é relevante até hoje.

Um cenário no qual o roteamento de veículos é muito importante é no processo de recolhimento (*picking*) dentro de armazéns. A operação de armazéns é uma atividade vital para qualquer cadeia de suprimentos. A competição que existe nos mercados e o investimento que os armazéns representam requerem que bastante atenção e esforço sejam aplicados na melhoria dos armazéns (GU et al., 2007). Segundo Peterson & Aase (2004), a atividade de recolhimento de pedidos normalmente conta por 50-75% do custo total de operação dentro de um armazém.

1.2 Caracterização dos Problemas de Roteirização

O VRP clásico é o problema de roteamento de veículos capacitados (*Capacitated Vehicle Routing Problem*, CVRP). O problema, de acordo com Cordeau et al. (2002), é definido em um grafo *G = (V,A)* onde $V = \{v_0, v_1, \dots, v_n\}$ é um conjunto de vértices e $A = \{(v_i, v_j): v_i, v_j \in V, i \neq j\}$ é um conjunto de arcos. O vértice v_0 representa o doca e os outros vértices representam clientes. Os arcos têm um custo (c_{ij}) e um tempo para atravessar (t_{ij}). Nos casos onde o custo e tempo são simétricos, é comum definir o VRP em um grafo não direcionado *G = (V, E)* onde $E = \{(v_i, v_j): v_i, v_j \in V, i < j\}$ é um aresta. Cada cliente tem uma demanda não negativa q_j e um tempo de serviço t_j. Uma frota de *m* veículos idênticos com capacidade *Q* é baseado no doca. O número de veículos ou é fixo ou é uma variável de decisão. O problema é criar um conjunto de no máximo *m* rotas de tal maneira que: (1) cada rota começa e termina no doca; (2) cada cliente recebe exatamente uma visita feita por um veículo; (3) a demanda total de cada rota não excede Q; e (4) o custo total das rotas é minimizado.

O problema é apropriado para uso em muitas situações na vida real, mas a grande variedade de circunstâncias presentes na prática traz a necessidade de criar variações específicas do VRP básico. Os problemas de roteamento podem ser caracterizados de

acordo com os critérios mostrados na Tabela 1, que são as características mais importantes dos problemas. (CONÇEIÇÃO, 2006; GOLDBARG & LUNA, 2000)

Tabela 1. Caracterização dos problemas de roteirização.

Característica	**Alternativas Possíveis**
Objetivo	• Minimizar custos fixos • Minimizar custos de operação na rota • Minimizar o número de veículos necessários.
Tipo de frota disponível	• Homogênea • Heterogênea
Número de docas	• Único • Vários
Operação	• Entrega • Coleta • Entrega e coleta
Tempo para servir um determinado arco ou nó	• Sem restrições • Janela de tempo • Tempo especifico
Tamanho da frota de veículos	• Um veículo • Mais de um veículo (limitada) • Mais de um veículo (ilimitada)
Natureza da demanda e parâmetros	• Determinística • Estocástica
Localização da demanda	• Nos vértices • Nos arcos • Ambos
Grafo de substrato	• Direcionado • Não Direcionado • Misto
Restrições de coleta/entrega	• Certos veículos proibidos de visitar certos clientes • Clientes requerem várias visitas durante um período de tempo • Clientes podem ser servidos por mais de um veículo
Jornada	• Sem restrições de duração • Limite de duração • Interrupções para almoço, etc.
Custos	• Variáveis (associados à rota escolhida) • Fixos • Custo do transporte alternativo para a demanda não atendida

1.3 Objetivo

O objetivo deste trabalho é desenvolver um modelo *on-line* do problema de roteamento de veículos com coleta fracionada e janelas de tempo na forma de prazo de entrega (*Vehicle Routing Problem with Time Windows and Split Pick-up* – VRPTWSP) para aplicação dentro de armazéns.

As características deste problema são:

- O objetivo é minimizar o número de veículos necessários;
- A frota de veículos é homogênea;
- Existe um único doca;
- Os pontos de demanda têm um tempo máximo para coleta e entrega na doca, de acordo com o nível de serviço desejado;
- Os pontos de coleta podem ser visitados por mais de um veículo (a coleta da demanda pode ser fracionada entre dois ou mais veículos).

A solução do problema é atingida com a utilização de um modelo matemático que minimiza o número de veículos necessários para recolher toda a demanda dentro do prazo de entrega. O modelo é inédito na literatura.

O problema aqui tratado é também *on-line*, ou seja, as rotas dos veículos podem ser modificadas em tempo real de acordo com mudanças no estado do sistema. Um sistema *on-line* não garante que a solução é o ótimo global, mas, ao invés disso, utiliza as informações disponíveis no momento para gerar o ótimo instantâneo. Este tipo de ótimo local é útil para sistemas não determinísticos, quando o futuro é imprevisível e então todas as informações não estão disponíveis no momento da decisão.

Nesta dissertação a situação considerada é a de um armazém que recebe pedidos e deve recolher os itens dos pedidos dentro de uma dada duração de tempo máxima (prazo de entrega) para garantir uma qualidade de serviço aos seus clientes. No caso, não é possível saber com precisão a hora de chegada dos pedidos futuros nem o tipo de produtos e quantidades que os pedidos conterão. Por isso é impossível determinar o

ótimo global durante o processo e um modelo *on-line* torna-se bastante atrativo já que oferece flexibilidade e a habilidade de determinar a melhor decisão no momento.

O modelo *on-line* é criado então com a intenção de ser útil e prático na realidade de um armazém. Um dos desafios para isto é que o VRPTWSP é um problema difícil de resolver e, portanto, pode requer bastante tempo computacional. Para o modelo ter uso prático, é necessário que seja rápido.

Capítulo 2

Revisão da Literatura

2.1 Métodos Clássicos de Resolução dos Problemas de Roteirização de Veículos

Os métodos clássicos geralmente colocam muita ênfase na resolução rápida do VRP para achar uma solução viável. Os modelos a serem discutidos aqui são o método de economia de Clarke e Wright (*the savings algorithm)*, o método de partição do roteiro gigante, o método de varredura (*the sweep algorithm)*, e o método de designação generalizada de Fisher e Jaikumar. Estes modelos foram desenvolvidos durante os anos 1960-1990. Durante os anos mais recentes houve melhorias no desempenho dos algoritmos, mas os modelos clássicos ainda são bastante utilizados na prática. Em geral, os atributos utilizados para julgar um algoritmo são exatidão e velocidade, mas na prática, flexibilidade e simplicidade são muito importantes também. (CORDEAU et al., 2002)

2.1.1 Método de Economia de Clarke e Wright (Savings Algorithm)

A heurística de economia de Clarke e Wright foi desenvolvida no ano 1964. Este algoritmo é um dos mais conhecidos e é ainda bastante utilizado na prática ainda. Ele é utilizado quando o número de veículos é uma variável de decisão e funciona tanto para os problemas direcionados como para os não-direcionados. A idéia básica do algoritmo é aproveitar a economia que se pode ter quando juntam-se duas rotas existentes. (OSMAN, 1993)

O algoritmo processa o problema da seguinte forma. O algoritmo inicia com uma solução viável onde cada veículo serve um cliente, criando n rotas do tipo (v_0, ..., v_i, v_0), onde v_0 é o doca central. Quando duas rotas (v_0, ..., v_i, v_0) e (v_0, v_j, ..., v_0) podem ser juntadas de forma viável em uma rota só (v_0, ..., v_i, v_j, ..., v_0), a economia de tal mudança é avaliada, onde a economia é calculada por $s_{ij} = c_{i0} + c_{0j} - c_{ij}$. Desta forma, o algoritmo (versão paralela) segue os seguintes passos (CONCEIÇÃO 2006):

Passo 1 – Calcular a matriz das economias s_{ij}

Passo 2 – Ordenar, decrescentemente as economias s_{ij}

Passo 3 – Para o maior valor s_{ij} ainda não utilizado, combinar as rotas i e j desde que sejam atendidas (a) a viabilidade de roteamento, (b) a restrição de capacidade do veículo, e (c) a restrição de tempo máximo de viagem.

Passo 4 – Repetir o passo 3, até esgotar todos os s_{ij}.

Na versão paralela do algoritmo, a fusão que oferece a maior economia é sempre escolhida e implementada, tirando da solução os segmentos (0 v_j) e (v_i, 0) e introduzindo o novo segmento (v_i, v_j), conforme foi mostrado nos passos acima. Existe outra versão do problema, a versão seqüencial, na qual o algoritmo trabalha com uma rota de cada vez, procurando atribuir o máximo de clientes possíveis (começando com os que oferecem mais economia) até esgotar a rota. Quando não existir mais combinações viáveis para aquela rota, o processo começa com outra rota, e assim continua até que não existem mais combinações de rotas viáveis. Na maioria dos casos, a versão paralela resulta em um custo menor quando o custo é determinado pela distância total percorrida (CORDEAU, 2002). A versão seqüencial, contudo, prioriza o aproveitamento das rotas, e geralmente cria menos rotas que a versão paralela (BELFIORE, 2006). Como o objetivo do problema estudado aqui é minimizar o número de veículos, a versão seqüencial é mais adequada. É comum aplicar um método de pós-otimização, como 2-opt ou 3-opt, para melhorar a solução obtida com o algoritmo. Estes métodos serão discutidos na secção 2.5.

O algoritmo de Clarke e Wright é bom por ser simples e rápido. Geralmente ele produz resultados razoáveis, mas às vezes também a solução é de baixa qualidade. Mas a falta de flexibilidade, como citado por Cordeau (2002) é o pior aspeto do algoritmo. É possível adicionar restrições ao algoritmo, mas normalmente tais adições causam uma

deterioração grande na exatidão da solução. Isto acontece por causa do algoritmo ser guloso (*greedy*), ou seja, o algoritmo sempre busca a melhor opção no momento e não tem como voltar depois de tomar a decisão (vista curta).

2.1.2 Método de Partição do Roteiro Gigante

O método de partição do roteiro gigante tem muitas variações, que foram desenvolvidas para servir a vários objetivos. O método apresentado aqui é bastante simples e serve para se ter uma idéia geral das heurísticas desta classe. Este algoritmo cria primeiramente as rotas para depois agrupá-las.

O primeiro passo do algoritmo é criar uma rota que passa por todos os clientes, um problema do caixeiro viajante. O problema do caixeiro viajante pode ser visto como um VRP mais simples, onde existe um único veículo que precisa passar por todos os locais de demanda minimizando a distância total percorrida. A resolução deste problema fornece uma solução em que os nós adjacentes estarão razoavelmente próximos um ao outro, o que é chamado de roteiro gigante. Na resolução deste problema inicial do caixeiro viajante, a solução não considera as demandas dos clientes nem as restrições dos veículos, assim sendo, o objetivo do passo é simplesmente criar uma rota de pequenas distâncias entre clientes. Depois, o passo de partição do roteiro gigante leva em conta as restrições.

Seja o roteiro gigante $v_0 - v_1 - v_2 ... - v_k - v_0$, e a distância (ou custo), $c(k,m)$, entre dois clientes v_k e v_m igual à distância total de uma rota viável para atender os clientes v_k, v_{k+1}..., v_{m-1}. Com as distâncias $c(k,m)$ definidas, determina-se a distância mínima entre o primeiro cliente da rota gigante e o doca central. Se a solução do problema de distância mínima consistir em p arcos, então a partição da rota gigante consistirá de p subrotas. Este tipo de algoritmo costuma achar soluções de maior qualidade comparado com o algoritmo de economias de Clarke e Wright (CONCEIÇÃO, 2006).

2.1.3 Algoritmo de Varredura (Sweep Algorithm)

O algoritmo de varredura tem seu princípio nos trabalhos de Wren (1971), Wren e Holiday (1972), e Gillett e Miller (1964). O algoritmo de varredura, em contraste com o roteiro gigante, agrupa primeiro e cria as rotas depois. A técnica de varredura é aplicável somente para problemas planares, o que desde o inicio limita bastante a aplicação (não seria aconselhável utilizar este algoritmo em um contexto urbano ou de armazéns porque o formato das ruas/corredores normalmente é de grade).

O algoritmo inicia com a escolha de um veículo disponível e começa a criar uma rota para este veículo. Para designar um vértice (cliente) para um veículo uma semi-reta centralizada no doca central é girada em um dado sentido e o vértice com menor ângulo é designado para o veículo enquanto a capacidade da rota não ultrapassa o limite. Depois de terminar com um veículo, outro disponível é escolhido e o processo se repete até que todo o plano seja varrido. Uma vez que os grupos de clientes são identificados, pode-se utilizar qualquer um dos algoritmos para a resolução do problema do caixeiro viajante para determinar a rota de cada grupo. A rota de um grupo de clientes pode ser chamada de uma pétala (*petal*). Este algoritmo normalmente não mostra desempenho melhor do que o algoritmo de economia de Clarke e Wright em termos de velocidade ou exatidão, e continua sendo inflexível.

2.1.4 Algoritmo de Fisher e Jaikumar

O algoritmo de Fisher e Jaikumar resolve o problema de designação generalizado (*generalized assignment problem, GAP*) para formar grupos de clientes, e depois a rota de cada grupo é determinado com a resolução do problema do caixeiro viajante. O problema pode ser dividido em quatro passos: (1) Escolher os pontos sementes para iniciar cada grupo. (2) Calcular o custo d_{ik} de designar cada cliente i para o grupo k. (3) Resolver o GAP, com custo d_{ij}, peso de clientes q_i, e capacidade do veículo Q para formar os grupos. (4) Resolver o problema do caixeiro viajante para cada grupo. A resolução do GAP (no passo (3)) é NP-difícil, e normalmente é resolvido utilizando uma técnica de relaxação Lagrangiana.

2.1.5 Pós-Otimização

É bastante comum utilizar um procedimento de melhoria depois de se obter o resultado da heurística, chamado de pós-otimização. Um método simples, embora não muito completo, seria aperfeiçoar cada rota, sem alterar os agrupamentos de clientes. Mas comum, porém, são métodos de pós-otimização nos quais os grupos de clientes podem ser rearranjados. Os métodos de 2-opt e 3-opt são deste tipo. Para 2-opt e 3-opt, o primeiro passo é criar uma rota gigante a partir do roteamento obtido. Por exemplo, se existirem p rotas, criam-se p cópias do doca onde a distancia entre duas cópias é nula. Este passo é mostrado com um exemplo de 3 rotas na Figura 1, onde v_0 representa o doca, v_1, ..., v_6 representam os clientes, e c_1, c_2 e c_3 são as cópias do doca (CONCEIÇÃO, 2006).

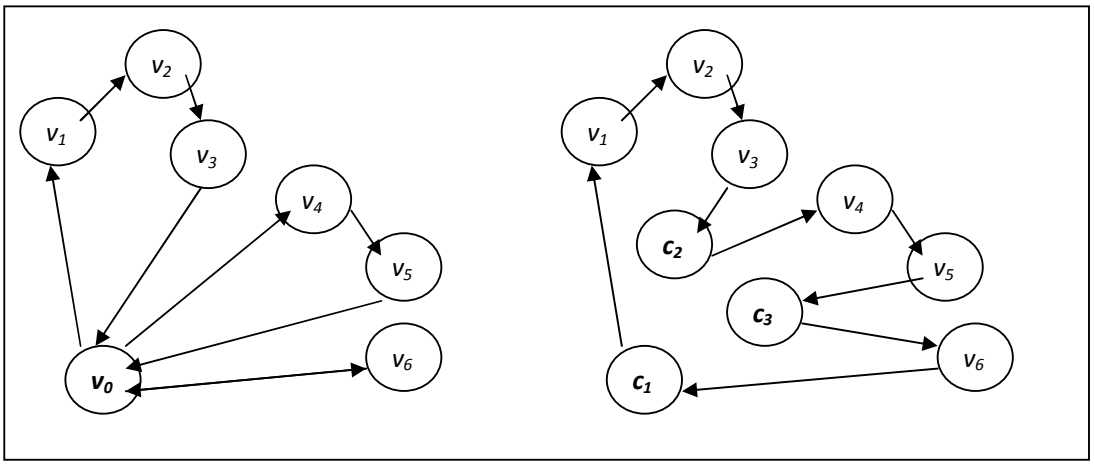

Figura 1. Formação de rota gigante

Depois de criar a rota gigante, examina-se se a substituição de dois ou três arcos da rota por outro par produza uma redução do custo total, verificando se as restrições de capacidade e tempo de viagem estão dentro da viabilidade do problema. No procedimento, se os arcos são trocados de tal maneira que duas cópias do doca se tornam adjacentes na rota gigante, então o tamanho da frota é diminuído e a roteirizarão inclui uma rota a menos.

2.2 Métodos Heurísticos de Resolução

As pesquisas mais recentes do VRP, nos últimos 15 anos, têm concentrado na criação de meta-heurísticas que conseguem fazer uma busca mais completa da área. A concentração das meta-heurísticas é principalmente baseada em duas técnicas gerais: busca local e busca de população. Busca local muda a solução atual para soluções próximas (na vizinhança). Os algoritmos mais comuns de busca local são *simulated annealing* (SA) e busca tabu (*tabu search*). Os dois serão discutidos aqui. A outra técnica, busca de população, cria um grupo de soluções boas (soluções pais) que são utilizadas e recombinadas para criar as soluções filhas. Exemplos de algoritmos deste tipo são busca genética (*genetic search*) e procedimentos de memória adaptável (*adaptive memory procedures*) que também serão discutidos nesta secção. As meta-heurísticas normalmente precisam de mais tempo operacional comparado com os métodos clássicos, mas produzem soluções de alta qualidade.

2.2.1 Simulated Annealing

Simulated Annealing (SA) é um meta-algoritmo utilizado para achar a solução ótima global de uma função dentro de um espaço grande. O processo foi criado por S. Kirkpatrick, C. D. Gelatt, e M. P. Vecchi no ano 1983. A idéia do algoritmo veio do processo metalúrgico "annealing", no qual um formato de treliça puro é formado em um sólido por um processo de aquecer o material até derramar, e depois esfriar o material lentamente até solidificar. Com a utilização deste processo o material cria a oportunidade de achar um estado de energia menor. No sentido de problemas de otimização combinatória, *simulated annealing* é um método de busca local aleatório, que procura soluções na vizinhança. A grande vantagem deste algoritmo é que ele pode escolher soluções piores (soluções de custo maior), o que evita ficar emperrado em um mínimo local. Para qualquer problema finito, a probabilidade que o algoritmo SA terminar com a solução ótima global aproxima 100% enquanto a duração máxima é prolongada, mas a convergência de algoritmos de simulated annealing normalmente

requer tempo exponencial, e por isso é muito comum utilizar a técnica como um algoritmo de aproximação. (AARTS & LENSTRA, 1997)

Simulated annealing é um *threshold algorithm*. Este tipo de algoritmo gera soluções vizinhas, e calcula a diferença de custo entre a solução vizinha e a solução atual. Depois, compara-se esta diferença com o *threshold*, ou limiar. Se a diferença for menor do que o limiar, se aceita a mudança para a solução vizinha. Caso contrário, a busca continua desde a solução atual. A seqüência de limiares é ($t_k|k = 0, 1, 2,...$), onde t_k é o limiar utilizado na iteração k do algoritmo. No caso de SA, t_k é uma variável com valor entre zero e infinito. Os t_k seguem uma função de distribuição de probabilidade tal que qualquer vizinho pode ser aceito com probabilidade positiva, mas esta função é criada para que as soluções vizinhas com custo muito maior do que a solução atual tenham pouca probabilidade de ser aceitas e as vizinhas com custo menor do que a atual tenham grande possibilidade de aceitação.

Durante cada iteração t de SA, uma solução é escolhida aleatoriamente. As soluções vizinhas são escolhidas com um processo definido pelo usuário. Se o custo for menor a mudança é aceita, ou seja, $x_{t+1} = x$. Senão, $x_{t+1} = x$ com probabilidade p_t e $x_{t+1} = x_t$ com probabilidade $1\text{-}p_t$, onde p_t costuma ser uma função da iteração t e a diferença entre os custos. É muito comum definir $p_t = \exp(-[f(x) - f(x_t)]/c_t)$, onde $f(x) - f(x_t)$ é a diferença de custo entre a solução atual e a solução vizinha.

Como mencionado acima, a convergência de SA geralmente requer muito tempo, e por isso o algoritmo precisa de uma maneira de controlar a duração. Este controle chama-se *cooling schedule* (programa de esfriamento), e é definido pelo usuário. O parâmetro utilizado para o controle, c_k (às vezes denotado como T, para *temperatura*) normalmente diminui pouco a pouco, com cada iteração t. Quando c_k for grande (no começo do algoritmo), o algoritmo deixa mudanças para praticamente qualquer vizinha, mesmo sendo uma solução pior. Isto faz com que o algoritmo mude muito de lugar e começa a "conhecer" varias áreas da região estudada. Com tempo, os valores de c_k vão diminuindo gradualmente e com isso a probabilidade de se aceitar vizinhas com custo maior também diminui. No final do algoritmo, quando $c_k = 0$, o algoritmo tem comportamento análogo ao de um algoritmo guloso. Resumindo, o algoritmo vai para grandes regiões do espaço com soluções boas no início, depois se concentra em áreas

cada vez menores, e no final tende a buscar só as soluções de maior descida. Existem alguns possíveis critérios para terminar o algoritmo, como por exemplo: (a) o custo não diminuir por, no mínimo π_1% durante, no mínimo, k_1 ciclos consecutivos de T iterações; (b) o número de mudanças aceitas tem sido menos de π_2% de T por k_2 ciclos consecutivos de T iterações: (c) k_3 ciclos de T iterações foram executados.

Mostra-se, na Figura 2, um pseudocódigo simples como exemplo do algoritmo SA, utilizando as seguintes definições de variáveis:

- Começa no estado ***s0*** e faz um máximo de ***kmax*** passos
- O chamado ***vizinho(s)*** gera aleatoriamente uma vizinha do estado s
- A chamada ***random()*** dá um valor aleatório [0,1)
- A chamada ***temp(r)*** define c_k dado a fração *r* do tempo que já foi gasto.
- sn = solução nova, en = energia nova.

Figura 2. Pseudocódigo para o algoritmo de Simulated Annealing

```
s := s0; e := E(s);                    // Estado, energia inicial
K := 0;                                // Conta da avaluação de energia
While k<kmax                           // Enquanto tem tempo
        sn := vizinho(s);              // Escolha uma vizinha
        en := E(sn) ;                  // Calcula a energia dela
```

2.2.2 Busca Tabu

Busca tabu (*tabu search*, TS) é uma técnica de busca local, onde soluções novas são procuradas na vizinhança da solução atual. TS diferencia-se de outras meta-heurísticas por utilizar *memória* para registrar as soluções recentes. Os algoritmos de TS são considerados os melhores (até hoje) para os problemas de roteamento de veículos, e por isso são muito pesquisados. (CORDEAU, 2002)

O algoritmo de TS utiliza a idéia de memória de curto tempo para seguir não somente os valores atuais da função objetivo e outras informações locais, mas também informações sobre o processo de exploração – especificamente as últimas soluções visitadas. Para uma iteração, *t*, o algoritmo muda da solução x_t para a melhor solução na vizinhança, x_{t+1}. Para evitar que o algoritmo fique parado em um ótimo local, o TS pode mudar para uma solução pior (uma solução de custo maior). Como visto na secção 2.2.1, *simulated annealing* também deixa acontecer mudanças para vizinhas piores. A diferença entre SA e TS é a forma como tais mudanças acontecem – SA não guia a escolha da vizinha enquanto TS sempre escolha a melhor opção entre as vizinhas viáveis. Quando TS escolhe uma solução de custo maior, no entanto, corre o risco do algoritmo voltar à solução x_t na próxima iteração (ou uma das próximas), ou seja, corre o risco do algoritmo ficar fazendo ciclos entre as mesmas soluções. É para evitar esta eventualidade que a memória do processo de exploração é necessária. TS cria uma lista de soluções tabu - soluções que não podem ser escolhidas. A lista tabu inclui *T* soluções que não podem participar da próxima mudança, o que evita um ciclo envolvendo menos de *T* soluções. É claro que este procedimento não proíbe ciclos maiores que *T*.

A lista tabu, composta de soluções já visitadas, pode ficar difícil de programar, e por isso muitas vezes a lista é composta de *mudanças* feitas ou certos atributos das mudanças, o que é mais fácil de programar. O problema com esta técnica é que, às vezes, uma solução que não foi visitada pode aparecer na lista tabu. Para evitar que uma solução boa seja bloqueada por este motivo, existe a possibilidade de rejeitar o status tabu, feito por condições de nível de aspiração. Se uma solução satisfizer uma das condições de nível de aspiração, a solução pode ser aceita, mesmo sendo tabu.

Um mecanismo para melhorar TS, o de intensificação e diversificação, tem sido implementado por vários pesquisadores com sucesso. Especificamente, a função objetivo *f* pode ser substituída por ſ, o que permite a intensificação e diversificação da busca. Às vezes intensificar a busca em uma das regiões é produtivo porque pode conter algumas soluções boas. Essa intensificação pode ser feita dando prioridade para soluções com atributos em comum com a solução atual. Este tipo de intensificação deve ser feito durante algumas iterações, mas depois é aconselhável mudar para outra região, ou seja, é bom diversificar. Intensificação e diversificação podem ser feitas com

a adição de termos na função objetivo – a intensificação penaliza soluções longe da solução atual, e a diversificação penaliza soluções perto da solução atual. Cada um destes termos tem um peso que é modificado durante a busca para que o processo vá de um para o outro (de intensificação a diversificação e vice versa).

Para mostrar os passos do TS, é conveniente definir os seguintes termos:

L = o conjunto de soluções viáveis

i = solução atual

i^* = a melhor solução já vista

k = a iteração

j = a solução nova

$N(i)$ = a vizinhança da solução atual

$N(i,k)$ = a vizinhança de i na iteração k

O processamento essencial do algoritmo, então, pode ser descrito da seguinte forma conforme Hertz et al. (1997):

Passo 1: Escolha uma solução inicial i em L e faça $i^* = i$.

Passo 2: Faça $k=k+1$ e gere um subconjunto V^* de soluções em $N(i,k)$ de tal maneira que uma das condições tabu $t_r(i,m)$ em T_r é violada ($r = 1,..., t$) ou no mínimo uma das condições de aspiração $a_r(i,m)$ em $A_r(i,m)$ é atingido.

Passo 3: Escolha a melhor j das mudanças que podem ser aplicadas a i dentro de V^*, com respeito à função objetiva f ou ƒ, e faça $i = j$.

Passo 4: Se $f(i) < f(j^*)$, então faça $i^* = i$.

Passo 5: Atualize as condições tabu e aspiração.

Passo 6: Se a condição de término for atinginda, pare. Senão, volte para passo 2.

2.2.3 Taburoute

A heurística de *Taburoute*, desenvolvida por Gendeau et al. em 1994, é mais complicado do que as implementações anteriores de TS e inclui varias atributos inovadoras. A definição da vizinhança da solução x_t para *taburoute* é o conjunto de soluções atingíveis desde x_t pela remoção de um vértice v da sua rota atual e sua inserção em uma nova rota s por um processo de inserção generalizado (generalized insertion procedure, GENT). Quando isto acontece, a reinserção de v na rota r fica tabu até a iteração $\theta + 1$, onde θ é escolhido aleatoriamente em um dado intervalo. O intervalo [5, 10] foi recomendado pelo Taillard no contexto do *quadratic assignment problem*, e é útil para evitar ciclos.

Para minimizar as chances de o algoritmo parar em um ótimo local (não global), *taburoute* examine soluções inviáveis com respeito às restrições de capacidade ou a duração das rotas. Para fazer isto, a função objetivo tem termos de penalidade que são auto-ajustantes para capacidade ou duração inviável. A cada 10 iterações os termos podem ser ajustado – se as 10 iterações anteriores foram viáveis, divide a penalidade por 2, se foram inviáveis multiplica por 2. A metodologia de *taburoute* também incorpora uma estratégia de diversificação, penalizando os vértices que tem sido mudado freqüentemente para melhorar as chances de escolher os outros vértices. Conforme descrito para a metodologia de TS, isto pode ser feito na função objetivo, adicionando um termo proporcional à freqüência absoluta de movimento do vértice v sendo considerado.

Os passos básicos, descritos em Gendreau et al. (1997), de taburoute são descritos a abaixo, com algumas definições a seguir.

W = o conjunto de vértices que são candidatos para reinserção em outra rota em cada iteração.

$q \leq |W|$ = número dos vértices em W para os quais uma reinserção tentativa é feito.

k = número de iterações sem melhoria na função objetiva.

Passo 1: *Iniciação*. Gerar [(n^0,5)/2] soluções iniciais e faça busca tabu com $W = V/\{v_0\}$, $q = 5m$, e $k = 50$. Este valor assegura que a probabilidade de selecionar um vértice de cada rota é no mínimo 90%.

Passo 2: *Melhoria da Solução*. Começando com a melhor solução vista no passo 1, faça busca tabu com $W = V/\{v_0\}$, $q = 5m$, e $k = 50n$.

Passo 3: *Intensificação*. Começando com a melhor solução vista no passo 2, faça busca tabu com $k = 50$. Aqui W é o conjunto dos $\lfloor |V| / 2 \rfloor$ vértices que tem sido mudado mais freqüentemente nos passos 1 e 2, e $q = |W|$.

Taburoute pode produz resultados de alta qualidade. A metodologia é muito flexível porque os parâmetros colocados na função objetivo podem ser alterados. Provavelmente o pior aspeto de *taburoute* é sua complexidade, já que tem nove parâmetros controlados pelo usuário. (CORDEAU, 2002)

2.2.4 Algoritmos de Busca Genética

O algoritmo de busca genética (*genetic search*, GS) pertence ao grupo de algoritmos evolucionários e será o primeiro algoritmo de busca de população discutido aqui. Algoritmos evolucionários são baseados na evolução de biologia como os processos de reprodução, mutação, e seleção natural. A idéia fundamental de algoritmos genéticos é a combinação de dois 'pais' para criar um 'filho'. No algoritmo as soluções potenciais, os cromossomos, 'evolvem' até soluções melhores. Os cromossomos normalmente são uma série de *bits* porque é mais fácil lidar com este formato de informação. A função objetivo do algoritmo, chamado de função de *fitness*, é minimizada para o VRP. Conforme a descrição de GS de Gendreau et al. (1997), pode-se considerar uma população $X^t = \{x^t_1,..., x^t_N\}$. Em cada iteração, X^t é transformada para X^{t+1}, com a produção de filhos – os filhos tomam o lugar dos pais. A idéia é continuar as características de uma boa solução para as novas gerações. Pequenas mutações aparecem nos filhos aleatoriamente para assegurar diversidade na população. Uma descrição simples do algoritmo segue abaixo, com algumas definições a seguir.

k_1 = número de gerações

$k_2 \leq N/2$ = número de seleções por geração.

Com cada iteração t = 1,..., k_1, aplica passos 1 até 3 k_2 vezes, depois faça passo 4.

Passo 1: Seleção. Selecione aleatoriamente dois pais de X^t com a probabilidade relacionada inversamente ao valor da função de fitness (objetivo), *f*.

Passo 2: *One-point crossover*. Cria dois filhos dos dois pais por uma troca de um *bitstring* localizado depois de um *cutpoint* seleta aleatoriamente, como mostrado no exemplo abaixo. No exemplo, o *cutpoint* é seleto depois do segundo *bit*.

Pai 1	1	0	1	0	0
Pai 2	**0**	**0**	**1**	**1**	**1**

Filho 1	1	0	**1**	**1**	**1**
Filho 2	**0**	**0**	1	0	0

Passo 3: Mutação. Aplica uma mutação aleatória a cada filho, trocando os *bits* com uma probabilidade pequena.

Passo 4: Nova população. Obter X^{t+1} de X^t, removendo as $2k_2$ piores soluções em X^t e colocando os $2k_2$ filhos recém criados nos lugares deles.

A aplicação de algoritmos GS aos problemas de roteirização de veículos tem algumas dificuldades, já que a metodologia não foi desenvolvido originalmente para otimização combinatória. O problema mais básico é que a representação com *bitstrings* não é natural para o VRP. Outro problema é que a criação de filhos (soluções) inviáveis é muito provável com o método de trocar em um ponto (passo 2). Como o VRP é um problema de seqüênciamento, todos os cromossomos têm o mesmo valor e somente a ordem muda. Mesmo com estas dificuldades, pesquisadores estão explorando as possibilidades de GS. Gendreau et al (1997) citam três exemplos de implementação de GS para o VRP com restrições de capacidade e janelas de tempo.

2.2.7 O Procedimento de Memória Adaptativa de Rochat e Taillard

O algoritmo de memória adaptativa (*adaptive memory procedure, AMP*) foi desenvolvido em 1995 por Rochat e Taillard. O que é chamado de memória adaptativa é um conjunto de soluções boas. Este conjunto atualiza-se dinamicamente com a entrada de soluções boas, e a saída de soluções com menos potencial. As novas soluções são criadas com a combinação de soluções no conjunto. Por isso, a atualização pode ser considerada uma forma geral de busca genética, uma busca de população. Para o VRP, dado um conjunto de boas soluções, novas soluções são obtidas pela extração de rotas; as rotas de maior qualidade têm maiores chances de ser extraídas. Enquanto se extrai rotas de veículos, tem que ter cuidado para não incluir as rotas com clientes já tratados. O processo termina com um conjunto de rotas selecionadas e alguns clientes sem rota. Por isso uma nova solução é obtida utilizando busca tabu para os clientes sem rota.

O algoritmo não é muito complexo, mas o uso de memória adaptativa requer mais tempo de processamento. Cordeu et al (2002) afirmam que "o conceito de memória adaptativa é provavelmente uma das idéias mais poderosos submetidas na área de metaheurísticas nos últimos anos".

2.3 Problemas de Roterização de Veículos na Literatura

Nesta seção será apresentada uma revisão da literatura referente aos problemas de VRP mais relevantes em relação ao problema estudado aqui (problema de roteirização de veículos com coleta fracionada e janelas de tempo).

2.3.1 Problema Clássico de Roteirização de Veículos (VRP e CVRP)

O problema clássico de roteirização de veículos foi descrito na secção 1.2, e será resumido de novo aqui. O problema consiste de um único doca, um dado número de veículos idênticos, e um conjunto de nós com demanda não negativa. Os arcos que conectam os nós têm uma distância, ou custo, não negativo para serem atravessados. As restrições do VRP são: (a) todas as rotas começam e terminam no doca, e (b) cada nó de demanda é visitado exatamente uma vez por exatamente um veículo. O problema capacitado, CVRP, adiciona a restrição de que a demanda total de uma rota não pode exceder a capacidade do veículo.

Gendreau, Hertz e Laporte (1994) implementaram uma heurística de busca tabu, chamado TABUROUTE, para o CVRP com as restrições adicionais de que cada cliente requer um tempo de serviço e que existe um limite de tempo máximo de jornada permitido para cada rota. O objetivo do problema estudado é minimizar a distância total percorrida. A heurística deixa soluções inviáveis entrarem, penalizando o objetivo quando isto ocorre, para evitar que a heurística fique parada em um mínimo local. Também utilizaram uma técnica de inserção poderosa que muda as soluções periodicamente, evitando assim que um ótimo local seja escolhido pela heurística.

Barbarosoglu e Ozgur (1999) desenvolveram um algoritmo utilizando busca tabu para resolver um problema de VRP para uma empresa de logística na Turquia. O problema que resolveram é o CVRP clássico. Os autores utilizaram uma estratégia de intensificação para examinar com mais precisão as soluções que prometem estar no

caminho de um ótimo local (ou global). A vizinhança foi criada com a troca de vértices de rotas diferentes e a utilização de trocas 2-Opt. De acordo com os autores, as soluções foram compatíveis com os melhores resultados até então, e a heurística foi suficientemente rápida e confiável para ser utilizada na prática.

Kelly e Xu (1999) criaram uma heurística de busca tabu para resolver problemas clássicos de VRP, inclusive o CVRP e a variação em que a duração de cada rota não pode exceder um dado limite. O processo de resolução tem duas fases. Na primeira, de geração, rotas únicas são criadas através de metodologias já conhecidos e relativamente simples. Na segunda fase, de integração, um algoritmo de busca tabu é utilizado para resolver o modelo de partição de conjuntos.

Campos e Mota (2000) desenvolveram duas heurísticas para o CVRP. A técnica geral para as duas heurísticas é de busca tabu. Na primeira heurística a solução inicial é sempre viável, e a busca tabu é utilizada para melhorar esta solução. Na segunda heurística obteve-se a solução inicial com a relaxação linear do modelo, o que significa que a solução inicial não precisa ser viável. Alguns arcos desta solução inicial são fixos antes da primeira fase de busca tabu, e depois a busca tabu é utilizada sem fixar qualquer arco da solução atual.

Achuthan, et al. (2003) apresentaram um algoritmo de *branch-and-cut* para achar uma solução exata do CVRP. O algoritmo utiliza novos cortes planos (cutting planes) que foram desenvolvidos. Os autores conseguiram resolver problemas com até 100 clientes e compararam os tempos computacionais para mostrar que os cortes melhoram o tempo computacional.

Vaidyanathan et al. (1999) estudaram uma variação interessante do CVRP na qual o problema precisa ser resolvido para um ambiente de produção *just-in-time*. O objetivo do problema é minimizar o número de veículos, mas os veículos devem entregar somente a quantidade de materiais necessários até o veículo voltar com outra rota. Com isto o tempo parado dos veículos deve ser minimizado. Os autores desenvolveram uma heurística para resolver o problema e utilizaram uma relaxação linear do problema para oferecer um limite inferior. A heurística tem melhores resultados quando a capacidade do veículo é grande.

Uma outra variação do CVRP clássico foi estudada por Sariklis e Powell (2000). Nesta variação, chamado de CVRP aberto, o veículo não precisa voltar para o doca depois de visitar os clientes, ou então precisa visitar os clientes na ordem reversa para voltar. A heurística apresentada para este problema tem duas fases, a primeira é de agrupar e a segunda determina o roteamento. Os autores relatam bons resultados comparados com outras heurísticas da literatura para a maioria dos casos, porém os tempos computacionais eram maiores para os problemas maiores.

2.3.2 Problema de Roteirização de Veículos com Janelas de Tempo (VRPTW)

O problema de roteirização de veículos com janelas de tempo (VRPTW) é uma extensão do VRP clássico. O problema difere-se do VRP com a adição de janelas de tempo definidas para a coleta (ou entrega) nos clientes. Uma janela de tempo especifica horários mínimo e máximo de atendimento do cliente, dando assim uma janela de tempo durante a qual o cliente precisa ser visitado.

Desrosiers, et al. (1988) estudaram o problema de múltiplos caixeiros viajantes com janelas de tempo para o caso de transporte escolar. Os autores minimizaram o número de veículos necessários utilizando o método de Relaxação Lagrangiana. Problemas com até 151 clientes foram resolvidos otimamente.

Desrochers, et al. (1992) desenvolveram um algoritmo para o VRPTW que é capaz de resolver problemas com até 100 clientes. O método de resolução foi geração de colunas para uma formulação de partições de conjuntos (*set partitioning*). O método obteve resultados muito bons, resolvendo problemas muito maiores do que existia antes, mas os autores notaram que a formulação não é muito boa para casos onde cada rota tem muitos clientes a serem visitados.

Thompson e Psaraftis (1993) implementaram heurísticas que fazem uma busca de vizinhança com a técnica de transferências cíclicas. As heurísticas foram testadas com três problemas diferentes, incluindo o VRPTW. As desvantagens mais claras são o pior

caso do algoritmo e a complexidade, mas os resultados foram competitivos e o tempo de computação foi razoável.

Badeau et al. (1997) apresentaram uma heurística de busca tabu paralela para o VRPTW. O método produziu resultados similares aos da busca tabu seqüencial, a versão paralela mostrou melhorias no tempo computacional. Portanto, os autores recomendam esta heurística para situações nas quais o tempo de resolução é um fator importante.

Kohl e Madsen (1997) apresentaram um método que utiliza relaxação Lagrangiana. As restrições que obrigam que cada cliente receba uma visita foram relaxadas. O problema Mestre acha os multiplicadores Lagrangiana e o sub-problema consiste em o problema de caminho mínimo com limites de janelas de tempo e capacidade. Um limite inferior do número de veículos foi implementado e o algoritmo foi testado em problemas de até 100 clientes. Os resultados foram bons, resolvendo alguns problemas que não tinham sido resolvidos anteriormente.

Varias heurísticas foram testadas no VRPTW por Tan et al. (2001). Especificamente, os autores impementaram Simulated Annealing, Busca Tabu, e Algoritmos Genéticos nos problemas testes de 56 e 100 clientes. O artigo apresentou bons resultados e representou um dos primeiros artigos a investigar vários algoritmos de inteligência artificial de uma só vez.

Bard, et al. (2002) estudaram o VRPTW com o objetivo de minimizar o número de veículos. Utilizaram o método de *branch-and-cut* e vários cortes foram apresentados. O método foi testado para instâncias de 50 e 100 nós e foi concluído que as restrições de eliminação de sub-tours foram as mais eficazes.

Uma variação do VRPTW que considere o número de veículos disponíveis é limitado foi estudada por Lau, et al. (2003). Já que o número de veículos é limitado, uma solução viável pode desobedecer algumas restrições. Uma solução onde algum cliente não é visitado pode ser viável, tanto quanto uma solução onde o veículo não visita o cliente durante a janela de tempo desejada. No caso das restrições de janelas de tempo, existe uma penalidade para chegar atrasado ao cliente. Os autores apresentaram um limite superior e implementaram busca tabu, cujos resultados ficaram próximo dos limites superiores.

2.3.3 Problema de Roteirização de Veículos com Entrega Fracionada (VRPSD)

O problema de VRP onde a entrega ou coleta em cada cliente pode ser divida entre dois ou mais veículos é o VRP com entrega (ou coleta) fracionada, em inglês o *vehicle routing problem with split deliveries* (VRPSD). O problema é então uma relaxação do VRP clássico, porém continua sendo um problema computacionalmente difícil (Archetti et al. 2005, Dror e Trudeau 1989). Em 1989 Dror e Trudeau apresentaram um argumento para este tipo de problema, mostrando a economia potencial de deixar esta divisão de entrega. No artigo um exemplo simples foi apresentado, o que será reproduzido aqui.

Exemplo: Existem três pontos de demanda com as seguintes demandas: $d_1 = 3$, $d_2 = 4$, e $d_3 = 3$. As distancias entre pontos, onde 0 representa o doca, são: $C_{0i} = 10$ para i = 1,2,3; $C_{12} = 5 = C_{23}$; e $C_{13} = 10$. A capacidade dos veículos é 5 unidades. A solução deste problema com o VRP clássico é de custo 60 e requer três veículos. A solução onde a demanda pode ser fracionada, o SDVRP, tem custo 50 e utiliza só dois veículos. A ilustração deste exemplo segue na Figura 3.

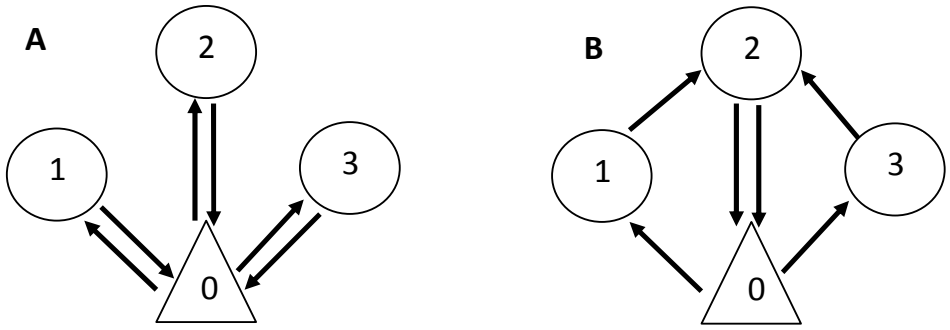

Figura 3: (A) A solução VRP, (B) a solução SDVRP

Além deste exemplo ilustrativo, Dror e Trudeau (1989) mostraram economias para uma grande variedade de problemas de roteamento. Em 1990 Dror e Trudeau publicaram uma análise mais profunda destas economias. Resultados de 540 problemas com até 150 clientes foram apresentados, onde os resultados do SDVRP foram comparados com

os resultados do VRP. Os resultados mostram que situações onde as demandas são pequenas (menos de 10% da capacidade do veículo), a diferença entre o SDVRP e o VRP é muito pequeno. Para os problemas onde a demanda nos clientes é maior que 10% da capacidade do veículo, economias muito significantes em termos de número de veículos e distância percorrida foram realizados com o SDVRP.

Em um artigo subseqüente, Dror, Laporte e Trudeau (1994) concentraram mais na resolução do SDVRP. Várias desigualdades válidas foram desenvolvidas, com uma hierarquia entre elas. Um algoritmo de *branch-and-bound*, baseada na relaxação de restrições, foi apresentado. Os autores mostraram que a restrições reduziram o gap entre os limites inferiores e superiores.

Mullaseril, et al. (1997) aplicaram o SDVRP para um problema de distribuição de alimentos para gado (livestock feed distribution). O artigo mostra que o problema tem bons resultados na prática.

Bellenguer, et al. (1997) apresentaram um novo limite inferior para o SDVRP. O limite é baseado na descrição do poliedro do SDVRP e uma nova família de desigualdades válidas. O método de *cutting plane* conseguiu resolver exatamente algumas instâncias, inclusive uma de 50 clientes, mas os autores admitam que instâncias grandes não gerassem sempre soluções viáveis.

Em 2006, Archetti, et al. desenvolveram um algoritmo de busca tabu para o SDVRP. O algoritmo é bastante fácil de implementar e conseguiu achar a solução ótimo de problemas pequenos (até 15 clientes) em menos de 1 segundo. A vizinhança do problema é obtida pela remoção de um cliente das rotas onde esta, e sua colocação em outra rota. Como o problema deixa que um cliente possa ser servido por mais de um veículo, a vizinhança pode ser criada também removendo o cliente de uma rota só. Em geral o algoritmo obteve resultados bons.

Archetti, et al. (2006) fizeram uma análise sobre o SDVRP que foi mais profundo do que o original feito for Dror e Trudeau (1989, 1990). Os autores mostraram que existem instâncias onde a economia realizada utilizando o VRP ao invés do SDVRP pode causar custos de até duas vezes maiores. Além de examinar as economias do SDVRP apresentado por Dror e Trudeau, a alteração em que a demanda de um ponto

pode ser maior do que a capacidade do veículo é também estudada, mostrando a mesma economia de 50%.

Um novo algoritmo para o SVRP foi apresentado em 2007 por Jin, et al. O Algoritmo tem duas etapas e utiliza várias desigualdades válidas para resolver o problema na otimalidade. Na primeira fase, os clientes são agrupados e alocados a um veículo e na segunda fase as rotas são desenhadas. As desigualdades válidas são utilizadas na primeira fase e os autores mostram que o uso das desigualdades reduz significantemente o número de iterações, e o tempo computacional, até achar a solução ótima.

2.3.4 Problema de Roteirização de Veículos com Janelas de Tempo e Entrega Fracionada (VRPTWSD)

O problema de roteamento de veículos com janelas de tempo e entrega fracionada (VRPTWSD) foi apresentada em 1995 por Frizzell e Griffin. O problema combina a extensão do VRP onde os clientes podem exigir entrega durante um dado intervalo de tempo, e a relaxação do VRP na qual um cliente pode ser visitado por mais de um veículo. Três heurísticas foram desenvolvidas para este problema, uma para construir rotas e duas para melhorar as soluções viáveis. Os autores criaram instâncias próprias para este novo problema, sendo todas elas com distancias de *grid network* ao invés de usar distâncias Euclidianas. As heurísticas também considerem tempos de entregas não-lineares.

Ho e Haugland (2004) implementaram um algoritmo de busca tabu para o VRPTWSD. Resultados para problemas com até 100 clientes foram relatados. Uma comparação interessante entre o problema com e sem repartição de demanda foi feita, mostrando que os benefícios de demanda fracionada dependem da relação entre a demanda e a capacidade dos veículos. Os autores notaram que as instâncias nas quais a repartição da demanda oferece maiores benefícios requerem mais tempo computacional para resolver.

2.4 Roteirização de Veículos em Armazéns

A otimização das rotas de recolhimento é um dos problemas estudados na literatura para reduzir o custo de operação de armazéns. O problema de roteamento em armazéns tem como objetivo achar a melhor seqüência (menor tempo ou custo) para o recolhimento de todos os itens de todos os pedidos, ou seja, busca-se descobrir o caminho mínimo (DE KOSTER & VAN DER POORT 1998; ROODBERGEN & DE KOSTER 2001).

A política de apanha é um fator de grande influência na determinação dos caminhos mínimos na operação de recolhimento em armazéns. Várias políticas de apanha podem ser utilizadas, por exemplo, apanha por pedido, agrupamento de pedidos, *sort-while-pick*, *sort-after-pick*, apanha por pedido com zonas, e agrupamento com zonas (GU et al. 2007; YOON & SHARP 1996).

A maioria dos modelos para resolução do problema de roteamento na literatura são variações do *Vehicle Routing Problema* (VRP – problema de roteamento de veículos), mas outros modelos também podem ser utilizados para o caso, como o modelo para seqüenciamento de máquinas paralelas com tempos de *setup*. Chen e Powell (1999) utilizaram a técnica de geração de colunas para otimizar este modelo, encontrando a hora de término (de todos os *jobs*) e o número ponderado de *jobs* atrasados e obtiveram resultados bons para problemas grandes.

2.4.1 Problemas de Roterização de Veículos em Armazéns na Literatura

De Koster e Vanderpoort (1998) fizeram uma comparação entre algoritmos para roteamento dentro de armazéns. O problema com um doca foi estudado e também uma situação mais moderna na qual os veículos podem depositar os itens no final de cada corredor. Um novo algoritmo de otimização e a heurística *S-shaped*, popular na prática, foram comparados. Os resultados mostram que para um dado número de corredores, quanto mais itens no pedido, melhor os resultados da heurística *S-Shaped*. Mesmo

assim, o novo algoritmo mostrou capacidade de reduzir a distância em entre 7% e 34% comparado com a heurística *S-shaped*.

Daniels, Rummel e Schantz, (1998) introduziram uma nova formulação para o problema de roteamento que incorpora também o problema de designação de produtos aos locais dentro do armazém. Para resolver o problema, um algoritmo de busca tabu é utilizado e comparado com outras heurísticas. Em geral, a nova formulação oferece uma maneira de identificar com alocar produtos aos locais com baixo custo.

Ascheuer, Grotschel, e Abdel-Hamid (1999) estudaram o problema para armazéns automátizados. O artigo é relevante para este trabalho por considerar o caso de um sistema on-line, onde o objetivo é minimizar a distância no momento da decisão, o que não garante o ótimo global ao longo do dia. O artigo oferece uma boa descrição teórica do problema, porém nenhuma formulação específico do problema foi apresentada. O artigo compara os resultados de várias heurísticas, incluindo uma desenvolvida pelos autores, para o problema do caixeiro viajante assimétrico através de simulação.

Roodbergen e de Koster (2001) apresentaram um algoritmo para minimizar a distância da rotas de recolhimento. O algoritmo foi desenhado especificamente para comparar armazéns onde os veículos podem mudar de corredor no final e também existam até três fileiras transversais para mudar de corredor. Os autores concluíram, através de simulação, que um corredor no meio do armazém pode diminuir bastante as distâncias das rotas, principalmente em armazéns grandes.

Capítulo 3

Definição do Problema Estático

Duas formulações do problema de roteamento de veículos com prazo de entrega (janelas de tempo) e coleta fracionada (VRPTWSP) serão estudados aqui. O modelo que será discutido primeiro é o modelo estático que começa no estado inicial, onde todos os veículos encontram-se estacionados na doca. O problema estático é definido em um grafo $G=(V, E)$ onde $V=\{1, 2, ..., n\}$ representa um conjunto de vértices e E é o conjunto de arcos que ligam os vértices. O vértice 1 representa o doca e os demais vértices são os locais demanda onde produtos podem ser recolhidos. O grafo consiste em um conjunto de arcos, E, onde o custo do arco (comprimento do arco) v_{ij} do arco $(i, j) \in E$ é não-negativo. Os veículos começam e terminam suas rotas no mesmo local do doca. Existe uma demanda inteira d_k para o produto k. Os veículos podem atravessar um arco $(i, j) \in E$ onde existe a demanda d_k no vértice j sem a obrigação de recolher qualquer fração da demanda d_k, ou seja, um veículo pode passar por um vértice sem recolher material demandada. Um veículo pode também recolher só uma fração da demanda total em um dado vértice, deixando então a demanda restante para ser recolhida por outros veículos. O número de veículos c que podem ser utilizado em uma solução é ilimitado. Os veículos são capacitados, com capacidade máxima de cada veículo igual a Cap. Cada rota também tem uma restrição de custo, Γ, que se refere ao nível de serviço do armazém (tempo máximo para recolher os produtos). Este nível de serviço é um prazo de entrega, já que obriga os veículos a atenderem a demanda dentro de uma duração de tempo específica.

A solução do problema é recolher toda a demanda sem exceder a capacidade dos veículos ou o nível de serviço. Dado isso, o objetivo é minimizar o número de veículos necessários, com o objetivo secundário de minimizar a distância percorrida pelos veículos.

3.1 Modelo Matemático – Formulação Estático

A notação do modelo estático, incluindo parâmetros e variáveis de decisão, e o próprio modelo matemático são apresentados e explicados a seguir.

Parâmetros

- *I:* conjunto de locais no armazém;
- *K:* conjunto de zonas de coleta;
- *C:* conjunto de veículos;
- *E:* conjunto de arcos *(i, j)* entre local *i* e *j*;
- *n:* número máximo de veículos (limite superior)
- v_{ij}*:* custo de atravessar arco *(i, j)*;
- d^k *:* demanda do produto *k*;
- *Cap:* capacidade de cada veículo;
- *Γ:* Custo máximo de cada rota – nível de serviço;
- C^c *:* Custo de ativação do veículo *c (número grande).*

Variáveis de Decisão

- x_{ij}^c *:* igual a 1 se o veículo c atravessa arco *(i, j);* 0 se não;
- y^c *:* igual a 1 se o veículo *c* é ativado; 0 se não;
- z^{ck} *:* fração da demanda d_k que o veículo *c* recolha;
- f_{ij}^{kc} *:* fluxo global no arco *(i, j)* com veículo *c* com destino á zona *k*;
- ct^c *:* custo total da rota do veículo *c* (distância).

Modelo Matemático Versão Estática

$$Min \sum_{c\in C} C^c * y^c + \sum_{c\in C}\sum_{(i,j)\in E} v_{ij}\, x_{ij}^c \quad (1)$$

Sujeito a:

$$\sum_{(1,j)\in E} x_{1j}^c = y^c \qquad \forall c \in C \quad (2)$$

$$x_{ij}^c \le y^c \qquad \forall \quad (i,j) \in E;\ c \in C \quad (3)$$

$$y^{c+1} \le y^c \qquad \forall\, c \in C; \quad (4)$$

$$\sum_{(i,j)\in E} x_{ij}^c - \sum_{(i,j)\in E} x_{ji}^c = 0 \qquad \forall\, j \in V;\ c \in C; \quad (5)$$

$$\sum_{(i,j)\in E} v_{ij} * x_{ij}^c = ct^c \qquad \forall\, c \in C; \quad (6)$$

$$ct^c \le \Gamma \qquad \forall\, c \in C \quad (7)$$

$$\sum_{c\in C} z^{ck} = 1 \qquad \forall k \in K; \quad (8)$$

$$z^{ck} \le y^c \qquad \forall k \in K; \quad c \in C; \quad (9)$$

$$\sum_{k\in K} d^k * z^{ck} \le Cap \qquad \forall\, c \in C \quad (10)$$

$$\sum_{c\in C} y^c \geq \sum_{k\in K} d^k \Big/ Cap \tag{11}$$

$$\sum_{(j,1)\in E} f_{ji}^{kc} - \sum_{(1,j)\in E} x f_{ij}^{kc} = -d^k * z^{ck} \qquad \forall\, k \in K, k > 1;\ c \in C; \tag{12}$$

$$\sum_{(i,k)\in E} f_{ik}^{kc} - \sum_{(k,i)\in E} f_{ki}^{kc} = d^k * z^{ck} \qquad \forall\, k \in K, k > 1;\ c \in C; \tag{13}$$

$$\sum_{(i,j)\in E} f_{ij}^{kc} - \sum_{(j,i)\in E} f_{ji}^{kc} = 0 \qquad \forall\, k \in K, k > 1;\ c \in C;\ j \in V; \tag{14}$$

$$\sum_{k\in K} f_{ij}^{kc} \leq Cap * x_{ij}^{c} \qquad \forall\, (i,j) \in E;\quad c \in C; \tag{15}$$

$$f_{ij}^{kc} \leq d^k * z^{ck} \quad \forall\, (i,j) \in E;\ k \in K;\ c \in C; \tag{16}$$

No modelo, a função objetivo (1) minimiza o número de veículos ativados com um peso pequeno no custo das rotas para evitar sub-ciclos. As restrições (2) e (3) garantem que um veículo ativado sai só uma vez da doca, que apenas veículos ativados andem pelos arcos, e que um veículo atravessa um arco só uma vez na mesma direção. Conjunto (4) garante que os veículos são ativados em ordem. As restrições (5) garantem que um veículo que entra em um dado nó *j* sai do nó *j*. As restrições (6) calculam o custo total (tempo) de cada rota e as restrições (7) garantem que o custo total para cada veículo seja menor do que o custo máximo permitido (nível de serviço). Conjuntos (8) e (9) garantem que toda a demanda seja recolhida por veículos ativados. As restrições (10) limitam o volume recolhido por cada veículo, que não pode superar a capacidade máxima do veículo. As restrições (11) representam um limite inferior no número de veículos exigidos. Os conjuntos (12), (13) e (14) são restrições de conservação de fluxo de produto nos arcos. As restrições (15) e (16) limitam a capacidade de cada rota.

3.2 Problemas Teste

O modelo foi testado em várias instâncias de tamanhos diferentes, com o objetivo de verificar o esforço computacional requerido. Os problemas testes foram criados para armazéns de formato retangular (*grid network*) onde o doca é situado em um canto do armazém. O armazém é dividido em seções pequenas, supondo-se que a demanda de uma seção seja agrupada. Deste modo, um recolhimento do produto *k* seria, na verdade, o recolhimento de todos os produtos na área de produtos *k*. Nos testes todos os arcos são bidirecionais, supondo-se então que os corredores têm espaço suficiente para a passagem dedois veículos de tal maneira que um veículo pode ultrapassar ou cruzar outro. Cada arco leva um custo de 1 unidade. O tempo máximo de serviço para cada instância foi o tempo mais apertado possível, ou seja, o mínimo custo necessário para ir até o ponto mais distante e voltar. Estes problemas usam sempre um armazém com 5 nós de largura (fileiras com 5 nós), mudando só o número de fileiras de acordo com o tamanho do problema. A capacidade dos veículos é de 50 itens.

Os testes foram criados com demanda para 25 nós seguindo uma distribuição uniforme entro 0 e 25. Utilizando a demanda dos 25 pontos, problemas de 10, 15, 20 e 25 nós foram testados, onde o problema de tamanho 10 consiste em buscar a demanda nos primeiros 10 nós. A demanda em cada nó é mostrada abaixo na Tabela 2.

Tabela 2: Demanda presente em cada nó do Conjunto A

nó:	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
demanda:	0	1	3	7	22	2	11	19	6	18	19	25	15	14	20

nó:	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*
demanda:	0	19	11	7	2	4	24	6	19	13

A Figura 4 mostra um exemplo ilustrativo deste tipo de armazém com 20 pontos (19 zonas de coleta, 1 doca). A demanda em cada ponto para este exemplo é mostrada na Tabela 2.

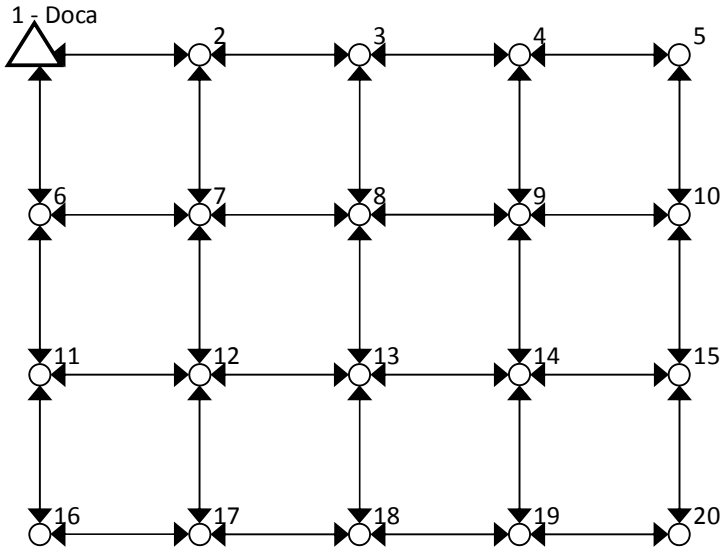

Figura 4. Armazém retangular de 20 pontos

3.3 Resultados do Modelo Estático

O modelo gera as rotas para cada veículo ativado, com o objetivo de minimizar o número de veículos ativados. Para o exemplo de 20 nós, cinco veículos foram ativados, criando as rotas mostradas nas Figuras 5, 6 e 7. A porcentagem da demanda que foi recolhido pelo veículo em cada ponto do armazém é mostrada também nas figuras. Como pode ser visto neste exemplo, todos os veículos começam e terminam no doca, um veículo pode passar por um ponto sem recolher itens, a demanda de um ponto pode ser dividida entre veículos, e a toda a demanda de cada nó é recolhida. Respeita-se também a capacidade dos veículos e o nível de serviço do armazém.

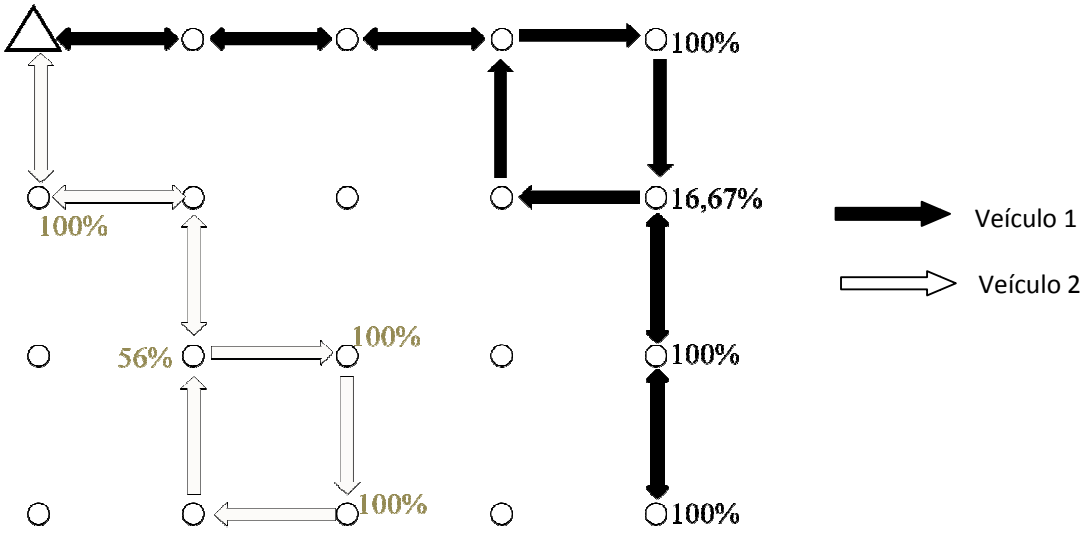

Figura 5. Rotas e Demanda Recolhido pelos Veículos 1 e 2

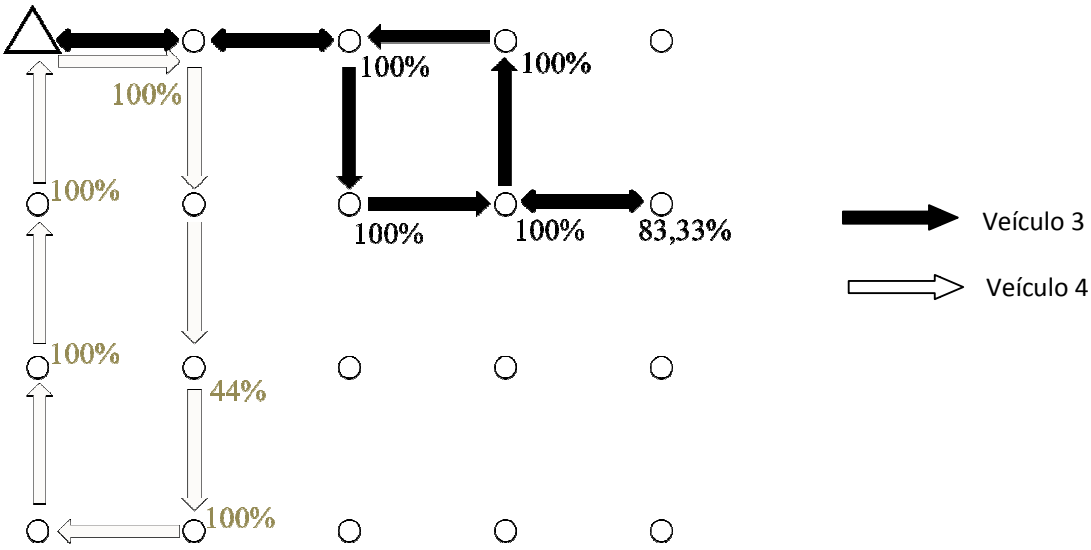

Figura 6. Rotas e Demanda Recolhido pelos Veículos 3 e 4

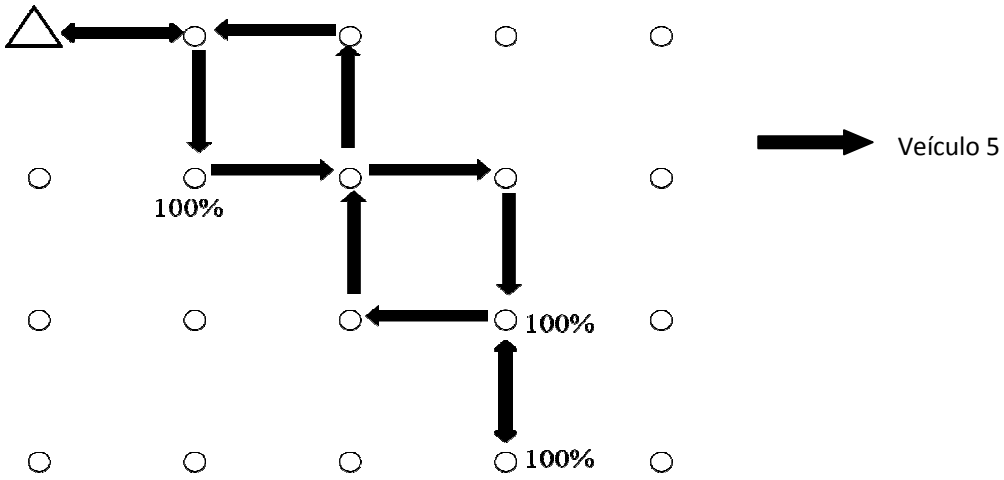

Figura 7. Rotas e Demanda Recolhido por Veículo 5

Os problemas foram testados em um computador com processador Pentium ® de 3.00GHz, 2.00 Gb de memória RAM e sistema operacional Windows XP Professional v.2002. O modelo foi programado com a ferramenta de modelagem AMPL e resolvido com o software de otimização CPLEX.

A Tabela 3 mostra o número de veículos que foram ativados, o tempo de rota de cada veículo, e o tempo computacional para resolver o problema.

Tabela 3. Resultados do modelo estático

Tamanho de Problema	No. De Vcls.	Custo (tempo) da Rota por Veículo						Tempo Computacional (segundos)
		Vcl. 1	Vcl. 2	Vcl. 3	Vcl. 4	Vcl. 5	Vcl. 6	
10	2	8	10	-	-	-	-	1,0
15	4	12	10	10	6	-	-	11,0
20	5	14	10	8	10	12	-	24,0
25	6	12	16	4	12	10	-	278,0

Os problemas de tamanho 10 e 15 foram resolvidos em 1 e 11 segundos, respectivamente. O problema de 20 zonas de coleta utilizou 24 segundos, o de 25 usou 278 segundos.

Capítulo 4

Definição do Problema *On-Line*

O modelo *on-line* trabalha com as rotas geradas anteriormente, ou seja, que estão sendo executadas, e altera essas rotas para incluir novos pedidos. Os produtos a serem recolhidos por um veículo *c* não podem ser retirados da rota do veículo porque é justamente essa rota anterior que vai garantir o nível de serviço para os itens nessa rota. Se por exemplo, deixamos outro veículo pegar a demanda que já foi atribuída, teríamos que modificar a duração deste novo veículo para garantir o nível de serviço daquela demanda. A decisão de não retirar a demanda de um veículo uma vez que foi atribuída permite que o nível de serviço seja garantido por causa do veículo, sem ter que atribuir um horário de chegada para cada item de demanda. A rota do veículo, contudo, pode ser alterada e/ou aumentada para incluir produtos dos pedidos recém-chegados. Com isso, a capacidade e o tempo de sobra dos veículos em rota podem ser aproveitados ao invés de esperar o veículo voltar para a doca para receber uma rota completamente nova. O modelo foi desenhado para, sempre que puder, colocar nova demanda em uma rota existente. O veículo é obrigado a pegar toda a demanda nova possível na zona que ele está no momento das modificações. Quando um item não cabe em qualquer veículo em andamento, por limites de capacidade ou de nível de serviço, o modelo busca ativar um veículo *stand-by* que é um veículo que não foi ativado anteriormente.

4.1 Modelo Matemático – Formulação *On-Line*

Parâmetros

O modelo utiliza alguns dos parâmetros do modelo estático, mas conta com a entrada de vários parâmetros extras. Entre elas estão algumas informações geradas previamente de demanda (demanda original), a rota anterior (usada para garantir o nível de serviço) e a entrada das informações atualizadas, como, por exemplo, a localização atual dos veículos e a capacidade livre. Os parâmetros para o modelo *on-line* seguem.

- *I:* conjunto de locais no armazém;
- *K:* conjunto de zonas de coleta;
- *C:* conjunto de veículos;
- *E:* conjunto de arcos *(i, j)* entre local *i* e *j*;
- y^c *:* Veículos em rota;
- *atv:* número de veículos já mobilizados;
- *stb:* número de veículos *stand-by* disponíveis;
- *n:* número máximo de veículos (limite superior)
- p_{ij}^c*:* arcos *(i,j)* já atravessados pelo veículo *c;*
- do^k*:* demanda original dos produtos (da otimização anterior);
- dn^k*:* demanda nova dos produtos;
- ds^{ck}*:* demanda original que já foi recolhido;
- da^k*:* demanda total atual (demanda original + demanda nova);
- z^{ck}*:* porcentagem da demanda original da zona *k* atribuído ao veículo *c.*
- Cs^c *:* custo de ativação de um veículo *stand-by c (número grande).*
- v_{ij}*:* custo de atravessar arco *(i, j)*;
- *Cap:* capacidade de cada veículo;
- *Γ:* Custo máximo de cada rota – nível de serviço;

Variáveis de Decisão

- x_{ij}^{c} *:* igual a 1 se o veículo c atravessa arco *(i, j)* na continuação da rota; 0 se não;
- g^{ck} *:* fração da demanda dn^k que o veículo *c* recolha;
- f_{ij}^{kc} *:* fluxo global no arco *(i, j)* com veículo *c* com destino á zona *k*;
- s^{c} *:* igual a 1 se o veículo *stand-by* (não ativo) for ativado; 0 se não;
- ct^{c} *:* custo total (distância) da rota do veículo *c.*

Modelo Matemático On-line

$$Min \sum_{c\in C} C^c * s^c + \sum_{c\in C}\sum_{(i,j)\in E} v_{ij}\, x_{ij}^c \tag{17}$$

Sujeito a:

$$\sum_{(1,j)\in E} x_{1j}^c = s^c \qquad \forall c \in C \tag{18}$$

$$x_{ij}^c \le y^c + s^c \qquad \forall \quad (i,j) \in E;\ c \in C \tag{19}$$

$$y^c + s^c \le 1 \qquad \forall\, c \in C; \tag{20}$$

$$\sum_{(i,j)\in E} p_{ij}^c + x_{ij}^c - \sum_{(i,j)\in E} p_{ji}^c + x_{ji}^c = 0 \qquad \forall\, j \in V;\ c \in C; \tag{21}$$

$$\sum_{(i,j)\in E} v_{ij} * x_{ij}^c + \sum_{(i,j)\in E} v_{ij} * p_{ij}^c \le \Gamma \qquad \forall\, c \in C; \tag{22}$$

$$g^{ck} \le y^c + s^c \qquad \forall k \in K; \quad c \in C; \tag{23}$$

$$\sum_{k\in K} dn^k * g^{ck} + \sum_{k\in K} do^k * z^{ck} \le Cap \qquad \forall\, c \in C \tag{24}$$

$$\sum_{c\in C}(y^c + s^c) \ge \left\lceil \sum_{k\in K} d^k \Big/ Cap \right\rceil \tag{25}$$

$$g^{ck} \le \sum_{(i,k)\in E} x_{ik}^c \qquad \forall k \in K, k > 1; \quad c \in C; \tag{26}$$

$$da^k = \sum_{c\in C} g^{ck} * dn^k + z^{ck} * do^k \tag{27}$$

$$\sum_{(j,1)\in E} f_{j1}^{kc} - \sum_{(1,j)\in E} f_{1j}^{kc} = -dn^k * g^{ck} - do^k * z^{ck} \quad \forall\, k \in K, k > 1;\ c \in C; \tag{28}$$

$$\sum_{(i,k)\in E} f_{ik}^{kc} - \sum_{(k,i)\in E} f_{ki}^{kc} = dn^k * g^{ck} + do^k * z^{ck} \qquad \forall\, k \in K, k > 1;\ c \in C; \tag{29}$$

$$\sum_{(i,j)\in E} f_{ij}^{kc} - \sum_{(j,i)\in E} f_{ji}^{kc} = 0 \qquad \forall\, k \in K, k > 1;\ \ c \in C;\ \ j \in V, j > 1; \tag{30}$$

$$\sum_{k\in K} f_{ij}^{kc} \leq Cap * (p_{ij}^{c} + x_{ij}^{c}) \qquad \forall\, (i,j) \in E;\ \ c \in C; \tag{31}$$

$$f_{ij}^{kc} \leq dn^{k} *\ g^{ck} + do^{k} *\ z^{ck} \qquad \forall\, (i,j) \in E;\ \ k \in K;\ \ c \in C; \tag{32}$$

No modelo, a função objetivo (17) minimiza o número de veículos *stand-by* ativados, com um pequeno peso também no custo das rotas para evitar sub-ciclos. Restrições (18) e (19) garantem que um veículo ativado sai só uma vez da doca, que apenas veículos ativados andem pelos arcos, e que um veículo atravessa um arco só uma vez na mesma direção. Conjunto (20) garante que um veículo que já foi ativado não seja contado como um veículo *stand-by*. As restrições (21) garantem que um veículo que entra em um dado nó *j* sai do nó *j*. As restrições (22) calculam o custo total (tempo) de cada rota e garantem que o custo total seja menor do que o custo máximo permitido (nível de serviço). Conjunto (23) garante que toda a demanda seja recolhida por um veículo ativado. As restrições (24) limitam o volume recolhido por cada veículo, que não pode estourar a capacidade máxima do veículo. As restrições (25) representam um limite inferior no número de veículos exigidos. As restrições (26) garantem que a demanda é atribuída a veículos que passam pelo ponto de demanda. As restrições (27) garantem que toda a demanda seja recolhida ou porque já foi atribuído a um veículo na última otimização ou foi atribuído como demanda nova. Os conjuntos (28), (29) e (30) são restrições de conservação de fluxo de produto nos arcos. As restrições (31) e (32) limitam a capacidade de cada rota.

4.2 Problemas Teste

Os problemas testes para o modelo *on-line* foram criados com o mesmo desenho de armazém que foi descrito para o modelo estático, ou seja, um armazém retangular de 5 zonas por fileira e 10, 15, 20 e 25 nós (veja secção 3.2). Usou-se o resultado do modelo estático como entrada para o modelo *on-line*, onde cada veículo já atravessou os primeiros três arcos e recolheu toda a demanda original dos nós onde passou se estava programado para recolher itens naqueles nós. Considera-se que uma nova demanda chega para todos os nós conforme mostrado na Tabela 4.

Tabela 4: Demanda nova presente em cada nó

nó:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
demanda:	0	2	4	19	10	9	11	1	3	4	13	16	1	2	6

nó:	16	17	18	19	20	21	22	23	24	25
demanda:	9	18	10	2	7	20	8	5	18	7

Com o objetivo de mostrar o resultado típico do modelo *on-line*, será examinado o caso do exemplo dado na seção *3.4* do armazém com 20 pontos de coleta. Especificamente, será observada a rota original do veículo 5, que foi mostrado na Figura 7 e, para conveniência, pode ser visto na Figura 8. Como entrada do modelo *on-line*, sabemos que o veículo localiza-se atualmente no nó 8 porque já atravessou os primeiros 3 arcos, seguindo o caminho 1-2-7-8. Sabe-se que ele já recolheu 100% da demanda original de nó 7 (11 unidades de capacidade). Também sabe-se que o veículo ainda vai buscar 100% da demanda original nos nós 14 e 19 (20 unidades de capacidade) já que o modelo não permite tirar demanda do veículo que já foi despachado. Com isso, o veículo vai buscar 31 unidades e tem 19 unidades de capacidade livre sem estourar o limite de 50 unidades. Deve-se lembrar também que existe um limite no tempo da rota total. Neste exemplo o nível de serviço é de 14 unidades de tempo.

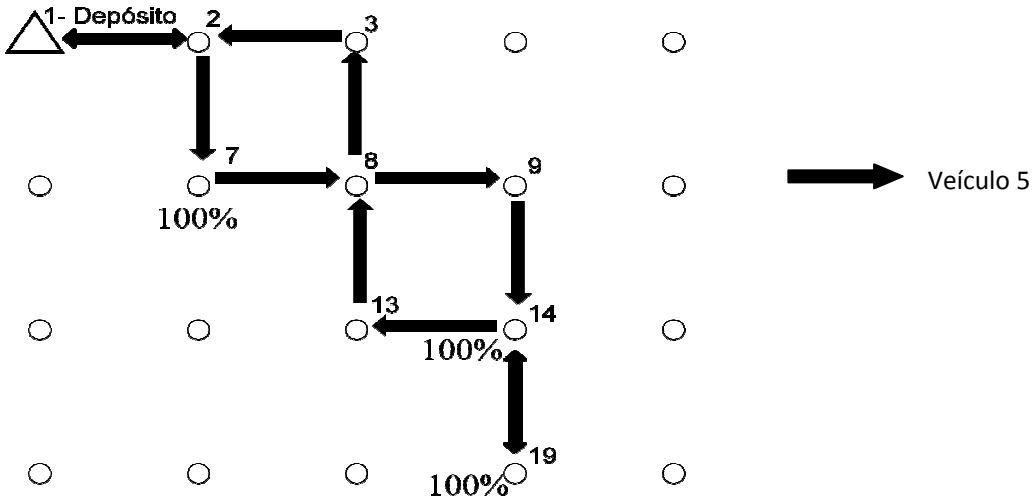

Figura 8. Rota do Veículo 5

O modelo *on-line* utiliza estas informações de capacidade, nível de serviço, e localização atual para decidir se um veículo pode ajudar a recolher demandas novas nas zonas onde já foi programado a passar ou em outras zonas, alterando ou aumentando a rota original para poder fazer isto. No resultado do veículo 5, o veículo teve capacidade para buscar demanda nova em zonas onde ele já estava programado para ir: 100% da demanda nova nas zonas 14,19 e 13 e 67% da demanda na zona 9. Mas mesmo buscando esta demanda nova, o veículo ainda tem capacidade sobrando. A demanda nova do produto 18 (10 unidades) foi aumentada na rota original, mudando a rota original conforme visto na Figura 9. Como pode ser visto, o veículo ainda passa em todos os pontos onde demanda que foi atribuído originalmente para recolher essa demanda.

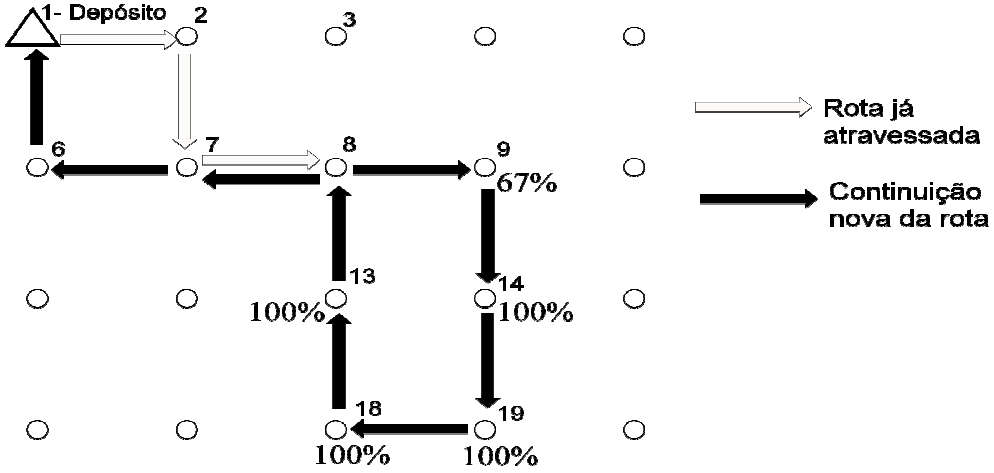

Figura 9. Rota do veículo 5 criada pelo modelo *on-line*, e demanda nova recolhido

Como o modelo estático precisou de 5 veículos e a maioria dos veículo já estavam utilizando quase a capacidade total, o modelo *on-line* precisou ativar mais três veículos para recolher toda a demanda nova.

4.3 Resultados – Modelo *On-Line*

Os problemas testes foram criados com 10, 15, 20 e 25 nós. A Tabela 5 mostra o número de veículos *stand-by* que foram ativados, o tempo computacional para resolver o problema, e a porcentagem de GAP de otimalidade.

Tabela 5. Resultados do modelo *on-line*

Tamanho de Problema	Número de Vcls. *Stand-by* ativados	Tempo CPLEX Computacional (segundos)	GAP (%)
10	2	1	0,0009
15	2	10	0,0060
20	3	136	0,0056
25	4	734	0,0053

Conforme visto na tabela, os tempos computacionais são elevados em comparação ao modelo estático. Como o modelo *on-line* limita o espaço de soluções viáveis consideravelmente, o modelo gasto muito tempo procurando uma solução viável. O problema com 25 zonas de coleta, por exemplo, apresentou um GAP de otimalidade de 0,0053% e o problema com 20 zonas de coleta teve um GAP de otimalidade de 0,0056%. Com este resultado, percebe-se que o modelo tem um limite inferior justo e que gasta muito tempo computacional para encontrar um bom limite superior (solução inteira viável). Para melhorar o tempo de resolução dos problemas, foi introduzida nas secções a seguir novas desigualdades válidas que fortaleçam o limite inferior em casos com onde este limite não é tão justo, e uma heurística que gera um bom limite superior.

Capítulo 5

Nova Classe de Desigualdades Válidas

Um limite inferior para o VRPTWSP, que foi utilizado no modelo de otimização proposta anteriormente, é calculado dividindo a demanda total a ser recolhida pela capacidade do veículo. Este limite inferior, então, tem base na capacidade dos veículos, mas o problema aqui estudado não tem só a capacidade como fator limitante, mas também o prazo de entrega. Até hoje não existe nenhum limite inferior melhor na literatura para o problema com janelas de tempo e coleta fracionada. Para as instâncias em que a capacidade limita o problema mais do que a qualidade de serviço, este limite inferior baseado na capacidade é justo. Trabalhando-se com grandes armazéns de alta rotatividade, isto tende a ser o caso. Mas é possível também ter casos onde o prazo de entrega é o fator limitante. Para estes casos, seria interessante ter um limite inferior com base na janela de tempo.

5.1 Exemplo Ilustrativo - Desigualdades Válidas

Um bom exemplo de tal caso é o problema de dez nós com demanda original e demanda nova conforme visto na Tabela 6. Os veículos têm capacidade de 100 unidades e o nível de serviço é de 10 unidades de tempo, sendo que cada arco demora 1 unidade para ser atravessado.

Tabela 6. Demanda Original e Demanda Nova para exemplo de 10 nós

Nó:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Demanda Original:	0	2	14	33	11	3	1	16	50	6
Demanda Nova:	0	2	5	8	1	4	1	2	6	6

O modelo estático, que atende a demanda original, utiliza 2 veículos. O primeiro veículo busca a demanda toda dos nós 2, 3, 7, e 8 e 27,27% da demanda no nó 4, seguindo a rota mostrado na Figura 10. Somando, a demanda original que o veículo 1 vai buscar é 42 unidades, deixando 58 unidades de capacidade extra que podia ser utilizada para demanda nova.

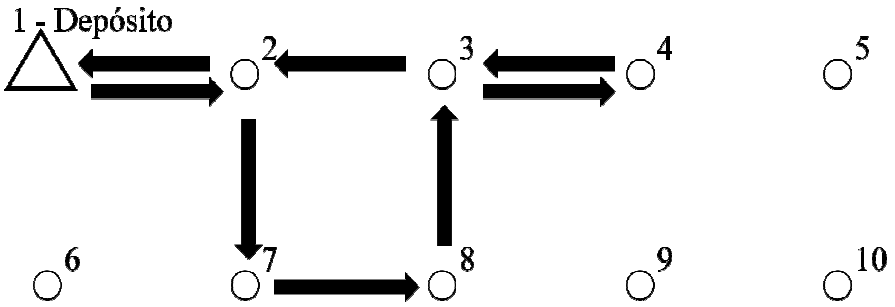

Figura 10 . Rota Original do Veículo 1

O veículo 2 busca 100% da demanda dos nós 5, 6, 9, e 10, e 72,72% da demanda no nó 4, seguindo a rota mostrado na Figura 11. O total da demanda original buscado pelo veículo 2 é 94 unidades, deixando 6 unidades de capacidade livre.

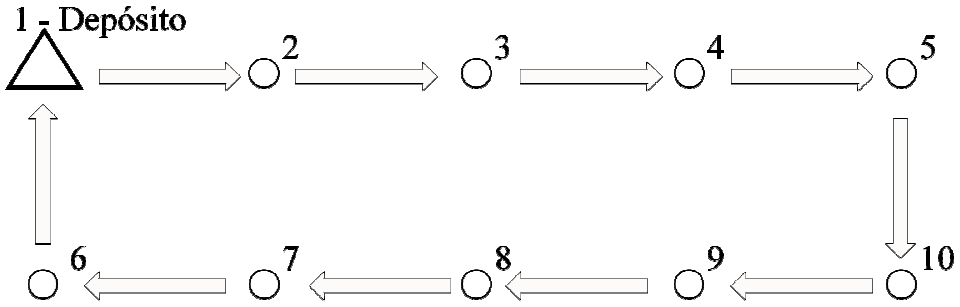

Figura 11. Rota Original do Veículo 2

Suponha agora que a demanda nova chega e o modelo on-line é rodado quando o veículo 1 está no nó 3 (já buscou a demanda original nos nós 2, 7, 8, e 3) e o veículo 10

está no nó 10 (já buscou a demanda original nos nós 5 e 10). Neste momento o veículo 1 precisa ainda buscar demanda original no nó 4 e tem 58 unidades de capacidade e o veículo 2 precisa buscar demanda original nos nós 6 e 9, e tem 6 unidades de capacidade sobrando para buscar a demanda nova.

A soma da demanda nova é de 39 unidades, o que caberia no veículo 1, e, portanto, o limite inferior baseado na capacidade diria que não há necessidade de mais um veículo para buscar a demanda nova. Quando se considera o nível de serviço, porém, observa-se que o veículo 1 não tem folga suficiente para ir ao nó 10 e voltar para o doca. Isto possibilita criar um desigualdade válida que corta a possibilidade do veículo 1 visitar o nó 10. Se a demanda nova no nó 10 fosse maior que a capacidade extra do veículo 2, este corte seria o suficiente para perceber que precisa de mais um veículo. Mas as 6 unidades de demanda nova no nó 10 neste exemplo cabem no veículo 2, e então não limite a solução. Neste caso, é necessário ir um pouco mais além. Observando os pontos de demanda não somente um por um, mas em pares, nota-se que existem *pares* de nós que não combinam. Por exemplo, o veículo 1 pode ir ao nó 5 *ou* pode ir ao nó 6, mas não tem folga no nível de serviço suficiente para visitar o nó 5 *e* o nó 6. Utilizando esta informação pode-se montar um grafo de conflito que gera desigualdades válidas do tipo:

$$\sum_{i \in I} x_{i5}^{1} + \sum_{i \in I} x_{i6}^{1} \leq 1 \qquad (33)$$

que limita soluções nas quais o veículo vai nos dois nós.

Para o veículo 1 deste exemplo, podem-se adicionar restrições ao modelo que cortam o nó 10 e os pares (5,6), (5,7), (5,8), (5,9). Para o veículo 2, podem-se adicionar restrições para os pares (6,2), (6,3), (6,4), (6,5), (7,3), (7,4), (7,5), (8,4), (8,5), e (9,5). Sem estas novas restrições, seria impossível detectar a necessidade de um novo veículo.

Assim, de uma forma geral, uma matriz de caminhos mínimos deve ser montado, utilizando por exemplo o algoritmo de Floyd-Warshall em $O(n^3)$ para determinar a possibilidade de alcançar os nós com base na localização e prazo de entrega restante de cada veículo. A partir dos cálculos de caminho mínimo, pode-se achar os nodos não alcançáveis e os pares de nodos não alcançáveis. A forma geral das desigualdades válidas para estes dois casos pode ser descrito da seguinte forma:

(a) Para Nodos Não-Alcançáveis:

Seja $Q \in K$ o conjunto dos nodos não alcançáveis onde há demanda nova. A forma geral das desigualdades válidas é:

$$\sum_{i \in I} x_{iq}^{c} \leq 1, \ \forall\, c \in C, \ \ q \in Q \in K \qquad (34)$$

(b) Para Pares de Nodos Não-Alcançáveis:

Seja $R \in K * K = \{(s,t) | s\ e\ t\ não\ são\ alcançáveis\}$ a relação binária que determina pares de nodos não alcançáveis. A forma geral das desigualdades válidas derivadas seria:

$$\sum_{i \in I} x_{is}^{c} + \sum_{i \in I} x_{it}^{c} \leq 1, \ \forall\, c \in C, \ \ (s,t) \in R \qquad (35)$$

5.2 Algoritmo para Desigualdades Válidas

Com base nestas desigualdades, pode-se propor um algoritmo de melhora de limites inferiores da forma descrito a seguir:

Passo 1: Gerar desigualdades que incorporem informação acerca de nodos não-alcançáveis para cada veículo. O número máximo de cortes gerados será $n * |v|$, onde n representa o número de veículos e v o vetor de nós (locais). Adicionar estas desigualdades ao programa linear e verificar se há resíduo no vetor s (vetor de veículos *stand-by*) superior a θ, onde θ é o ponto de corte. Este ponto de corte é uma decisão heurística que seria determinada baseada na aplicação.

- Se há resíduo superior a θ, pare, e substitua as desigualdades geradas por
  $$\sum s \geq \lceil \sum s_{PL} \rceil \quad (36)$$
  onde s_{PL} é a solução de programação linear.
- Se não, vá ao passo 2.

Passo 2: Monte um grafo de conflito 2 a 2, que determina quais pares de nodos não são alcançáveis. Incorpore esta informação ao modelo, com desigualdades na forma de (34). O número máximo de cortes gerados será $n * |v| * |v|$. Adicione estas desigualdades ao programa linear e verifique se há resíduo no vetor s superior a θ.

- Se há resíduo superior a θ, pare, e substitua as desigualdades geradas por (36) onde s_{PL} é a solução de programação linear.
- Se não, vá ao passo 3.

Passo 3:

- Se *(limite superior – limite inferior)/limite superior* < *5%* pare.
- Senão dispare o CPLEX.

Espera-se com este algoritmo computar limites inferiores que sejam mais justos em situações onde o principal limitante seja o nível de serviço e não a capacidade dos veículos. É importante salientar que estas situações são menos prováveis á medida que aumenta o tamanho dos armazéns, uma vez que a capacidade dos veículos não cresce na mesma proporção. Considerando-se o ambiente de aplicação de tais modelos, deve-se notar que o nível de serviço deve ser suficiente para garantir que um veículo possa

alcançar a zona mais remota do armazém somada a uma folga responsável por acomodar tempos de coleta e eventuais falhas de equipamento. É nesse contexto que se argumenta que o ponto de corte θ (ocupação mínima do veículo *stand-by* mobilizado para atender nova demanda) deve ser calibrado de acordo com as particularidades de cada aplicação e armazém. Para efeito dos testes aqui realizados, connsidera-se θ = 0,10, mas entende-se que ele deva ser maior para o uso destas técnicas em aplicações reais.

Também nota-se que no algoritmo aqui apresentado o grafo de conflito foi montado apenas 2 a 2. Em casos onde o armazém é grande e o nível de serviço é um fator limitante, pode ser desejável aumentar este grafo de conflito e examinar os nodos 3 a 3. Neste caso pode-se adicionar mais um passo ao algoritmo, onde o número máximo de cálculos seria $n * |v| * |v| * |v|$.

5.3 Resultados Exemplo Ilustrativo – Desigualdades Válidas

Referindo ao exemplo ilustrativo apresentado na secção 5.1, modificou-se o algoritmo para incorporar as desigualdades válidas. No primeiro passo foi incorporada informação acerca de nós não-alcançáveis para cada veículo. Neste exemplo o único nó não alcançável é o nó 10 pelo veiculo 1. Foi adicionado ao programa linear então a seguinte restrição.

$$\sum_{i \in I} x_{i,10}^{1} \leq 0 \qquad (37)$$

Mas deve-se lembrar que neste exemplo ocorte sozinho não é o suficiente porque o veículo 2 tem capacidade suficiente para buscar a demanda no nó 10. A solução da relaxação de programação linear, então, não tem resíduo no vetor de veículos *stand-by*.

Continuando então para o passo 2, monta-se o grafo de conflito. Neste exemplo são adicionados 14 restrições do tipo (33) referentes aos pares de nodos não alcançáveis. A solução do programa linear, com a adição destas 14 desigualdades válidas, mostra um resíduo no vetor de veículo *stand-by* de 0,50. Já que 0,50 é maior que o valor de θ de 0,10, sabe-se então que um veículo *stand-by* será necessário, o que permite adicionar a restrição (38) à programação inteira, melhorando o limite inferior.

$$\sum_{c \in C} s_c \geq 1 \qquad (38)$$

Capítulo 6

Heurística para o Problema *On-Line*

O problema *on-line* aproveita os avanços da tecnologia que permitam a comunicação rápida com os veículos em andamento e o uso de dados atualizados no modelo. Para este modelo ser aplicável na prática, é muito importante ter rapidez na resolução do modelo, senão o seu estado já terá mudado na hora de passar as novas informações para os veículos. O modelo mostrou tempos de resolução de até 734 segundos (para o teste com 25 nós), o que provavelmente não seria aceitável para o modelo ser usado *on-line*.

Os resultados do modelo mostram que existe uma formulação forte, na qual a maioria do tempo de resolução é gasto procurando uma solução viável. Uma vez que uma solução viável é encontrada, o modelo consegue achar o ótimo rapidamente. O que é necessário então, é uma heurística para achar, rapidamente, uma boa solução viável.

Uma heurística com base nas rotas originais foi desenvolvida. A heurística trabalha primeiro com os veículos já ativados. A rota original, antes de chegar a demanda nova, é dada como entrada. Cada veículo é obrigado a pegar toda a demanda nova que puder no nó onde está no momento. Depois, as demandas novas são avaliadas, e as demandas novas que ficam mais pertos da rota são escolhidas. Isto quer dizer, então, que o veículo vai tentar pegar primeiro as demandas nos nós onde já estava programado para ir. Se, depois de alocar estas demandas novas, o veículo ainda tiver disponibilidade de espaço, os nós que ficam mais próximos da rota serão avaliados.

Para o veículo sair da rota original para pegar uma demanda nova, ele tem que ter capacidade física para pegar essa demanda nova (ou uma fração da demanda) e também precisa ter folga no nível de serviço suficiente para pegar a demanda e voltar para a rota original. Para cada zona com demanda nova, a heurística determina qual o nó da rota original mais perto, e a demanda que seria buscada a partir desse nó. Considera a rota original mostrada na Figura 12. O nível de serviço neste exemplo é 14 e a rota original

gasta 10 unidades de tempo, então tem 4 sobrando. Este veículo vai buscar demanda nova na sua própria rota primeiro, e depois nos nós mais pertos. Suponha que o veículo ainda tem capacidade física, e que a demanda nova mais perto da rota original encontra-se no nó 20. A heurística vai determinar qual nó é mais perto, neste caso o nó 18, e se o veículo tem folga para ir até aquele nó e voltar. Se tiver, esta extensão na rota original é aceita e o veículo pode buscar a demanda.

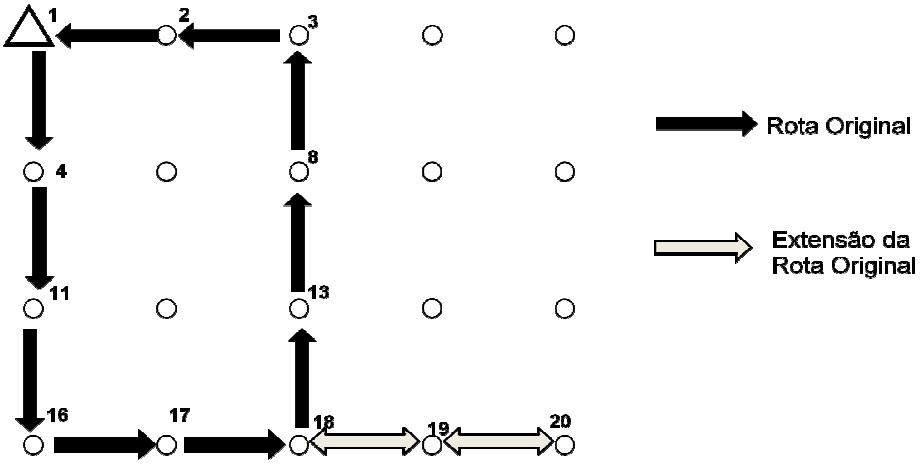

Figura 12. Exemplo de Rota - Heurística

Se sobrar demanda nova depois que todos os veículos já ativados tenham buscado tudo que podiam, os veículos *stand-by* precisam ser ativados. Para determinar a rota dos veículos *stand-by*, a heurística baseia-se no algoritmo de Clark e Wright (veja secção 2.1.1). O algoritmo de Clark e Wright foi escolhido por causa da simplicidade de implementação em comparação com as outras heurísticas. A versão seqüencial do algoritmo de Clark e Wright foi implementada já que o objetivo é minimizar o número de veículos e a versão seqüencial prioriza exatamente isso. A heurística tenta, então, encher todo o veículo antes de começar uma rota nova com outro veículo.

Os resultados desta heurística nos problemas testes do modelo *on-line* mostraram resultados ótimos. A heurística achou o ótimo para todos os testes (de tamanhos 10, 15, 20, e 25) com menos de 1 segundo. Isto quer dizer que o limite inferior e superior são iguais. Para os problemas testes aqui apresentados, então, não é necessário rodar o

modelo de otimização para achar o ótimo. Para uso na prática, no entanto, seria necessário usar esta solução como entrada para o modelo nos casos em que o limite inferior e o limite superior não são iguais. Espera-se que com o uso desta heurística para gerar limites superiores e uma solução viável, a formulação forte aqui proposta tenha tempo computacional reduzido.

Capítulo 7

Conclusões

Neste trabalho, duas formulações para o problema de roteamento de veículos com coleta fracionada e janelas de tempo foram introduzidos. A primeira, o modelo estático, considera o estado inicial, na qual todos os veículos encontram-se estacionados no doca. O resultado deste modelo para a maior instância testada, de 25 nós, foi de 278 segundos. O modelo é uma nova formulação para a literatura.

O segundo modelo introduzido foi o modelo *on-line*, que considera que os veículos já se encontram em andamento e podem mudar ou aumentar as rotas para acomodar demanda nova. Este modelo tem um espaço de soluções limitado porque o começo da rota e alguns nós onde o veículo ainda precisa buscar demanda original já foram determinados. Considerando isto, o tempo computacional do modelo é maior, com o exemplo de 25 nós utilizando 734 segundos. Bastante tempo é utilizado na busca de uma solução viável. É muito importante que o modelo on-line seja resolvido em pouco tempo porque o resultado é baseado nas informações atualizadas.

Com a intenção de melhorar o desempenho do modelo *on-line* introduziram-se novas desigualdades válidas para fortalecer o limite inferior nos casos em que o nível de serviço é o fator limitante, e não a capacidade física. As desigualdades são baseadas na avaliação de nós não alcançáveis e pares de nós não alcançáveis, criando um grafo de conflito. Desenvolveu-se um algoritmo para o uso destas desigualdades válidas.

Para melhorar o limite superior, uma heurística foi desenvolvida. A heurística coloca primeiro a demanda nova nos veículos já ativados, priorizando as demandas novas mais pertos da rota original. O veículo continua na rota original, mas uma extensão pode ser adicionada a esta rota. As rotas para os veículos *stand-by*, se foram necessários, são determinados com a versão seqüencial do algoritmo de Clark e Wright. Para os problemas teste a heurística conseguiu resolver o problema com menos de 5 segundos e achar a solução ótima.

Com um novo modelo on-line, desigualdades válidas que ofereçam um limite inferior justo quando o fator limitante é o prazo de entrega, e uma heurística que cria soluções viáveis e, no caso dos testes aqui apresentados até ótimas, o problema tem um bom começo. Mas ainda tem muito que pode ser estudado com respeito a este problema. Seria muito interessante estudar este modelo com dados reais para ver sua aplicação na prática. Para se fazer isto, também teria que, provavelmente, acrescentar um tempo de coleta, que aqui foi considerado pequeno em comparação ao tempo de transtorno. Seria especialmente interessante fazer uma simulação utilizando o modelo para estudar o comportamento com tempo e como calibrar o modelo. Pode-se avaliar o caso de qual seria a melhor opção: disparar o modelo on-line e ativar novos veículos ou esperar o retorno dos veículos para então realocá-los. Esta seria uma abordagem alternativa interessante para o problema.

Referências Bibliográficas

AARTS, E. H. L.; e LENSTRA, J. K. ***Local Search in Combinatorial Optimization***. Editora John Wiley & Sons Ltd. 1997. p. 1-17

ACHUTHAN, N.R.; CACCETTA, L.; HILL, S.P. An Improved Branch-and-Cut Algorithm for the Capacitated Vehicle Routing Problem. **Transportation Science**, vol.37, n.2, p.153-169, 2003.

ARCHETTI, C; MANSINI, R.; SPERANZA, M.G. Complexity and Ruducibility of the Skip Delivery Problem. **Transportation Science**, vol. 39, n. 2, p. 182-187, 2005.

ARCHETTI, C; SPERANZA, M.G.; HERTZ, A. A Tabu Search Algorithm for the Split Delivery Vehicle Routing Problem. **Transportation Science**, vol. 40, n. 1, p. 64-73, 2006.

ARCHETTI, C; SAVELSBERGH, M.W.P.; SPERANZA, M.G. Worst-Case Analysis for Split Delivery Vehicle Routing Problems. **Transportation Science**, vol. 40, n. 2, p. 226-234, 2006.

ASCHEUER, N.; GROTSCHEL, M.; ABDEL-HAMID, A.A-A. Order Picking in an Automatic Warehouse: Solving Online Asymmetric TSPs. **Mathematical Methods of Operations Research**, vol. 49, p. 501-515, 1999.

BADEAU, P.; GUERTIN, F.;GENDREAU, M.; POTVIN, J.; TAILLARD, E. A Parallel Tabu Search Heuristic For the Vehicle Routing Problem with Time Windows. **Transportation Research Part C: Emerging Technologies**, vol. 5, n. 2, p. 109-122, 1997.

BALLOU, R.H. **Business Logistics Management:** Planning, Organizing and Controlling the Supply Chain. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1999, 816 p.

BARBAROSOGLU, G.;OZGUR, D. A Tabu Search Algorithm for the Vehicle Routing Problem. **Computers & Operations Research**, vol. 26, p. 255-270, 1999.

BARD, J.F.; KONTORAVDIS, G.; GANG, Y. A Branch-and-Cut Procedure for the Vehicle Routing Problem with Time Windows. **Transportation Science,** vol. 36, n. 2, p. 250-269, 2002.

BAKER, B.M.; AYECHEW, M.A. A Genetic Algorithm for the Vehicle Routing Problem. **Computers & Operations Research**, v.30, p. 787-800, 2003.

BELENGUER, J.M.; MARTINEZ, M.C., MOTA, E. A Lower Bound for the Split Delivery Vehicle Routing Problem. **Operations Research,** vol. 48, n. 5, p. 801-810, 2000.

BELFIORE, P.P. **Scatter Search Para Problemas de Roterização de Veículos com Frota Heterogênea, Janelas de Tempo e Entregas Fracionadas**. São Paulo, 2006. 203 p. Tese (Doutor em Engenharia) - Escola Politécnica, Universidade de São Paulo, 2006.

CAMPOS, V.; MOTA, E. Heuristic Procedures for the Capacitated Vehicle Routing Problem. **Computational Optimization and Applications**, vol. 16, n. 3, p. 265-277, 2000.

CHEN , Z.-L.; POWELL, W.B. Solving parallel machine scheduling problems by column generation. **INFORMS Journal on Computing**, v. 11, p. 78-94, 1999.

CONCEIÇÃO, S.V. "Modelos e Heurísticas para os Problemas de Dimensionamento de Frota e Roteirização de Veículos", Notas de aula, Escola de Engenharia da UFMG, 2006.

CORDEAU, J-F.; GENDREAU, M.; LAPORT, G.; POTVIN, J-Y; E SERNET, F. A Guide to Vehicle Routing Heuristics. **Journal of the Operational Research Society**, v. 53, p. 512-522, 2002.

DANIELS, R.L.; RUMMEL, J.L.; SCHANTZ, R. A Model for Warehouse Order Picking. **European Journal of Operational Research**, vol. 105, p. 1-17, 1998.

DE KOSTER, R.; VAN DER POORT, E. Routing orderpickers in a warehouse: A comparison between optimal and heuristic solutions. **IIE Transactions**, v. 30, p. 469-480, 1998.

DESROSIERS, J.; SAUVÉ, M.; SOUMIS, F. Lagrangian Relaxation Methods for Solving the Minimum Fleet Size Multiple Traveling Salesman Problem with Time Windows. **Management Science**, v.34, n. 8, p. 1005-1022, 1988**.**

DESROCHERS, M.; DESROSIERS, J.; SOLOMAN, M. A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. **Operations Research,** vol. 5, n. 2, p. 109-122,vo. 40, n. 2, p.342-354, 1992.

DROR, M.; LAPORTE, G.; TRUDEAU, P. Vehicle Routing with Split Deliveries. **Discrete Applied Mathematics,** vol. 50, p. 239-254, 1994.

DROR, M.; TRUDEAU, P. Split Delivery Routing. **Naval Research Logistics,** vol. 37, p. 383-402, 1990.

DROR, M.; TRUDEAU, P. Savings by Split Delivery Routing**. Transportation Science**, vol. 23, p. 141-145, 1989.

FRIZZELL, P.W.; GRIFFIN, J.W. The Split Delivery Vehicle Scheduling Problem with Time Windows and Grid Network Distances. **Computers & Operations Research,** vol. 22, n. 6, p. 655-667, 1995.

GENDREAU, M.; HERTZ, A.; LAPORTE, G. A Tabu Search Heuritic for the Vehicle Routing Problem. **Management Science,** vol. 40, n. 10, p. 1276-1290, 1994.

GENDREAU, M.; LAPORTE, G.; e POTVIN, J-Y. "Vehicle routing: Modern heuristics", **Local Search in Combinatorial Optimization**. Editor John Wiley & Sons Ltd. p. 1-17. 1997.

GILLETT, B.E.; MILLER, L.R. A Heuristic Algorithm for the vehicle dispatch problem. **Operations Research**, vol. 12, p. 568-581, 1964.

GOLDBARG, M.C.; LUNA, H.P.L. **Otimização Combinatória e Programação Linear:** Modelos e Algoritmos. Editora Campus, 2000, 649p.

GU, J., GOETSCHALCKX, M.; MCGINNIS, L.F. Research on warehouse operation: A comprehensive review. **European Journal of Operational Research***,* v. 177, p. 1-21, 2007.

HERTZ, A.; TAILLARD, E.; DE WERRA, D. "Tabu search", **Local Search in Combinatorial Optimization.**. Editora John Wiley & Sons Ltd. 1997. p. 121-136.

HO, S.C.; HAUGLAND, D. A Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows and Split Deliveries. **Computers & Operations Research,** vol. 31, p. 1947-1964, 2004.

JIN, M.; LIU, K.; BOWDEN, R.O. A Two-Stage Algorithm with Valid Inequalities for the Split Delivery Vehicle Routing Problem. **International Journal of Production Economics,** vol. 105, p. 228-242, 2007.

KELLY, J.P.; XU, J. A Set-Partitioning-Based Heuristic for the Vehicle Routing Problem. **INFORMS Journal on Computing,** vol. 11, n. 2, p. 161-172, 1999.

KIRKPATRICK, S.; GELATT, C.D Jr.; VECCHI, M.P. Optimization by Simulated Annealing. **Science**, vol. 220, n. 4598 p. 671-680, 1983.

KOHL, N.; MADSEN, O.B.G. An Optimization Algorithm for the Vehicle Routing Problem with Time Windows Based on Lagrangian Relaxation. **Operations Research,** vol. 45, n. 3, p. 395-406, 1997.

LAU, H.C.; SIM, M.; TEO, K.M. Vehicle Routing Problem with Time Windows and a Limited Number of Vehicles. **European Journal of Operational Research,** v. 148, p. 559-569, 2003.

MULLASERIL, P.A.; DROR, M.; LEUNG, J. Split-Delivery Routeing Heuristics in Livestock Feed Distribution. **Journal of the Operational Research Society,** vol. 48, p. 107-116, 1997.

OSMAN, I.H. Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem. **Annals of Operations Research**, vol. 41, p. 421-451, 1993.

PETERSON, C.G.; AASE, G. A comparison of picking, storage, and routing policies in manual order picking. **International Journal of Production Economics**, vol. 92, p. 11-19, 2004.

ROCHAT Y. e TAILLARD, E.D. Probabilistic diversification and intensification in local search for vehicle routing. **Journal of Heuristics**. vol. 1, p. 147-167, 1995.

ROODENBERG, K.J.; DE KOSTER, R. Routing Order Pickers in a Warehouse with a Middle Aisle. **European Journal of Operational Research**, vol. 133, p. 32-43, 2001.

SARIKLIS, D.; POWELL, S. A Heuristic Method for the Open Vehicle Routing Problem. **Journal of the Operational Research Society,** v. 51, p. 564-573, 2000.

SIMCHI-LEVI, D.; CHEN, X.; BRAMEL, J. **The Logic of Logistics**, Theory, Algorithms, and Applications for Logistics and Supply Chain Management. New York, NY, USA: Springer, 2005, 366 p.

TAN, K.C.; LEE, L.H.; ZHU, Q.L.; OU, K. Heuristic Methods for Vehicle Routing Problem with Time Windows. **Artificial Intelligence in Engineering,** vol. 15, p. 281-295, 2001.

THOMPSON, P.M.; SPARAFTIS, H.N. Cyclic Transper Algorithms for Multivehicle Routing and Scheduling Problems. **Operations Research,** vol. 41, n. 5, p. 935-946, 1993.

VAIDYANATHAN, B.S.; MATSON, J.O.; MILLER, D.M.; MATSON, J.E. A Capacitated Vehicle Routing Problems for Just-in-Time Delivery. **IIE Transactions,** vol. 31, n. 11, p. 1083-1092, 1999.

WREN, A. *Computers in Transport Planning and Operation*. Ian Allan: London. 1971. 152 p.

WREN, A.; HOLLIDAY, A. Computer scheduling of vehicles from one or more depots to a number of deliver points. **Operations Research Quarterly**, vol. 23, p. 333-344, 1972.

YELLOW, P. A computational modification to the savings method of vehicle scheduling. **Operations Research Quarterly**, vol. 21, p. 281-283, 1970.

YOON, C. S.; SHARP, G. P. A Structured Procedure for Analysis and Design of Order Pick Systems. **IIE Transactions**, v. 28, p. 379–389, 1996.

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

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo