HUMBERTO CÉSAR BRANDÃO DE OLIVEIRA

ALGORITMO EVOLUTIVO NO TRATAMENTO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM JANELA DE TEMPO

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para a obtenção do título de Bacharelado em Ciência da Computação.

Orientador
Prof. Guilherme Bastos Alvarenga

Lavras
Minas Gerais - Brasil
2005

HUMBERTO CÉSAR BRANDÃO DE OLIVEIRA

ALGORITMO EVOLUTIVO NO TRATAMENTO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM JANELA DE TEMPO

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para a obtenção do título de Bacharelado em Ciência da Computação.

Aprovada em *20 de Janeiro de 2005*

Prof. Fortunato Silva de Menezes

Prof. Ricardo Martins de Abreu Silva

Prof. Guilherme Bastos Alvarenga
(Orientador)

Lavras
Minas Gerais - Brasil

Sumário

Lista de Figuras

Lista de Tabelas

Dedico este trabalho à minha família, Humberto, Cristina, Marina e Bruna, e a minha namorada, Mariane, que por muitas vezes me ajudaram nos momentos difíceis. Amo vocês.

Agradecimentos

Agradeço a todos que me ajudaram durante o período do curso de Ciência da Computação. Sem a ajuda de vocês com certeza eu não teria conseguido. Ao meu pai maior, muito obrigado.

Resumo

Problemas de Roteamento de Veículos vêm sendo cada vez mais estudados para obter uma maior economia nos gastos com o transporte de pessoas e mercadorias. Em específico, o Problema de Roteamento de Veículos com Janela de Tempo (*PRVJT*) tem a particularidade de considerar o tempo de disponibilidade dos consumidores em suas restrições, tratando assim uma particularidade bastante comum nos problemas de coleta ou entrega do mundo real. Usando a distância total como principal objetivo, este trabalho implementa um eficiente algoritmo evolutivo no tratamento do *PRVJT*.

Palavras-chave

problema de roteamento de veículos, *PRV*, problema de roteamento de veículos com janela de tempo, *PRVJT*, otimização combinatória, pesquisa operacional, algoritmo evolutivo, algoritmo genético.

Abstract

Vehicle Routing Problem has been extensively analysed to get save on money spend, with people and merchandise transport. Specifically, Vehicle Routing Problem with Time Windows (*VRPTW*) has the particular way of dealing the time available of the customers in their restrictions, treating in this way, a very common particularitity in pick-up or delivery problems in real world. Using the total distance like a main objective, this work implements an efficient evolutive algorithm in treatement of the *VRPTW*.

Key-words

vehicle routing problem, *VRP*, vehicle routing problem with time windows, *VRPTW*, combinatorial optimization, operational research, evolutive algorithm, genetic algorithm.

Capítulo 1

Introdução

1.1 Considerações Iniciais

Custos com o transporte de mercadorias vêm ganhando uma atenção especial nas últimas décadas onde a minimização dos gastos é um grande foco para empresas que visam sobreviver e crescer no mercado competitivo atual. Os resultados da literatura vêm contribuindo de maneira significante no Problema de Roteamento de Veículos (*PRV*). Como este problema no mundo real apresenta uma enorme dinamicidade, foi proposta a subdivisão desta classe de problemas a fim de se estudar de maneira específica algumas destas situações presentes em instâncias reais. Uma delas é a consideração da capacidade de carga do veículo e o tempo em que os consumidores podem ser atendidos (janela de tempo), tendo apenas um depósito central de mercadorias, ficando conhecido na literatura como Problema de Roteamento de Veículos com Janela de Tempo (*PRVJT*).

Segundo Larsen [Larsen (1999)], os custos relacionados ao transporte de pessoas e mercadorias é muito grande, com tendência ao crescimento, motivado pela expansão atual das fronteiras comerciais de todo tipo de negócio. Pesquisas sugerem que de 10% a 15% do valor final das mercadorias comercializadas correspondem ao custo de seu transporte [Fisher et.Al. (1997)]. Segundo Bodin [Bodin et.Al. (1983)], o custo de distribuição nos Estados Unidos em 1980 foi estimado em 400 bilhões de dólares. Uma parcela destes custos poderia ser reduzida com o tratamento de

diversos problemas de roteamento, onde o *PRVJT* tem uma importante contribuição.

1.2 Objetivos e Justificativas

O objetivo deste trabalho é resolver o Problema de Roteamento de Veículo com Janela de Tempo através do Algoritmo Genético, criando novos e aproveitando bons operadores presentes na literatura.

A escolha do algoritmo genético se deu pelo fato do mesmo possuir bons resultados tanto para problemas de roteamento de veículos quantos para outros problemas da classe $\mathcal{NP}$-difícil.

O Problema de Roteamento de Veículos com Janela de Tempo é bastante estudado na literatura quando sua minimização é a priori o número de veículos, e posteriormente considerando a distância total percorrida. Um dos objetivos que merece um estudo mais aprofundado é da minimização da distância total em primeira instância, visto que este pode ser o objetivo desejado por empresas de transporte. Esta é a justificativa deste trabalho atacar em especifico a distância total percorrida do *PRVJT* como objetivo principal da minimização.

1.3 Escopo do Trabalho

Este trabalho descreve uma visão geral do problema de roteamento de veículos (*PRV*) na seção 2.2 e a sua variação com restrição no tempo de atendimento aos consumidores (Problema de Roteamento de Veículos com Janela de Tempo - *PRVJT*) na seção 2.3, bem como conhecer algumas meta-heurísticas como algoritmos genéticos (seção 2.6.6), busca tabu (seção 2.6.5), *Simulated anneling* (seção 2.6.4) e também heurísticas construtivas (seção 2.6), algoritmos exatos (seção 2.4) e heurísticas de aproximação (seção 2.5).

A seção 3 descreve o que foi implementado neste trabalho, subdividido em:

- Algoritmo Genético (seção 3.2);

- População Inicial (seção 3.1.1);
- Seleção (seção 3.1.2);
- Operador de Cruzamento (seção 3.1.3);
- Operadores de Mutação (seção 3.1.4).

Na seção 4 são apresentados os resultados encontrados pelo algoritmo implementado neste trabalho, sendo sub-divididos em resultados em cada instância testada (seção 4.1), comparação com outros algoritmos em cada classe de instâncias (seção 4.2), e melhores resultados da literatura (seção 4.3).

A conclusão e proposta para futuros trabalhos podem ser encontradas respectivamente nas seções 5 e 6.

Capítulo 2

Referencial Teórico

2.1 Otimização

Como citado por Andrade em [Andrade et.Al. (2004)], uma instância de um problema de otimização, segundo Papadimitriu [Papadimitriou et.Al. (1982)], consiste no par (F, c), onde F é um conjunto qualquer, constituído pelos pontos viáveis, e c é uma função de custo, um mapeamento:

$$c : R \mapsto R^1 \tag{2.1}$$

O problema consiste em encontrar um $f \in F$ para cada

$$c(f) \leq c(y) \forall y \in F \tag{2.2}$$

Cada ponto f é denominado ótimo global para a instância, e é comumente referenciado por ótimo.

Desta forma, um *Problema de Otimização* estende-se ao conjunto I de instâncias de um problema de otimização. Os principais constituintes de um problema de otimização são:

- Vizinhança: dado um ponto viável $f \in F$ num determinado problema com instâncias (F, c), sua vizinhança consiste no mapeamento 2.3 definido para cada instância.

$$N : F \mapsto 2^F \tag{2.3}$$

- Ótimo Local e Global: em certas instâncias de problemas, encontrar uma solução ótima pode ser uma tarefa impossível do ponto de vista computacional. Nesses casos, há como encontrar uma solução f, sendo essa a melhor solução na vizinhança $N(f)$. Em uma instância (F, c) de um problema de otimização, com vizinhança N, a solução viável $f \in F$ é definida como sendo ótimo local em relação a N se:

$$c(f) \leq c(g) \forall g \in N(f) \tag{2.4}$$

Na figura 2.1, observando os pontos ótimos assinalados, pode-se ter a clara distinção entre o ótimo local e o ótimo global: os pontos A, B e C são ótimos locais, mas apenas o ponto B é o ótimo global.

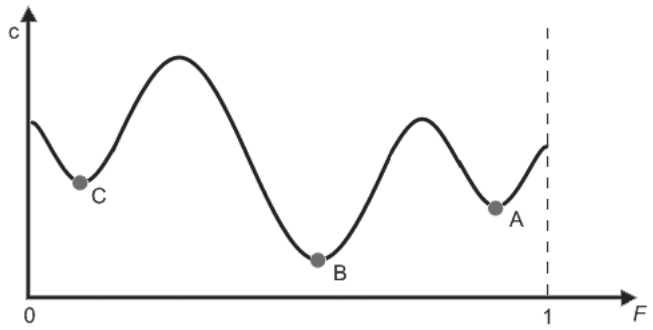

Figura 2.1: Problema Euclideano Unidimensional

No exemplo, utilizado por [Papadimitriou et.Al. (1982)], o problema abordado é conhecido por *1-dimensional Euclidean Problem*, ou simplesmente Problema Euclideano Unidimensional. Considerando a instância (F, c) onde $F = [0, 1] \subseteq R^1$ e a função de custos c traçada na figura 2.1, a vizinhança pode ser definida por:

$$N_\epsilon(f) = \{x : x \in Fe|x - f| \leqslant \epsilon\} \tag{2.5}$$

2.2 Problema de Roteamento de Veículos (*PRV*)

Como descrito por Andrade em [Andrade et.Al. (2004)], o mais conhecido e estudado problema de veículos segundo Larsen [Larsen (2000)] é o problema do caixeiro viajante (*PCV*), mais conhecido na literatura como *Traveling Salesman Problem* (*TSP*), que consiste em um vendedor que visita um conjunto de cidades e retorna à cidade onde começou a viagem. O objetivo consiste em minimizar a distância total percorrida. Este problema de programação matemática basicamente consiste na busca de um ciclo hamiltoniano em um grafo, sendo considerado um problema $\mathcal{NP}$-difícil segundo Melo [Melo (2001)], ou seja, não existe algoritmo em tempo polinomial determinístico que resolva o problema; apenas algoritmos exponenciais.

O problema de roteamento de veículos (*PRV*), mais conhecido na literatura como *Vehicle Routing Problem* (*VRP*) é uma generalização do problema do caixeiro viajante, consistindo em determinar rotas de veículos, onde uma rota é um ciclo que começa em um depósito, visitando um subconjunto de clientes ou pontos de demanda em uma dada ordem e retornando ao depósito. Todos os pontos de demanda devem ser visitados apenas uma vez e o total da demanda dos clientes de uma rota não pode exceder a capacidade do veículo, segundo Larsen [Larsen (2000)].

No mundo real o número de restrições aferidas ao problema é um fator complicante para a modelagem matemática. É perfeitamente aceitável que existam

múltiplos pontos de fornecimento, os depósitos ou facilidades, veículos com diversas capacidades, restrições de entrega como tempo ou distância. Todos estes fatores devem ser considerados como essenciais para a correta e fiel modelagem do tipo de problema tratado [Andrade et.Al. (2004)].

2.3 Problema de Roteamento de Veículos com Janela de Tempo (*PRVJT*)

O Problema de Roteamento de Veículos com Janela de Tempo (*PRVJT*), é uma generalização do bem conhecido Problema de Roteamento de Veículos (*PRV*) descrito na seção 2.2, introduzido por Dantzig [Dantzig et.Al. (1959)]. No *PRVJT* a frota de veículos deve visitar, efetuar um serviço, em um determinado número de consumidores. Todos os veículos iniciam e terminam em um único depósito. Para cada par de consumidores ou consumidor e depósito, existe um custo associado. Este custo denota quanto é dispendioso um o veículo a dirigir-se de um consumidor a outro. Cada consumidor deve ser visitado exatamente uma vez. Adicionalmente, cada consumidor demanda uma determinada quantidade de bens empregados (denotado como peso da carga). Para todos veículos da frota, têm-se um limite superior de carga suportado pelo mesmo (capacidade de carga). No caso mais básico, todos os veículos são do mesmo tipo e contam com a mesma capacidade. Então basicamente, o objetivo do *PRV* é encontrar o conjunto de consumidores atendidos por cada veículo da frota de maneira a diminuir os custos com o transporte, como descrito em [Larsen (1999)].

No *PRVJT* cada consumidor tem uma janela de tempo associada. Seu objetivo é descrever o tempo de início e término de atendimento, sendo que o veículo deve iniciar o serviço neste intervalo. Caso ele chegue antes do tempo descrito pelo início da janela de tempo, o mesmo deve esperar para que inicie o serviço. Um tempo é gasto pelo veículo junto ao consumidor, visto que este presta serviços (tempo de serviço).

Tanto no *PRV* como no *PRVJT* encontram-se vários tipos de objetivos de minimização na literatura. Pode-se a priori considerar somente a distância total percor-

rida (como neste trabalho). Outra forma de encontrar o menor custo é determinar o conjunto mínimo de rotas possíveis, e somente depois minimizar a distância total percorrida. Alguns autores também consideram o tempo de espera do veículo em um consumidor como um dos objetivos de minimização do sistema.

2.3.1 Modelo Matemático para o *PRVJT*

Como descrito por Larsen [Larsen (1999)], o Problema de Roteamento de Veículo com Janela de Tempo pode ser formulado da seguinte maneira: Um conjunto de V veículos idênticos, representado pelo conjunto $V = 1, .., M$, necessitam realizar entregas em uma região. Os N consumidores dentro desta região estão representados pelo conjunto C, que são vértices de um grafo $G = (C, A)$. Adicionalmente, incluem-se dois outros vértices, o vértice 0 representa o depósito central de onde partirão todos os veículos. Para facilitar a representação matemática do problema , se duplica o vértice do depósito central, ou seja, o vértice $N + 1$ também representará o depósito central, significando no problema o vértice de chegada de todos os veículos. As variáveis t_{ij} e c_{ij} representam respectivamente o tempo e a distância necessários para ir do vértice i para o vértice j. Cada consumidor i também está associado a uma demanda, ou seja, uma quantidade de encomenda q_i. Além disso, cada consumidor deverá ser atendido por um único veículo, não sendo permitido a divisão de uma encomenda por dois ou mais veículos. Quanto à janela de tempo, definida como o intervalo $[a_i, b_i]$, indica que a partir do instante inicial ai é permitido o início da entrega ou coleta no consumidor i. Caso a chegada do veículo no consumidor i se dê antes do instante a_i o veículo deverá esperar. O veículo nunca poderá chegar depois do instante b_i, pois viola a restrição de tempo do problema. Este tipo de restrição de tempo é conhecido na literatura como janela de tempo rígida ou *hard time window*. Os veículos são idênticos e possuem uma capacidade máxima de carga Q. A variável de decisão x_{ijv} determina se o veículo v faz o percurso do consumidor i para o consumidor j, recebendo o valor 1, se verdadeiro, e 0 em caso contrário.

Pode-se agora definir matematicamente o problema de otimização como:

$$\textbf{Minimize} \sum_{v \in V} \sum_{i \in G} \sum_{j \in G} c_{ij}\, x_{ijv} \textbf{ sujeito a:} \tag{2.6}$$

$$\sum_{v \in V} \sum_{j \in G} x_{ijv} = 1;\ \forall i \in C \tag{2.7}$$

$$\sum_{i \in C} q_i \sum_{j \in G} x_{ijv} \leq Q;\ \forall v \in V \tag{2.8}$$

$$\sum_{j \in G} x_{0jv} = 1;\ \forall v \in V \tag{2.9}$$

$$\sum_{j \in G} x_{i(N+1)v} = 1;\ \forall v \in V \tag{2.10}$$

$$\sum_{i \in G} x_{ijv} - \sum_{j \in G} x_{hjv} = 0;\ \forall h \in C, \forall v \in V \tag{2.11}$$

$$s_{iv} + ts_i + t_{ij} - K(1 - x_{ijv}) \leq s_{jv}; \forall i, j \in G, \forall v \in V \tag{2.12}$$

$$a_i \leq s_{iv} \leq b_i;\ \forall i \in G, \forall v \in V \tag{2.13}$$

$$x_{ijv} \in \{0,1\}; \forall i,j \in G, \forall v \in V \tag{2.14}$$

A fórmula 2.7 garante que somente um veículo v chega a cada consumidor i. Cada veículo v atenderá somente um conjunto de consumidores cuja demanda total não ultrapasse a sua capacidade Q (fórmula 2.8). As fórmulas 2.9 e 2.10 garantem que cada veículo v parte e retorna ao depósito central, respectivamente. A fórmula descrita por 2.11 indica a continuidade das rotas, ou seja, se um veículo chega a um consumidor ele deverá sair do mesmo para o consumidor seguinte, perfazendo todos os trechos entre consumidores. A restrição de tempo (com relação ao início da janela de tempo) é garantida pela fórmula 2.12, onde o instante de chegada de um veículo v a um consumidor $j(s_{jv})$ não poderá ocorrer antes do tempo de chegada no consumidor anterior $i(s_{iv})$ mais o tempo de serviço no primeiro (ts_i), mais o tempo de percurso no trecho (i,j) que é t_{ij}. É assumido uma velocidade constante tal que o tempo de percurso t_{ij} é igual a distância entre i e j. A constante K sendo suficientemente grande garante que a equação seja somente uma restrição efetiva quando x_{ijv} seja igual a 1, ou seja, quando o veículo v percorra a trecho (i,j). O respeito ao fim da janela de tempo do consumidor é garantido pelo fórmula 2.13, onde o instante de chegada de um veículo v em um consumidor i está dentro do limite da janela do mesmo. A fórmula 2.14 garante a integralidade das variáveis do problema.

Observa-se no modelo apresentado, um custo associado a cada arco do grafo, geralmente a distância do consumidor i ao consumidor j, sendo neste caso o objetivo a minimização da distância total percorrida.

2.3.2 Complexidade do *PRVJT*

De acordo com Cormem em [Cormen et.Al. (1999)], problemas que possuem algoritmos polinomiais para resolvê-los são considerados “fáceis", enquanto problemas que somente possuem algoritmos exponenciais para resolvê-los são considerados “difíceis".

Algoritmos polinomiais possuem a função de complexidade $O(p(n))$, onde $p(n)$ é um polinômio. Um exemplo para esta classe de problema é a pesquisa binária, cujo custo pode ser expressado por $O(\log n)$. Estes problemas são ditos da classe $\mathcal{P}$ [Cook (1971)].

A segunda classe de problemas, descrita pelos algoritmos exponenciais no tempo de execução têm sua função de complexidade descrita da forma $O(c^n)$, onde $c > 1$. Este grupo contém problemas cujos melhores algoritmos conhecidos são não-polinomiais. Algoritmos com complexidade não polinomial demandam tal quantidade de tempo para executar que mesmo problemas de médio porte não poderão ser resolvidos computacionalmente. Estes problemas são ditos da classe $\mathcal{NP}$ (*Non-deterministic Polynomial*) [Cook (1971)].

Como citado por Alvarenga em [Alvarenga (2004)], Cook prova a existência de uma sub-classe de problemas em $\mathcal{NP}$ correlacionados entre si, onde existe uma grande chance de serem realmente intratáveis, ou seja, de não existir um algoritmo polinomial determinístico para resolvê-los. Os problemas dessa sub-classe são chamados $\mathcal{NP}$-Completos. O primeiro problema $\mathcal{NP}$-Completo, provado por Cook, é o Problema da Satisfabilidade (conhecido como *SAT*, que consiste em dizer se existe ou não uma combinação de estados (sim/não) para variáveis booleanas de uma expressão, que a tornem verdadeira. Cook provou que se houver um algoritmo determinístico capaz de resolver a *SAT* em tempo polinomial então $\mathcal{P} = \mathcal{NP}$. Através de transformações em tempo polinomial, a *SAT* já foi reduzida a uma série de outros problemas, como o Problema do Caixeiro Viajante, o Problema de Coloração de Grafos, o Problema da Clique, e muitos outros. Conseqüentemente, se algum destes problemas forem resolvidos por algoritmos polinomiais a *SAT* também estará resolvida indiretamente, fazendo $\mathcal{P} = \mathcal{NP}$. Por isso, acredita-se que dificilmente tal algoritmo exista.

Encontrar solução para o *PRVJT* implica em obter simultaneamente a solução de vários problemas $\mathcal{NP}$-Difíceis, tais como o Problema do Caixeiro Viajante (*PCV*) e o Problema da Mochila (*Knapsack Problem*), sendo conseqüentemente considerado $\mathcal{NP}$-Difícil, citado por Alvarenga em [Alvarenga (2004)] e provado por Johnson em [Johnson et.Al. (1979)].

Os problemas de roteamento de veículos das classes *PRV* ou *PRVJT*, são geralmente problemas “difíceis" de serem resolvidos, pois o número de clientes, veículos e restrições podem variar bastante em cada problema, tornando uma solução exata inviável na maioria dos casos [Passos et.Al. (2003)]. Com apenas um veículo, o problema de roteamento de veículos com janela de tempo é $\mathcal{NP}$-Completo [Qily (1999)], idêntico ao Problema do Caixeiro Viajante (*PCV*). No caso mais comum, o número de veículos utilizados é maior do que um, tornando o problema $\mathcal{NP}$-Difícil [Johnson et.Al. (1979)]. Isso implica que soluções ótimas para a maioria das instâncias do *PRVJT* apenas podem ser alcançadas por algoritmos exatos em um tempo exponencial.

Conseqüentemente, apenas soluções em instâncias de ordem reduzidas podem ser encontradas utilizando algoritmos exatos como descritos na seção 2.4. Para contornar este problema, heurísticas são utilizadas para encontrar soluções aceitáveis para o *PRVJT*.

Segundo Alvarenga [Alvarenga (2004)], a utilização de heurísticas devem apresentar duas características básicas para garantir bons resultados:

1. Qualidade da solução: capaz de encontrar soluções próximas do ótimo, em tempo bem inferior ao necessário pelos métodos exatos disponíveis; o que justificaria sua utilização;

2. Robustez: a qualidade da solução não deve variar demasiadamente com diferentes instâncias de um mesmo problema para o qual a heurística foi projetada, ou mesmo, variar demais quando a heurística é aplicada várias vezes para a mesma instância.

Atendendo a estas características de boas soluções, tem-se os métodos de aproximação, que garantem soluções a uma determinada distância do ótimo (seção 2.5).

2.4 Métodos exatos para solucionar o *PRVJT*

Os métodos de busca por soluções denomidados exatos são aqueles que sempre encontram a melhor solução para o problema (se é que esta existe), ou seja, a solução ótima. Esta solução deve satisfazer de forma ótima a função objetivo correspondente ao problema em questão, respeitando todos os parâmetros que se aplicam à resolução do mesmo [Andrade et.Al. (2004)].

Segundo Cormen [Cormen et.Al. (1999)], um problema pode ou não possuir um algoritmo exato para sua solução. Existindo esse algoritmo, o mesmo pode não encontrar um ótimo em tempo hábil, ou seja, o algoritmo pode levar décadas para encontrar a solução desejada. Quando isso ocorre, diz-se que o mesmo é inviável para a instância abordada [Andrade et.Al. (2004)].

Com isto, mais uma questão deve ser analisada na escolha da forma exata de resolução dos problemas: a viabilidade do método escolhido. Como pode-se observar, nem sempre um método exato consiste na melhor forma de abordagem para um problema. Aprofundando ainda mais a questão de viabilidade, tanto algoritmos da classe $\mathcal{P}$ quanto algoritmos da classe $\mathcal{NP}$ podem ser inviáveis: tomando por exemplo um algoritmo exato com teto $O(n^{100})$, mesmo pertencendo à classe $\mathcal{P}$ este algoritmo pode não viável computacionalmente para, por exemplo, $n = 10000$ entradas [Andrade et.Al. (2004)].

Especificamente, os problemas classificados como polinomiais (pertencentes à classe de problemas $\mathcal{P}$) são em sua maioria viáveis para aplicação de métodos exatos para resolução [Andrade et.Al. (2004)].

2.4.1 Programação dinâmica

Como citado por Passos em [Passos et.Al. (2003)], a programação dinâmica baseia-se na técnica de *branch-and-bound* para alcançar o ótimo. Um limite inferior é verificado para cada nó incluído na solução. Quando algum novo nó propicia um limite inferior ao atual, este limite inferior torna-se o novo limite inferior. Cada nó referente ao cliente é visitado somente uma vez. Este algoritmo é implementado em [Kaan et.Al. (1987)] e resolvido para até 15 consumidores.

2.4.2 Relaxação de Lagrange

O modelo matemático do *PRVJT* [Larsen (1999)], com suas restrições, com por exemplo garantir que todo cliente só é visitado uma única vez, sofre um relaxamento através do método de Lagrange. A partir desse relaxamento um limite inferior global é calculado. A partir desse limite inferior a técnica do *branch-and-bound*, é aplicada. A cada novo cliente visitado verifica-se se o limite inferior global, não está sendo ultrapassado pela rota estabelecida, caso esse novo cliente seja incluído na solução [Passos et.Al. (2003)].

2.5 Métodos aproximados para solucionar o *PRVJT*

Cormen ([Cormen et.Al. (1999)]) denota os métodos aproximativos como uma possibilidade viável de se encontrar soluções para problemas exponenciais onde os métodos exatos não se aplicam computacionalmente. Os algoritmos aproximativos, por possuírem tempo polinomial, podem resolver problemas exponenciais com uma boa margem de aproximação à solução ótima [Andrade et.Al. (2004)].

Como descrito por Andrade em [Andrade et.Al. (2004)], os métodos aproximativos possuem a característica de buscar soluções com desempenho consideravelmente bom, em tempo hábil e viáveis para a maioria dos problemas onde são utilizados. Segundo Cormen ([Cormen et.Al. (1999)]), os algoritmos aproximativos sempre se aproximam da solução ótima, com qualidade de solução dependente do tempo de execução. A relação de aproximação de um algoritmo é $a(n)$ se, para qualquer entrada de tamanho n, o custo C da solução produzida pelo algoritmo de aproximação está dentro de um fator $a(n)$ do custo C^* de uma solução ótima:

$$max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq a(n) \tag{2.15}$$

2.5.1 Construção de rotas

O algoritmo inicia-se com todas as rotas simples possíveis com apenas um consumidor. E a cada interação, calcula-se quais duas rotas podem ser combinadas com a maior economia de recursos [Backer et.Al. (1986)].

2.5.2 *Time-oriented-nearest-neigbour heuristc*

Proposta por Solomon em [Solomon (1986)], toda rota é inicializada por um consumidor ainda não roteado mais próximo do depósito central. Esta relação de proximidade é temporal quanto geográfica. A cada interação o consumidor mais próximo, geográfica e temporalmente, ao último cliente adicionado é considerado para inserção na rota em questão. Quando a pesquisa por um novo consumidor é feita, e um consumidor não é encontrado, uma nova rota é iniciada.

2.6 Métodos heurísticos para solucionar o *PRVJT*

Como descrito por Andrade em [Andrade et.Al. (2004)], os métodos heurísticos compõem uma gama relativamente nova de soluções para problemas de otimização combinatória. Tais métodos possuem origens distintas, geralmente utilizadas especificamente para determinado problema combinatório. Quando um método é aplicado especificamente no problema, sob a forma de um algoritmo, este é denominado um método heurístico, ou simplesmente uma heurística. Existe também na literatura o termo meta-heurística. A distinção entre tais denominações faz-se por:

- Meta-heurística: possui grande abrangência, podendo ser aplicada à maioria dos problemas de otimização combinatória. Pode-se citar como exemplo as Meta-heurísticas *ACO* (*Ant Colony Optimization*), *GA* (*Genetic Algorithm*), *SA* (*Simulated Annealing*) e *TS* (*Tabu Search*).

- Heurística: é a instanciação de uma meta-heurística, ou seja, a aplicação da mesma em um problema específico de otimização. Por exemplo, em

[Dorigo et.Al. (1991)], denominou-se *AntSystem* a heurística *ACO* aplicada à resolução do Problema do Caixeiro Viajante. Na realidade, a heurística *AntSystem* foi primeiramente aplicada ao *TSP*, para depois ganhar dimensões de meta-heurística, quando Dorigo [Dorigo et.Al. (1999)] a flexibilizou para tal.

A princípio, os métodos aproximativos e os métodos heurísticos podem parecer iguais, o que não é fato. Ambos os métodos de resolução buscam, de maneira viável, trazerem soluções próximas àquela ótima, mas as semelhanças param por aqui [Andrade et.Al. (2004)].

Um método aproximativo garante que a solução se aproxima do ótimo a cada iteração, e realmente o faz, além de ser denotado matematicamente como eficaz. Já os métodos heurísticos não garantem qualquer tipo de melhora de solução conforme o número de iterações, além de não ter garantia de convergência, como descrito por Andrade em [Andrade et.Al. (2004)].

Em primeira instância, os métodos aproximativos parecem ser mais eficientes que os heurísticos, o que também não ocorre. As heurísticas, embora tenham os problemas explicitados acima, costumam, na prática, convergir em tempo extremamente rápido quando comparadas aos métodos aproximativos, fato peculiar que fez com que esse método tivesse uma grande disseminação nos últimos anos [Andrade et.Al. (2004)].

2.6.1 *Push-Forward Insertion Heuristic (PFIH)*

Como citado por Larsen em [Larsen (1999)], o *PFIH* possui uma estratégia construtiva eficiente para calcular o custo de um novo consumidor em uma rota. Este custo é calculado de acordo com sua posição geográfica, com o fim de sua janela de tempo e o ângulo existente entre ele e o depósito central.

O *PFIH* é descrito da seguinte maneira: Considere uma rota qualquer R_q com m consumidores, $R_q = (C_1, ..., C_m)$, onde $C1$ é o primeiro consumidor atendido e Cm é o último. O depósito central é considerado como C_0 e também C_{m+1}. Considere também, $[a_1, b_1]$ e $[a_m, b_m]$ os intervalos que definem a janela de tempo

do primeiro e último consumidor, respectivamente. Lembrando que o intervalo corresponde aos horários inicial e final dentre os quais o veículo deve chegar ao consumidor. A chegada pode acontecer antes, porém será necessário esperar para o início do serviço. A viabilidade de inserir um novo consumidor na rota R_q é testada inserindo-o entre cada par de consumidores, sendo a posição de menor aumento na distância percorrida na rota R_q viável selecionada. Se o consumidor C_j é inserido entre o consumidor C_0 e C_1, o tempo de chegada t_1 ao consumidor C_1, irá sofrer um deslocamento para frente, podendo causar o mesmo para todos os consumidores seguintes (*Push Forward*). Como o veículo pode estar aguardando em algum consumidor da rota R_q, é possível que o valor do deslocamento neste consumidor seja zero. A partir deste consumidor, o tempo de chegada do veículo nos consumidores seguintes permanece inalterado.

A heurística *PFIH* inicia uma rota com um único consumidor i, escolhido conforme seu custo de inicialização, dado pela equação 2.16.

$$custo_i = \left[-\alpha d_{0i} + \beta b_i + \gamma \frac{p_i}{360} d_{0i}\right] \tag{2.16}$$

Onde:

$\alpha = 0.7$; $\beta = 0.1$; $\gamma = 0.2$;

$d_{0i} =$ distância do depósito central ao consumidor i;

$b_i =$ limite superior da janela de tempo de chegada ao consumidor i;

$p_i =$ ângulo da coordenada polar do consumidor i, referente ao depósito central;

Os valores para os parâmetros α, β e γ foram definidos empiricamente por Solomon em [Solomon (1987)].

A partir do primeiro consumidor escolhido, conforme a equação 2.16, os demais são testados um a um, entre cada posição possível na rota em construção. A posição e o consumidor que resultarem no menor acréscimo da distância total

percorrida, sem violação de capacidade e janela de tempo é escolhido. Após não mais haver consumidores possíveis para inserção na rota em construção, esta é encerrada e inicia-se novamente o mesmo processo com uma nova rota vazia, sendo o primeiro consumidor aquele de menor custo de acordo com o cálculo da fórmula 2.16, dentre os que ainda não foram roteados.

2.6.2 Heurística *r-Opt*

Uma das heurísticas mais utilizadas em problemas de roteamento de veículos é a r-Opt, onde r arcos são removidos e trocados por outros r arcos. O valor atribuídos a r mais encontrado na literatura é 3, mas 2 é outro valor bastante referenciado na literatura, como citado por Passos em [Passos et.Al. (2003)].

2.6.3 Heurística *shift-sequence*

Proposta por Schulze e Fahle em [Schulze et.Al. (1999)]. Nela um cliente é movido de uma rota para outra, então, checando todas posições possíveis de inserção.

Se uma inserção é inviável (por violar restrições de carga ou tempo), o consumidor é então inserido em outra rota. Este procedimento se repete até que a viabilidade das rotas alteradas seja atingida.

2.6.4 *Simulated anneling (SA)*

O recozimento simulado (como pode ser traduzido) é uma técnica estocástica de relaxação que tem sua origem na mecânica estatística [Metropolis et.Al. (1953)]. Como descrito por San em [San et.Al. (2001)], a metodologia do *Simulated anneling* extrai sua analogia do processo do recozimento de sólidos. No processo do recozimento, um sólido é aquecido a uma alta temperatura e refrigerado gradualmente para que ele se cristalize. Neste processo, o aquecimento permite que os átomos se movam de maneira aleatória. Se o processo de refrigeração for feito rapidamente os átomos não alcançam um equilíbrio térmico. Se o sólido for resfriado lentamente, dá aos átomos bastante tempo para se alinharem a fim alcançar

um estado mínimo da energia. Isso é uma analogia sobre a definição da otimização combinatória, onde o ato de encontrar o melhor posicionamento pode ser assimilado a minimização de uma função objetivo.

Como mencionado por San em [San et.Al. (2001)], o interesse de se resolver problemas de otimização combinatória utilizando o comportamento observado na natureza de recozimento simulado começou na década de 80 com Kirkpatrick [Kirkpatrick et.Al. (1983)]. O recozimento simulado usa a idéia de uma aproximação estocástica para dirigir a busca. Permite, a partir de uma solução S, encontrar uma solução em sua vizinhança $V(S)$, mesmo que esta nova solução S' signifique uma piora no valor da função objetivo.

O *Simulated anneling* originalmente se comporta da seguinte maneira: Se um movimento em S, produzindo S', melhorar o valor da função objetivo, então o movimento é sempre aceito. Mais precisamente, a nova solução S' é aceita se $\Delta \leq 0$, onde $\Delta = C(S') - C(S)$. Para permitir que a busca escape de mínimos locais, movimentos que aumentem o valor da função objetivo, ou seja, $\Delta > 0$, somente são aceitos dentro da probabilidade $e^{(-\Delta/T)}$, onde T é a variável chamada temperatura. O valor de T varia de um valor relativamente grande a um valor pequeno, tendendo a 0. Este valor é controlado com a refrigeração do sistema, que especifica o quão lento T irá diminuir.

O *Simulated anneling* implementado por San em [San et.Al. (2001)] utiliza a heurística *PFIH* (descrita na seção 2.6.1) como ponto de partida antes de aplicar a técnica do recozimento simulado.

Heurísticas baseadas na busca local são empregadas nas iterações do *SA* e a nova solução S' será aceita ou rejeitada de acordo com o novo custo encontrado $C(S')$ e com a temperatura atual do sistema T. Movimentos bruscos no espaço de busca são menos freqüentes com a temperatura T próxima de 0.

A notação seguinte auxilia no entendimento do algoritmo do *Simulated anneling*.

$T_i =$ Temperatura inicial do sistema.

$T_f =$ Temperatura final do sistema.

$T_m =$ Temperatura onde a melhor solução foi encontrada.

$S =$ Solução corrente.

$V(S) =$ Função que encontra um elemento na vizinhança de S.

$S' =$ Vizinho de S.

$C(S) =$ Função objetivo.

$Sm =$ Melhor solução encontrada.

$\tau =$ Decremento da temperatura constante no limite de $0 < \tau < 1$.

Algoritmo 1 Simulated Anneling

Pré-condição: $0 < \tau < 1$ e $T_f \geq T_i$

$S \leftarrow S_m \leftarrow$ PushForwardInsertionHeuristic; {solução inicial}
$T \leftarrow Ti$; {inicializando a temperatura do sistema}
enquanto $T \neq T_f$ **faça**
  $S' \leftarrow V(S)$; {encontrando um vizinho (busca local)}
  $\Delta \leftarrow C(S') - C(S)$;
  se $\Delta \leq 0$ **então**
    $S \leftarrow S'$; {atualizando a solução encontrada}
  senão
    $\theta \leftarrow NumeroAleatorio$; {sorteando um valor}
    se $e^{(-\Delta/T)} \geq \theta$ **então**
      $S \leftarrow S'$; {admite piora na solução}
    fim se
  fim se
  se $C(S) < C(S_m)$ **então**
    $S_m \leftarrow S$; {atualizando melhor solução encontrada}
  fim se
  $T \leftarrow \left\lceil \frac{T}{1+\tau T} \right\rceil$; {atualizando a temperatura do sistema}
fim enquanto

2.6.5 Busca TABU (*BT*)

De acordo com Arakari em [Arakari (1998)] a busca tabu foi introduzida por Glover em [Glover (1986)]. Uma visão detalhada foi feita por Glover nos anos de 1989 e 1990 através de um tutorial.

A busca tabu (*BT*) parte de uma solução inicial e através de uma seqüência de movimentos caminha para outra solução. O conjunto de movimentos é chamado de lista de candidatos e o método utilizado para comparar os movimentos é conhecido por avaliador [Arakari (1998)].

Em muitos problemas de otimização combinatória há uma infinidade de ótimos locais que são inferiores ao ótimo global. Portanto, uma heurística que realiza movimentos até que um ótimo local seja alcançado, nem sempre alcançará um ótimo global. Consequentemente, seria interessante desenvolver uma heurística que uma vez encontrado um ótimo local continue a busca por ótimos locais melhores e talvez encontrando até o ótimo global. A *BT* permite isto impondo restrições na lista de candidatos [Arakari (1998)].

Agora, pode ocorrer que na tentativa de buscar outras soluções o movimento realizado conduza diretamente de volta ao ótimo local já visitado. Para prevenir esta volta, um número de atributos relacionados ao movimento são registrados numa lista tabu. Os movimentos que revertem os movimentos já aceitos são considerados tabu, eles não serão aceitos a menos que satisfaçam algum critério de aspiração. A *BT* escolhe o melhor movimento admissível, cujo movimento não seja tabu ou satisfaça o critério de aspiração [Arakari (1998)].

Possuindo os atributos do movimento consegue-se verificar se este é tabu ou não varrendo a lista tabu. Os atributos tabu são armazenados na lista tabu por um número pequeno de iterações e então são removidos. Atributos armazenados na lista intensificam a busca pelo travamento de um número de feições, restringindo o espaço de busca. Porém, a diversificação é conseguida quando se considera movimentos que não envolvem os atributos tabu de forma a direcionar a busca para novas e confiantes áreas do espaço de solução. Este processo de travamento de atributos e depois estrategicamente esquecê-los é chamado de componente de memória de termo curto da busca tabu [Arakari (1998)].

Os critérios de aspiração são utilizados de modo que movimentos interessantes que são tabu não sejam rejeitados completamente.

Uma das primeiras tentativas de aplicar a *BT* ao *PRV* é devido a Willard [Willard (1989)]. O *PRV* é transformado, através da replicação dos depósitos,

em um *PCV* e as vizinhanças são definidas como todas as soluções viáveis que possam ser atingidas através da solução corrente utilizando permutação 2-opt ou 3-opt, no entanto os resultados foram muito ruins, sendo superados por heurísticas tais como varredura [Gillet et.Al. (1974)] e o algoritmo de duas fases de Christofides em [Chistifides et.Al. (1979)].

Pureza e França apresentam um implementação da *BT* aplicado diretamente ao *PRV* [Pureza et.Al. (1991)]. A aplicação é muito básica e não há critério de aspiração, onde os movimentos considerados são baseados em movimentos de permutação de nós de Dror e Levy [Dror et.Al. (1986)]. Duas listas tabus foram utilizadas, uma para bordas deletadas e outra para bordas acrescentadas. Os resultados foram melhores do que a abordagem anterior, porém é inferior a implementações de *BT* posteriores.

Osman [Osman (1993)] fez uma implementação cujo os movimentos são similares aos propostos por Pureza e França. A abordagem de Osman utiliza o mecanismo de permutação l, com $l = 2$, isto inclui uma combinação de movimentos 2-opt, reatribuição de vértices a diferentes rotas e permutação de vértices entre duas rotas. O mecanismo 1-permutação foi utilizado para gerar a lista de candidatos e uma estrutura especial de dados foi criada para reduzir o tempo de avaliação da lista de candidatos. A implementação de Osman produziu resultados levemente melhores para problemas com restrições de tempo e capacidade.

Outra aplicação de *BT* é de Taillard [Taillard (1993)], em alguns aspectos é similar ao de Osman (vizinhança 1-permutação, restrições tabu). Utiliza um esquema de *BT* robusta com uma lista tabu de comprimento variando de $0,4n$ a $0,6n$. Para problemas Euclideanos, onde os clientes estão uniformemente distribuídos em volta do depósito central, Taillard sugere que os problemas sejam particionados em setores. Cada subproblema é então tratado como um independente *PRV* e resolvido utilizando *BT* para um pequeno número de iterações. Este método de particionamento produziu resultados expressivos, encontrando para os testes de Taillard novas melhores soluções ou as melhores soluções conhecidas.

Semet e Taillard [Semet et.Al. (1993)] utilizaram a *BT* para resolver um *PRV* real, o problema envolve uma cadeia de supermercados na Suíça, possui janela de

tempo para cada pedido de entrega, além de um frota heterogênea (21 caminhões e 7 *trailers*), outras restrições são impostas: muitos supermercados não podem ser acessados pelo trailer e outros somente podem aceitar alguns caminhões. Semet e Taillard descrevem uma técnica para atribuir otimamente veículos às rotas, por ser excessivamente dispendioso computacionalmente ele é somente executado no fim de uma iteração, quando se quer atualizar a solução e não para avaliar cada tentativa de movimento. Utiliza técnicas de redução de tempo de *CPU*, como por exemplo: reduzir o tamanho da vizinhança. Outra forma é agregar os pedidos, múltiplos pedidos de uma dada loja podem ser agregados para um único pedido, quando o volume total é menor do que a capacidade do menor caminhão que pode alcançar a loja. Esta agregação pode dificultar a descoberta de uma solução ótima para o problema, mas reduz, além do tempo de *CPU*, o espaço de soluções. Duas listas tabus foram estudadas. A primeira proíbe um pedido de retornar a alguma rota particular e a segunda proíbe lojas de receber entregas por alguma rota particular. Os resultados indicaram que procedimento da *BT* obteve melhores soluções do que aquelas geradas pelo método anteriormente utilizado pela companhia. Houve redução de custos de 10 a 15 porcento.

TABUROUTE é uma algoritmo desenvolvido na década de 90 por Gendreau [Gendreau et.Al. (1994)], sua implementação possui características próprias. O movimento básico é feito através de permutação de vértices, utilizando *GENI* [procedimento de inserção generalizada feito por Gendreau [Gendreau et.Al. (1992)]. Este procedimento consiste em inserir um vértice entre dois dos seus p vizinhos mais próximos na rota enquanto realiza um reotimização local da rota.

Outra característica é a busca estar oscilando entre soluções viáveis e inviáveis. Isto é útil para problemas altamente restritos onde é difícil mover diretamente de uma solução viável para outra. *TABUROUTE* não utiliza lista tabu mas etiquetas (*tags*) tabu aleatórias. Sempre que um vértice for removido de uma rota r para outra rota s numa iteração t, o retorno deste vértice a rota r é proibida até a iteração $t + q$, onde q é um inteiro aleatório de intervalo 5 a 10 [Arakari (1998)].

TABUROUTE possibilita a utilização de uma estratégia de diversificação por intermédio da penalização de vértices que se movem freqüentemente com a fina-

lidade de aumentar a probabilidade dos vértices de movimento lento. Processo geral do algoritmo começa com uma inicialização (geração de soluções iniciais) passando para um melhoramento da solução(executa a *BT*) e uma fase de intensificação da solução [Arakari (1998)].

Os resultados de *TABUROUTE* são expressivos. O método produz resultados superiores a outras heurísticas conhecidas [Willard (1989), Pureza et.Al. (1991), Osman (1993)] para 10 dos 14 problemas testados [Arakari (1998)].

2.6.6 Algorítmo Genético (*AG*)

Os algoritmos genéticos foram introduzidos por John Holland [Holland (1975)], com intuito de aplicar a teoria da evolução das espécies elaborada por Darwin [Darwin (1859)], utilizando os conceitos da evolução biológica como genes, cromossomos, cruzamento, mutação e seleção a teoria computacional, ou seja, formalizar matematicamente e explicar rigorosamente processos de adaptação em sistemas naturais e desenvolver sistemas artificiais (simulados em computador) que mantenham os mecanismos originais, encontrados em sistemas naturais. O processo de evolução executado por um algoritmo genético corresponde a um procedimento de busca no espaço de soluções potenciais para o problema e, como enfatiza [Michalewicz (1992)], esta busca requer um equilíbrio entre dois objetivos aparentemente conflitantes: a procura das melhores soluções na região que se apresenta promissora ou fase de intensificação e a procura de outra região, que espera-se ainda mais promissora, ou exploração do espaço de busca. Este equilíbrio é o que potencializa o método, na busca de um ótimo global, diferenciado, por um lado, dos métodos convencionais que predominantemente fazem intensificação pois usam uma informação local, com é a derivada, por outro lado, diferencia-se dos métodos de busca aleatória nos quais predominam a exploração do espaço de busca [Carlos (2002)].

Atualmente, os *AG* tem se constituído ferramentas poderosas para resolver problemas onde o espaço de busca é muito grande e os métodos convencionais se mostraram ineficientes [Srivinas et.Al. (1994)].

Diferentemente dos métodos de busca e otimização tradicionais, *AG* traba-

lham não com uma única solução, mas com uma população de soluções; em vez dos parâmetros dos problemas, da codificação desses parâmetros; utilizam ainda informações de custo ou recompensa e regras de transição probabilísticas [Arakari (1998)].

Embora *AG* sejam, atualmente, uma classe de procedimentos pois existem variações na sua forma, faremos uma explicação sucinta do funcionamento básico destes algoritmos. Inicialmente, temos a geração de uma população formada por indivíduos (criados aleatoriamente) que podem ser vistos como possíveis soluções do problema. Após o processo de seleção uma porcentagem dos mais adaptados permanecem e os outros são descartados, por isso cada indivíduo é avaliado por uma função de avaliação recebendo uma nota ou índice. Para obter uma boa diversidade de soluções é aplicado ao indivíduos selecionados operadores de cruzamento e mutação gerando a próxima descendência. Este processo chamado de reprodução é continuado até que algum critério seja alcançado (número de iterações ou solução satisfatória alcançada) [Arakari (1998)].

A população é composta por um conjunto de cadeias de bits ou caracteres (cromossomos). Cada cromossomo é avaliado por uma função de avaliação (fitness) e atribuído a ele (figura 2.2).

Figura 2.2: Estrutura de um cromossomo

A seleção é feita para que do conjunto inicial de indivíduos, após muitas gerações, os indivíduos mais aptos permaneçam. Um método de seleção muito utilizado é o método da roleta, onde indivíduos de uma geração são escolhidos para fazer parte da próxima geração, através do sorteio de roleta. Neste método, os indivíduos mais aptos possuem uma porcentagem maior na roleta em relação aos indivíduos menos aptos, gira-se a roleta um determinado numero de vezes, dependendo do tamanho da população, e são escolhidos os indivíduos para a próxima

geração [Arakari (1998)].

O cruzamento (*crossover*) é um operador responsável pela recombinação de características dos pais durante a reprodução e é aplicado com probalidade dada pela taxa de cruzamento. Este operador pode ser de:

- um ponto: um ponto de cruzamento é escolhido e a partir deste as informações genéticas dos pais são trocadas.
- Multi-pontos: é a generalização da idéia de troca de material genético através de pontos, onde muitos pontos de cruzamento podem ser utilizados.

O operador mutação altera arbitrariamente um ou mais genes de um cromossomo. Desta maneira, assegura-se que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca será zero, além de contornar o problema de mínimos locais, pois com este mecanismo, altera-se levemente a direção de busca [Arakari (1998)].

A dificuldade inicial para a implantação de *AG* ao *PRV* é a codificação de seus parâmetros, pois não é natural para este tipo de problema. Uma representação mais sofisticada deve ser utilizada para codificar as rotas. E esta codificação pode ser feita das mais variadas formas [Arakari (1998)].

Hjorring fez um estudo de três formas de cruzamento para o Problema de Roteamento de Veículos em [Hjorring (1995)]. A primeira utiliza cruzamentos padrões e uma codificação que aborda o *PRV* como um problema de particionamento (os clientes são particionados em diferentes rotas). A codificação feita pelo *AG* toma uma cadeia, $I = i_1 i_2 ... i_n$ de n genes. Cada gene é um número de $i...K$, onde o j-ésimo gene, ij, é o índice do veículo que serve o j-ésimo cliente. De forma que cada cliente, j, é atribuído ao ij-ésima rota. Este tipo de codificação faz com que soluções levemente diferentes possam ter codificações muito diferentes e ao se aplicar os operadores padrões, freqüentemente resultarão em soluções inviáveis. Para contornar o problema dos indivíduos inviáveis foi feita um reparo nos indivíduos pela criação de rotas “sobrecarregadas", de forma que os clientes que estavam em rotas inviáveis fossem removidos até a rota se tornar viável.

A segunda forma utiliza um cruzamento feito inicialmente para o problema do caixeiro viajante *PCV* e trabalha com a ordem dos clientes; uma outra heurística, método da pétalas, gera as soluções para o *PRV*. A abordagem para o *PCV* utiliza simplesmente a ordem dos clientes a serem visitados, como meio de codificação, ou seja a representação do caminho. Como operador de cruzamento é utilizado o Recombinação de Bordas (Edge Recombination) desenvolvido por Whitley e Starkweather em [Whitley et.Al. (1989)]. Este operador constrói uma tabela de bordas formado pelas bordas de ambos os pais. Se uma borda é comum a ambos os pais, ele entra uma vez e é marcado com um sinal negativo. Esta tabela possui os vértices e as ligações existentes entre os vértices, uma vez a tabela montada, o operador dá o início a geração escolhendo um vértice aleatoriamente e verificando se há ligações com sinal negativo ou vértices com menos ligações. O vértice escolhido é eliminado da tabela e processo se repete até formar um trajeto.

A terceira forma trabalha diretamente sobre as soluções do *PRV*. Este operador de cruzamento, chamado de rota semente, combina rotas dos pais de forma que um deles é o pai semente da onde provêem a rota semente. O filho é gerado da seguinte forma: se faz uma cópia do outro pai (que não é o pai semente) e se retira a rota semente. Neste subconjunto de clientes ordenados se aplica o método das pétalas, gera-se, então uma solução para o *PRV* incompleta, após isto se introduz de volta a rota semente completando o filho.

Hjorring [Hjorring (1995)] concluiu que os resultados apresentados tanto pela primeira forma (cruzamento padrão e *PRV* codificado como problema de particionamento) como pela segunda forma (recombinação de bordas) foram desanimadores. Em ambos, a porcentagem média acima do melhor conhecido da população inicial era 14%, utilizando o *AG* somente foi possível melhorar isto para 13% e $8,4\%$ para a segunda forma. A terceira abordagem executou melhorou o desempenho do *AG*, apresentando um intervalo de apenas $1,36\%$.

Thangiah, Nygard e Juell [Thagiah et.Al. (1991)] proporam um sistema de *AG* para resolver o *PRVJT*. Seu sistema chamado de *GIDEON* consiste de dois módulos, um módulo de agrupamento global que atribui clientes aos veículos por um processo chamado setorização genética (*GENSECT*) e outro módulo que faz uma

otimização local das rotas (*SWITCH-OPT*).

Inicialmente, os clientes são divididos em setores ou agrupamentos utilizando um algoritmo de varredura que lembra o de Gillett e Miller [Gillet et.Al. (1974)]. A cada cliente é atribuído um pseudo ângulo em coordenadas polares, de maneira que a diferença angular entre dois clientes adjacentes seja igual. Os clientes são divididos em K agrupamentos K é o tamanho da frota de veículos baseados num conjunto de ângulos "sementes". Cada cliente é atribuído a um determinado agrupamento de acordo com seu ângulo.

O *AG* é utilizado para encontrar a melhor configuração de ângulos sementes. Durante o processo de setorização genética cada cromossomo representa um conjunto de ângulos sementes. Cada ângulo semente é representado por 3 bits e para um problema de K veículos, $K-1$ ângulos sementes são necessários (o primeiro ângulo semente é igual a 0 e portanto o comprimento de cada cromossomo será $[K-1] \times 3$. A população inicial é gerada aleatoriamente. Um conversão de bits para ângulo semente é realizada. A função de avaliação é o custo total das rotas dos agrupamentos formados a partir do conjunto de ângulos sementes derivados do cromossomo. Foram utilizados operadores de cruzamento tradicionais (2-pontos).

Portanto, os clientes são particionados de acordo com os ângulos sementes e as rotas são determinadas por uma heurística de inserção do mais barato (*cheapest insertion*).

O módulo de otimização de rota procura melhorar a solução pela movimentação e troca de clientes dentro e entre os agrupamentos. Quatro tipos operações são realizadas pelo módulo: mover 1 cliente, mover 2 clientes, trocar 1 cliente e trocar 2 clientes e em cada uma delas é avaliado o custo da rota. Estas operações de otimização local são realizadas até que não haja mais redução de custo. Esta nova solução pode ser introduzida novamente no primeiro módulo (GENSECT). Thangiah [Thagiah et.Al. (1991)] verificou que duas iterações são suficientes para obter boas soluções.

GIDEON apresentou um redução média de $3,9\%$ no tamanho da frota e $4,4\%$ na distância percorrida pelos veículos para 56 problemas. Blanton e Wainwritght [Blanton et.Al. (1993)] descrevem um outro esquema de *AG* para o *PRVJT*, eles

concluíram que os operadores de cruzamento tradicionais não eram eficientes para resolver problemas de otimização com múltiplas restrições e por isso apresentaram dois novos operadores de cruzamento: Combinação em Cruz#1[$MX1$] (*Merge Cross*) e Combinação em Cruz $\#2$ [$MX2$]. Eles se baseiam na noção de uma global precedência entre os genes independente de qualquer cromossomo. Isto é, dado um conjunto de N possíveis genes, deve existir uma relação de precedência de modo que há um desejo genérico de um gene i ocorrer antes do gene j dentro de um cromossomo. No *PRVJT*, cada gene é um cliente e tem associado uma janela de tempo de serviço, portanto há uma natural precedência entre os clientes por causa da sua janela de tempo correspondente. Esta precedência temporal é utilizada para escolher qual dos genes permanece no momento do cruzamento. Um exemplo é dado a seguir:

Vetor de precedência: 0 1 2 3 4 5 6 7 8 9
Cromossomo A: 2 5 6 10 7 3 8 4 9
Cromossomo B: 4 1 6 9 3 8 2 0 5 7

O primeiro gene de $A[2]$ é anterior ao correspondente gene de $B[4]$ como podemos notar no vetor de precedência, portanto ele permanece no filho AB e o primeiro gene de $B[4]$ e trocado para manter a viabilidade resultando em:

Cromossomo A: 2 5 6 1 0 7 3 8 4 9
Cromossomo B: 2 1 6 9 3 8 4 0 5 7
Filho AB: 2 x x x x x x x x x

Os segundos genes são comparados e neste caso o segundo gene de B permanece no filho AB e os genes 5 e 1 são permutados no cromossomo A. Continuando o cruzamento temos:

Filho AB: 2 1 6 5 0 7 3 4 8 9

Blanton e Wainwritght em [Blanton et.Al. (1993)] utilizam uma instância baseado em precedência temporal, porém, não há nenhuma razão que impeça que os operadores possam utilizar um outro tipo de precedência como distância, capacidade ou um outro critério de otimização. O exemplo utilizado se refere a um único veículo. Para múltiplos veículos, um método de codificação foi utilizado onde um cromossomo representa uma permutação de n clientes para ser divididos em k veículos. Uma heurística gulosa atribui os primeiros k clientes a k veículos.

O restante $n - k$ clientes são atribuídos individualmente. Uma inserção por tentativa de um cliente para alguma subrota é avaliada para cada veículo e a melhor subrota é selecionada (com menor custo).

Os operadores de cruzamento MX foram comparados com operadores de cruzamento padrões como recombinação de bordas (*Edge Recombination*), cruzamento cíclico (*Cycle Crossover*) e PMX para 4 problemas variando de 15 a 99 clientes. Em geral, os operadores MX se mostraram mais adaptados a encontrar soluções viáveis do que os operadores padrões.

Potvin e Bengio [Potivin et.Al. (1996)] criaram um sistema de roteamento genético para o *PRVJT*, que é aplicado diretamente as soluções do *PRVJT* evitando o problema de codificação. Este sistema conhecido como *GENEROUS*, realiza o cruzamento pela ligação do primeiro cliente da rota do pai 1 ao último cliente da rota do pai 2, como na heurística de troca 2-opt.

Um operador de reparos é aplicado para tornar a solução gerada viável. Primeiramente, vértices em duplicata são eliminados. Depois, cada vértice fora de rotas são acrescentados a nova solução utilizando o critério de inserção do mais barato. Se um vértice não pode ser inserido, a solução gerada é descartada. Embora o algoritmo consuma excessivamente os recurso computacionais, ele produz boas soluções aplicados aos problemas de Solomon [Solomon (1987)].

Alvarenga apresentou em [Alvarenga (2004)] o algoritmo *HGC1* que prioriza a minimização da distância total (como neste trabalho). Implementou operadores de mutação baseados na migração de consumidores entre rotas encontrando resultados satisfatórios.

Capítulo 3

Proposta deste trabalho

3.1 Algoritmo Genético

Este trabalho iniciou com o intuito de produzir um Algoritmo Genético eficiente (propondo novos operadores) para a resolução do Problema de Roteamento de Veículos com Janela de Tempo, dentro de seu conceito padrão, estipulado por Holland em [Holland (1975)], sendo representado de maneira sucinta:

Algoritmo 2 Algoritmo Genético Clássico

1: $P \leftarrow$População Inicial;
2: **enquanto** condição não satisfeita **faça**
3: $P' \leftarrow$Seleção(P);
4: $P \leftarrow$Cruzamentos(P');
5: $P \leftarrow$Mutações(P);
6: **fim enquanto**
7: Solução $\leftarrow$Melhor indivíduo(P);

3.1.1 População Inicial

Na criação da população inicial do algoritmo genético neste trabalho, foram utilizadas variações do algoritmo *Push-Forward Insertion Heuristic*. Como descrito em [Alvarenga (2004)], a execução da heurística *PFIH* (seção 2.6.1) sobre uma

determinada instância gera apenas uma solução, não sendo suficiente para produzir a diversificação entre os indivíduos da população do Algoritmo Genético. Para contornar este problema, são propostas neste trabalho duas estratégias para tal diversidade:

- Variação sobre α, β e γ (constantes da fórmula 2.16), dando-as valores aleatórios no intervalo de 0 a 1.

- Eliminação da fórmula 2.16 na determinação da ordem dos indivíduos a serem roteados, construindo esta fila de maneira totalmente aleatória.

Tais estratégias são suficientes para gerar indivíduos diferentes. A segunda é capaz de gerar uma população com maior diversidade genética, mas em conseqüência disso, traz consigo o fato de produzir indivíduos com um alto valor da função objetivo, se comparados com os resultados da fórmula original. Para um bom funcionamento do Algoritmo Genético, fica proposto uma dosagem dentre tais variações para poder assim construir uma população de variabilidade genética capaz de não se prender facilmente a mínimos locais. O tamanho da população utilizada neste trabalho foi de 30 indivíduos.

3.1.2 Seleção

Este trabalho implementou duas estratégias de seleção:

- Método do torneio;

- Método do roleta giratória.

A estratégia de seleção que melhor se adaptou na aplicação do Algoritmo Genético no *PRVJT* foi a roleta. O torneio demonstrou uma convergência muito rápida para mínimos locais, apresentando uma deficiência na variabilidade da população, tornando os indivíduos muito parecidos no decorrer das gerações.

3.1.3 Operadores de Cruzamento

Uma estratégia de cruzamento proposta por Alvarenga em [Alvarenga (2004)] para a resolução do Problema de Roteamento de Veículo com Janela de Tempo se baseia na escolha aleatória de rotas dos indivíduos sendo cruzados, onde estas são inseridas no indivíduo resultante. Se pelo menos um dos consumidores da rota a ser inserida no novo indivíduo pertencer a alguma rota já inserida na solução, esta rota é descartada. Se ao fim das tentativas de inserções de rotas, ainda existirem consumidores fora da solução (pelo fato das intersecções), estes são enxertados na solução através do algoritmo *PFIH*. Vale ressaltar que este operador tenta equilibrar o número de rotas vindos de ambos os pais.

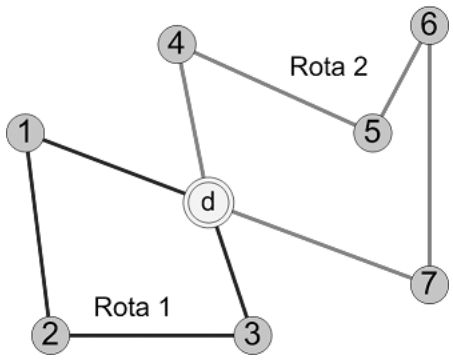

Figura 3.1: Indivíduo 1

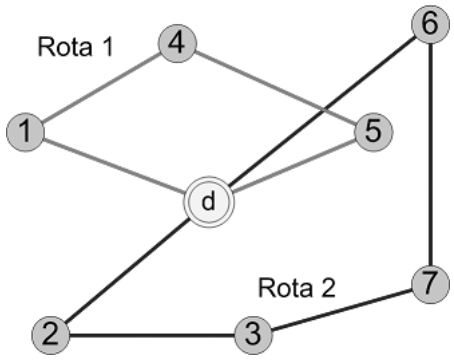

Figura 3.2: Indivíduo 2

Observando as figuras 3.1 e 3.2, suponha que a rota número 2 do indivíduo 2 ($I2R2$), com o conjunto de consumidores $\{2, 3, 6, 7\}$, é a primeira escolhida para compor o resultado do cruzamento. Para qualquer rota escolhida do indivíduo 1, sempre existirá uma intersecção entre das rotas $I1R1$ e $I1R2$ com a rota $I2R2$ já presente no indivíduo resultante.

$$I1R1 \cap I2R2 \neq \emptyset \tag{3.1}$$

$$I1R2 \cap I2R2 \neq \emptyset \tag{3.2}$$

Por este fato, as rotas $I1R1$ e $I1R2$ são descartadas, e a rota $I2R1$ é inserida na solução, completando o indivíduo, sendo exatamente igual a um de seus pais (indivíduo 2).

Esta estratégia de cruzamento pode produzir indivíduos com códigos genéticos exclusivos de ambos os pais, ou ainda, parte do código genético dos pais, e uma parte código genético mutante (enxerto executado pelo algoritmo *PFIH*), ou podendo ser também exatamente igual a um dos pai, como no exemplo desta seção.

3.1.4 Operadores de Mutação

Duas classes de operadores testadas neste trabalho se adaptaram bem ao Problema de Roteamento de Veículos com Janela de Tempo:

- Operadores de migração, transferindo consumidores, um a um;

- Operadores com pontos de corte, transferindo conjuntos de consumidores;

Um operador de migração implementado é determinado pela escolha de uma rota qualquer do indivíduo, onde nesta, todos os consumidores serão testados em todas as posições das outras rotas, encaixando-o na melhor posição possível caso não viole nenhuma restrição. Este operador pode ser implementado para efetuar a migração somente se encontrar uma melhor solução (intensificação), ou migrar mesmo se isso signifique uma piora na função objetivo (diversificação). Um exemplo de mutação efetuando a migração de consumidores pode ser observado nas figuras 3.3 e 3.4.

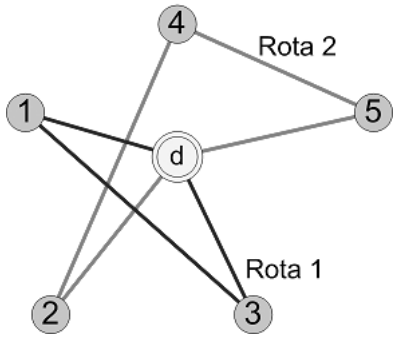

Figura 3.3: Indivíduo antes da mutação de migração

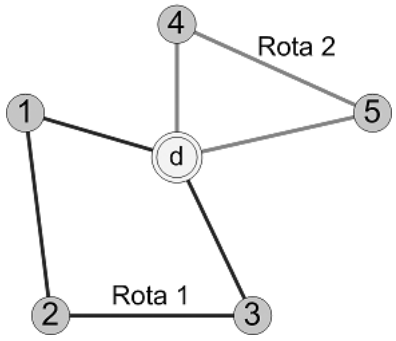

Figura 3.4: Indivíduo depois da mutação de migração

Suponha que a rota escolhida aleatoriamente seja a rota número 2 da figura 3.3, tentando encaixar seus consumidores $\{2, 4, 5\}$ na rota 1. Supondo que o algoritmo detecte que os consumidores 4 e 5 não podem migrar para a rota número 1 por violar restrições de tempo ou carga, e o consumidor número 2 pode migrar e ainda oferece um ganho sobre a função objetivo (neste caso, diminuindo a distância). Sendo assim, o consumidor 2 seria encaixado entre os consumidores 1 e 3 da rota número 1, como mostra a figura 3.4.

Os operadores de mutação do tipo corte tem a capacidade de migrar todo um sub-conjunto de rota. Observe na figura 3.5 que a rota número 2 contém 2 consumidores $\{3, 4\}$ que podem migrar para a rota número 1, ocasionando uma melhora na função objetivo. Neste caso, este operador efetuaria o corte entre os consumidores 4 e 6 da rota número 2, tentando migrar a ramificação $\{3, 4\}$ para as demais rotas, como mostra a figura 3.6. A ramificação pode não ser transportada com a mesma seqüência para a rota número 1. Normalmente é utilizada uma adaptação da heurística *PFIH* para decidir onde os consumidores se encaixarão.

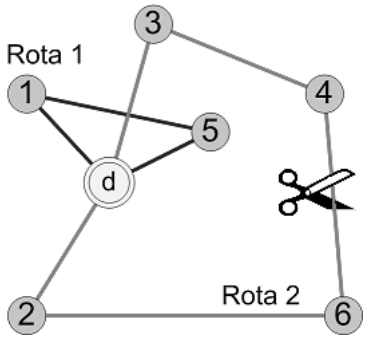

Figura 3.5: Indivíduo antes da mutação de corte

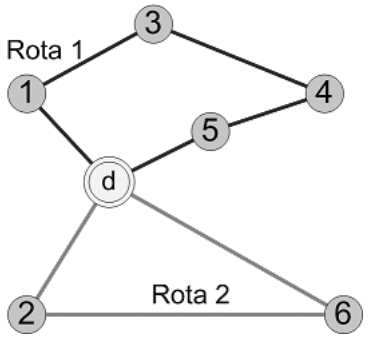

Figura 3.6: Indivíduo depois da mutação de corte

Note que estes operadores também podem ser aplicados a qualquer classe dos problema de roteamento de veículos. Como o *PRVJT* possui um alto número de restrições, estes simples operadores não são suficientes para proporcionar uma evolução satisfatória para toda a heterogeneidade das instâncias. Por este motivo, foram desenvolvidas variações para alcançar resultados satisfatórios. Uma dessas variações é o operador com duplo ponto de corte, produzindo 3 fragmentos de rota. Os fragmentos das extremidades são retirados da rota original para o melhor roteamento dos consumidores.

Suponha que uma rota R já atende C consumidores e estes já ocupam toda a capacidade de carga do veículo. Para qualquer outro consumidor roteado nesta rota R, por qualquer operador de mutação, não haverá intervalo para tal operação sem que haja uma restrição de carga violada. Para tal dificuldade, este trabalho propõe um operador de mutação que retira K consumidores da rota R, com K variando de 1 até o número de consumidores de R. Após esta retirada, pode-se direcionar este conjunto de rotas (solução parcial do problema, pois não contém todos os consumidores) para qualquer outro operador de mutação. Após sua ação, os consumidores previamente retirados da solução devem ser inseridos na mesma, utilizando qualquer heurística de roteamento, por exemplo, o *PFIH*. Esta estratégia na fuga de mínimos locais também é eficiente para restrições temporais dos consumidores (janela de tempo), não se limitando apenas a capacidade de carga dos veículos.

Outro operador de mutação implementado exerce a retirada do w consumidores da solução (aleatoriamente), onde w é maior ou igual a 0 e menor que o número total de consumidores da solução. Após esta retirada, o indivíduo é reaplicado a função de mutação do algoritmo genético. Após a aplicação os consumidores retirados a priori são inseridos no indivíduo resultante. Esta operação possibilita que outros operadores possam agir na solução já que esta pode estar inserida em um mínimo local com dificuldades de ampliar seu espaço de busca.

Neste trabalho o método utilizado da aplicação dos operadores de cruzamento aceita a aplicação de vários operadores em seqüência, como também pode ocorrer o caso de nenhum operador ser aplicado no indivíduo.

No caso da aplicação de um operador de mutação gerar o melhor indivíduo da população, imediatamente o indivíduo deve ser retornado, evitando o caso de outro operador piorar a melhor solução até então encontrada.

Algoritmo 3 Aplicar operadores de mutação (indivíduo)

1: $P1 \leftarrow probabilidadePercentualDeAplicarOperador01$;
2: $P2 \leftarrow probabilidadePercentualDeAplicarOperador02$;
3: **se** $ValorAleatorio(0, 100) < P1$ **então**
4: indivíduo $\leftarrow aplicarOperador01$;
5: **se** indivíduo.$distanciaTotal$ <MelhorIndivíduo.$distanciaTotal$ **então**
6: $Retornar$ indivíduo;
7: **fim se**
8: **fim se**
9: **se** $ValorAleatorio(0, 100) < P2$ **então**
10: indivíduo $\leftarrow aplicarOperador02$;
11: **se** indivíduo.$distanciaTotal$ <MelhorIndivíduo.$distanciaTotal$ **então**
12: $Retornar$ indivíduo;
13: **fim se**
14: **fim se**
15: ...;
16: $Retornar$ indivíduo;

3.2 Algoritmo Evolutivo

Este trabalho implementou um Algoritmo Evolutivo, baseado no Algoritmo Genético descrito na seção 3.2 excluindo seu operador de cruzamento. Outra diferença básica foi a seleção de indivíduos, que por causa da exclusão do cruzamento, onde dois indivíduos geram apenas um descendente, foi necessário selecionar exatamente o número de indivíduos da população, diferente do Algoritmo Genético, onde é necessário selecionar o dobro da população.

As outras características do Algorimo Genético foram mantidas, e uma comparação entre estas duas meta-heurísticas é feita na seção 4.2.

3.3 Base de Testes

Existe uma grande quantidade de publicações utilizando heurísticas na resolução do *PRVJT*. Para descobrir a qualidade e robustez dos algoritmos, estes são apli-

cados sobre as instâncias de Solomon [Solomon (1987)], justamente pelo fato de haver um maior número de resultados publicados sobre elas, possibilitando portanto fácil comparação e análise.

Os testes deste trabalho foram feitos sobre as instâncias de Solomon com 100 consumidores, podendo ser encontradas em [Intances].

3.4 Tecnologias e Padrões

A linguagem utilizada para o desenvolvimento deste trabalho foi Java, sobre a máquina virtual da empresa Sun microsystems (J2SDK 5.0) [J2SE]. A escolha desta linguagem se baseou no fato de da Máquina Virtual Java oferecer um *framework* muito vasto já implementado, facilitando a codificação, além da própria linguagem facilitar no desenvolvimento orientado a objetos.

Toda o sistema foi baseado na Convenção de Codificação Java [JCC], facilitando o entendimento de terceiros. Além disso, todo o algoritmo foi implementado afim de garantir uma documentação do código fonte, sendo possível através do sistema [JavaDoc].

Os testes foram efetuados em diferentes sistemas operacionais: *Windows XP Professional* [Windows XP] e *Slackware 10.0 (kernel linux* $2.4.26$*)* [Slackware], sendo que ambos os sistemas apresentaram tempos de execução do algoritmo muito semelhantes.

O computador utilizado para a coleta dos resultados foi um Intel Pentium IV, 2.4 GHz, 512 KB de memória cache L2, com 1024 MB de memória RAM.

Capítulo 4

Resultados

Para a avaliação dos resultados encontrados neste trabalho, o mesmo foi comparado aos resultados de outras publicações que também utilizam a distância total como primeiro objetivo na minimização do Problema de Roteamento de Veículos com Janela de Tempo ([Taillard (1993)], [Rousseau (1999)], [Tan et.Al. (2001)] e [Alvarenga (2004)]). Todos eles foram aplicados sobre as 56 instâncias de Solomon, onde cada um possui 100 consumidores.

As instâncias de Solomon são divididas em seis classes: R1, R2, C1, C2, RC1 e RC2. As instâncias dos tipos R1 e R2 apresentam consumidores com as coordenadas euclidianas totalmente aleatórias. Já as instâncias dos tipos C1 e C2, apresentam os consumidores de maneira aglomerada (em *clusters*). As instâncias dos tipos RC1 e RC2 apresentam um misto das duas primeiras características (esparsos e aglomerados). Uma característica entre os tipos R1, C1 e RC1 é que suas instâncias possibilitam que poucos consumidores sejam atendidos por um veículos, necessitando de um número maior de veículos para atender todo a demanda. Já os tipos R2, C2 e RC2 apresentam na solução poucos veículos, assim atendendo uma grande quantidade de consumidores em cada rota. As instâncias de Solomon podem ser encontradas no site [Intances].

O algoritmo implementado neste trabalho foi nomeado *AE1* (Adaptação Evolutivo 1) e é referenciado nas tabelas de resultados como [Oliveira (2005)]. Foram

colhidas 11 amostras da execução do *AE1*. As médias dos resultados, agrupadas pelos tipos das instâncias são comparadas com os outros algoritmos que também consideram o minimização de distância, como principal objetivo. Os resultados apresentam o número médio de veículos, distância média percorrida e o percentual a que cada resultado está dos melhores resultados publicados na literatura (com relação a média da distância total).

A seção 4.1 apresenta os resultados obtidos neste trabalho (para todas as instâncias de Solomon). A seção 4.2 apresenta a média dos resultados deste e de outros trabalhos, calculando suas qualidades, através da distância entre os melhores resultados obtidos na minimização da distância total para o *PRVJT*. Já a seção 4.3 expõe tabelas com os melhores resultados já obtidos na minimização da distância total para cada instância de Solomon.

4.1 Resultados na minimização de distância do *PRVJT*

As tabelas desta seção representam os resultados obtidos pelo algoritmo implementado neste trabalho para as 56 instâncias de Solomon.

Tabela 4.1: Resultados deste trabalho: classe R1

Instância	Veículos	Distância
R101	20	1644.82
R102	18	1476.67
R103	14	1226.03
R104	11	997.66
R105	16	1372.66
R106	13	1250.28
R107	11	1087.13
R108	10	957.20
R109	13	1173.42
R110	12	1105.03
R111	12	1076.20
R112	11	977.03

Tabela 4.2: Resultados deste trabalho: classe C1

Instância	Veículos	Distância
C101	10	828.94
C102	10	828.94
C103	10	828.06
C104	10	824.78
C105	10	828.94
C106	10	828.94
C107	10	828.94
C108	10	828.94
C109	10	828.94

Tabela 4.3: Resultados deste trabalho: classe RC1

Instância	Veículos	Distância
RC101	17	1656.62
RC102	15	1484.91
RC103	12	1281.11
RC104	11	1162.00
RC105	16	1548.93
RC106	14	1402.44
RC107	13	1258.47
RC108	11	1140.72

Tabela 4.4: Resultados deste trabalho: classe R2

Instância	Veículos	Distância
R201	8	1159.12
R202	5	1044.27
R203	5	893.31
R204	4	746.81
R205	5	964.72
R206	4	887.90
R207	4	811.93
R208	3	707.01
R209	5	866.40
R210	4	930.66
R211	4	761.75

Tabela 4.5: Resultados deste trabalho: classe C2

Instância	Veículos	Distância
R201	3	591.56
R202	3	591.56
R203	3	591.17
R204	3	590.60
R205	3	588.88
R206	3	588.49
R207	3	588.29
R208	3	588.32

Tabela 4.6: Resultados deste trabalho: classe RC2

Instância	Veículos	Distância
RC201	5	1304.20
RC202	5	1118.66
RC203	5	945.44
RC204	4	788.66
RC205	7	1162.57
RC206	5	1074.01
RC207	5	970.78
RC208	4	779.84

Foram descritos os melhores resultados obtidos em 11 amostras da execução do Algoritmo Evolutivo.

Para uma análise da qualidade das soluções, foram calculadas as médias das distâncias totais encontradas para todas as amostra, e a partir dai foi subtraido o desvio padrão para avaliar se o Algoritmo Evolutivo tem a capacidade de encontrar os resultados com uma constância desejada. O desvio padrão para a média dos veículos foi de 0.0374 e o desvio padrão para a distância total foi de 1.6703. Os valores podem ser considerados baixos, visto que a média de veículos é de 8.48 e a da distância total é de 988.77.

4.2 Comparação por classes do *PRVJT*

Este trabalho superou ou igualou os melhores resultados da literatura nas classes de problemas C1, R2, C2 e RC2, apresentando-se inferior somente nas classes R1 e RC1, sendo que na classe R1 ele superou individualmente cada autor comparado, mas não chegando a superar a união dos melhores resultados publicados até então. O algoritmo implementado neste trabalho é representado pela referência [Oliveira (2005)].

Tabela 4.7: Quadro comparativo: classe R1

Trabalho	Veículos	Distância	% Distância
Melhores publicados	13.00	1189.65	-
Taillard (1997)	12.17	1209.35	+1.6559%
Rousseau (1999)	12.83	1201.10	+0.9625%
Tan (2001)	13.83	1260.71	+5.9732%
Alvarenga (2004)	13.33	1200.05	+0.8742%
Oliveira (2005)	13.42	1195.34	+0.4783%

Tabela 4.8: Quadro comparativo: classe C1

Trabalho	Veículos	Distância	% Distância
Melhores publicados	10.00	828.38	-
Taillard (1997)	10.00	828.38	0.0000%
Rousseau (1999)	10.00	828.38	0.0000%
Tan (2001)	10.11	858.81	+3.6734%
Alvarenga (2004)	10.00	828.38	0.0000%
Oliveira (2005)	10.00	828.38	0.0000%

Tabela 4.9: Quadro comparativo: classe RC1

Trabalho	Veículos	Distância	% Distância
Melhores publicados	12.75	1340.54	-
Taillard (1997)	11.50	1389.22	+3.6314%
Rousseau (1999)	12.50	1370.26	+2.2170%
Tan (2001)	13.63	1447.06	+7.9461%
Alvarenga (2004)	13.00	1344.89	+0.3245%
Oliveira (2005)	13.62	1366.90	+1.9664%

Tabela 4.10: Quadro comparativo: classe R2

Trabalho	Veículos	Distância	% Distância
Melhores publicados	3.92	907.83	-
Taillard (1997)	2.82	980.27	+7.9795%
Rousseau (1999)	3.18	966.94	+6.5111%
Tan (2001)	3.82	1058.52	+16.5989%
Alvarenga (2004)	4.64	910.17	+0.2576%
Oliveira (2005)	4.64	880.53	-3.0072%

Tabela 4.11: Quadro comparativo: classe C2

Trabalho	Veículos	Distância	% Distância
Melhores publicados	3.00	589.86	-
Taillard (1997)	3.00	589.86	0.0000%
Rousseau (1999)	3.00	594.01	+0.7036%
Tan (2001)	3.25	617.10	+4.6180%
Alvarenga (2004)	3.00	589.86	0.0000%
Oliveira (2005)	3.00	589.86	0.0000%

Tabela 4.12: Quadro comparativo: classe RC2

Trabalho	Veículos	Distância	% Distância
Melhores publicados	5.50	1030.30	-
Taillard (1997)	3.38	1117.44	+8.4577%
Rousseau (1999)	3.75	1113.29	+8.0549%
Tan (2001)	7.00	1169.41	+13.5019%
Alvarenga (2004)	6.00	1032.88	+0.2504%
Oliveira (2005)	5.00	1018.02	-1.1919%

Um dos testes efetuados durante o projeto foi a execução do algoritmo genético sem o operador de cruzamento proposto por Alvarenga em [Alvarenga (2004)] (explicado na seção 3.1.3 deste trabalho). Este teste merece destaque pois demonstra que tal operador não surte efeitos benéficos perante os resultados, ou seja, não traz vantagens sobre o objetivo do problema (minimização da distância total). Note que esta afirmação não condena tal operador como sendo ruim dentro de qualquer algoritmo genético, mas sim que dentro do contexto deste sistema, ele não destacou-se perante os resultados finais. Por este motivo o algoritmo foi nomeado de “Algoritmo Evolutivo 1"(*AE1*), pois é executado somente com mutações. A seguir temos a tabela com a média dos resultados, categorizados pelas classes de problemas, representando a execução do algoritmo com o cruzamento (*AG1*) e a execução do algoritmo sem o cruzamento (*AE1*).

Tabela 4.13: Comparativo entre *AG1* e *AE1*

	AG1		*AE1*	
Classe	Veículos	Distância	Veículos	Distância
R1	13.33	1201.69	13.42	1195.34
C1	10.00	828.38	10.00	828.38
RC1	13.25	1385.51	13.62	1366.90
R2	3.82	897,92	4.64	880.53
C2	3.00	589,86	3.00	589.86
RC2	4.25	1035,07	5.00	1018.02

Ambos os algoritmos foram limitados a executar durante 30 minutos para cada instância. Nota-se uma melhor média de veículos na execução do *AG1* se com-

parado ao *AE1*, mas em todos os casos foi igual ou inferior na distância total percorrida.

4.3 Melhores resultados da literatura para o *PRVJT* na minimização de distância

Esta seção indica melhores resultados já obtidos pela literatura para o Problema de Roteamento de Veículos com Janela de Tempo considerando a minimização da distância total como primeiro objetivo. Os resultados são mostrados indicando a instância, número de veículos, distância total e autor do resultado.

Muitos autores igualam alguns destes melhores resultados da literatura. Isso ocorre principalmente nas classes C1 e C2, onde por exemplo, este trabalho igualou todos os resultados de Rouchat [Rouchat et.Al. (1995)]. O autor indicado pela referência em cada instância é aquele que primeiro publicou seu respectivo resultado. Este trabalho possui os melhores resultados, até então publicados, em 36 instâncias de Solomon, sendo que em 19 delas foi o primeiro trabalho a apresentar tais resultados.

Tabela 4.14: Melhores resultados da literatura: classe R1

Instância	Veículos	Distância	Autor
R101	20	1642.88	[Alvarenga (2004)]
R102	18	1472.62	[Alvarenga (2004)]
R103	14	1213.62	[Rouchat et.Al. (1995)]
R104	11	997.42	[Alvarenga (2004)]
R105	15	1372.53	[Alvarenga (2004)]
R106	13	1249.49	[Alvarenga (2004)]
R107	11	1087.13	[Oliveira (2005)]
R108	10	957.20	[Oliveira (2005)]
R109	13	1173.42	[Oliveira (2005)]
R110	11	1080.36	[Rouchat et.Al. (1995)]
R111	12	1068.08	[Alvarenga (2004)]
R112	10	953.63	[Rouchat et.Al. (1995)]

Tabela 4.15: Melhores resultados da literatura: classe C1

Instância	Veículos	Distância	Autor
C101	10	828.94	[Rouchat et.Al. (1995)]
C102	10	828.94	[Rouchat et.Al. (1995)]
C103	10	828.06	[Rouchat et.Al. (1995)]
C104	10	824.78	[Rouchat et.Al. (1995)]
C105	10	828.94	[Rouchat et.Al. (1995)]
C106	10	828.94	[Rouchat et.Al. (1995)]
C107	10	828.94	[Rouchat et.Al. (1995)]
C108	10	828.94	[Rouchat et.Al. (1995)]
C109	10	828.94	[Rouchat et.Al. (1995)]

Tabela 4.16: Melhores resultados da literatura: classe RC1

Instância	Veículos	Distância	Autor
RC101	15	1623.58	[Rouchat et.Al. (1995)]
RC102	13	1477.54	[Rouchat et.Al. (1995)]
RC103	11	1261.67	[Shaw (1998)]
RC104	10	1135.48	[Cordeau et.Al. (2000)]
RC105	16	1518.60	[Alvarenga (2004)]
RC106	13	1381.46	[Alvarenga (2004)]
RC107	12	1212.83	[Alvarenga (2004)]
RC108	11	1212.83	[Alvarenga (2004)]

Tabela 4.17: Melhores resultados da literatura: classe R2

Instância	Veículos	Distância	Autor
R201	8	1159.12	[Oliveira (2005)]
R202	3	1144.27	[Oliveira (2005)]
R203	5	893.31	[Oliveira (2005)]
R204	4	746.81	[Oliveira (2005)]
R205	5	964.72	[Oliveira (2005)]
R206	4	887.90	[Oliveira (2005)]
R207	4	811.93	[Oliveira (2005)]
R208	3	707.01	[Oliveira (2005)]
R209	4	866.40	[Oliveira (2005)]
R210	4	930.66	[Oliveira (2005)]
R211	4	761.75	[Oliveira (2005)]

Tabela 4.18: Melhores resultados da literatura: classe C2

Instância	Veículos	Distância	Autor
R201	3	591.56	[Rouchat et.Al. (1995)]
R202	3	591.56	[Rouchat et.Al. (1995)]
R203	3	591.17	[Rouchat et.Al. (1995)]
R204	3	590.60	[Rouchat et.Al. (1995)]
R205	3	588.88	[Rouchat et.Al. (1995)]
R206	3	588.49	[Rouchat et.Al. (1995)]
R207	3	588.29	[Rouchat et.Al. (1995)]
R208	3	588.32	[Rouchat et.Al. (1995)]

Tabela 4.19: Melhores resultados da literatura: classe RC2

Instância	Veículos	Distância	Autor
RC201	8	1301.43	[Alvarenga (2004)]
RC202	7	1116.23	[Alvarenga (2004)]
RC203	5	945.44	[Oliveira (2005)]
RC204	4	788.66	[Oliveira (2005)]
RC205	7	1161.81	[Alvarenga (2004)]
RC206	7	1074.01	[Oliveira (2005)]
RC207	5	970.78	[Oliveira (2005)]
RC208	4	779.84	[Oliveira (2005)]

Capítulo 5

Conclusão

Este trabalho apresenta uma importante colaboração para o Tratamento do Problema de Roteamento de Veículos com Janela de Tempo. Os resultados sobre as instâncias de Solomon alcançados pelo algoritmo *AE1* podem ser considerados expressivos, já que superam ou igualam os melhores resultados da literatura em cinco das seis classes do problema.

Um fato a se destacar, é a adaptação do Algoritmo Genético, que se comportou bem perante o problema, mesmo não efetuando nenhum tipo de cruzamento durante a execução do algoritmo, sendo representado pela simplificação:

Algoritmo 4 Algoritmo Genético Simplificado

1: $P \leftarrow$População Inicial;
2: **enquanto** condição não satisfeita **faça**
3: $P' \leftarrow$Seleção(P);
4: $P \leftarrow$Mutações(P');
5: **fim enquanto**
6: Solução $\leftarrow$Melhor indivíduo(P);

Esta simplificação não propõe a não utilização da operação de cruzamento para o *PRVJT*, nem mesmo afirma que o operador aqui utilizado não possui utilidade,

mas sim declara que no contexto deste trabalho, o operador implementado não resultou em melhoras sobre os resultados finais.

A partir do momento que este algoritmo passa a executar sem a operação de cruzamento (característica do algoritmo genético) ele passa a ser um algoritmo evolutivo.

Este trabalho encontrou melhores resultados nas classes R2, C2 e RC2, onde em 27 instâncias, ele superou ou igualou os resultados em 24 delas, sendo inferior em apenas 3 delas (RC201, RC202 e RC205). Isso demonstra que o algoritmo esta melhor adaptado para instâncias onde o número de veículos é pequeno, atendendo um número maior de consumidores em cada rota.

Capítulo 6

Proposta para trabalhos futuros

Implementar novos operadores que se adaptem melhor nas classes R1, C1 e RC1, onde o número de consumidores atendidos em cada rota é relativamente pequeno.

Outra implementação seria a de operadores para reduzir o número de veículos, alterando o objetivo principal do problema (minimizar a priori o número de veículos, para posteriormente atacar a distância total), já que a redução da distância total apresentou bons resultados.

Um outra proposta para trabalho futuro seria a avaliação experimental rigorosa para determinar qual é a verdadeira influência de cada operador dentro do algoritmo evolutivo, buscando novas probabilidades de aplicação destes operadores para alcançar melhores resultados.

Referências Bibliográficas

[Alvarenga (2004)] G.B. Alvanrega, G. R. Mateus. ***A Two-Phase Genetic and Set Partitioning Approach for the Vehicle Routing Problem with Time Windows*** , Fourth International Conference on Hybrid Intelligent Systems (HIS04), IEEE Computer Society Press, (2004).

[Andrade et.Al. (2004)] Andrade, Carlos Eduardo; Batista, Flávio Lúcio Nogueira; Toso, Rodrigo Franco. ***Modelo de Otimização para Transporte de Cargas em Ambientes Reduzidos***; Monografia, Departamento de Ciência da Computação, Universidade Federal de Lavras. p. 24-25 (2004).

[Arakari (1998)] Reinaldo Gen Ichiro Arakaki ***O Problema de Roteamento de Veículos e algumas Metaheurísticas***, Instituto Nacional de Pesquisas Espaciais - INPE, (1998);

[Best Know Solution] Best Known Solutions Identified by Heuristics for Solomon's (1987)., ***URL: http://www.sintef.no/static/am/opti/projects/top/vrp/bknown.html***, 27 de Outubro de 2004.

[Backer et.Al. (1986)] E. K. Backer; J. R. Schaer; ***Solution improvement heuristic for the vehicle routing problem with time window constraints***, (1986).

[Blanton et.Al. (1993)] Blanton, J.L.; Wainwright, R. L. ***Multiple vehicle routing with time and capacity constraints using genetic algorithms***. In Proceeding of the 5th International Conference on Genetic Algorithms. Forrest, S. editor. p. 452-459. (1993).

[Bodin et.Al. (1983)] Bodin, L., Golden, B., Assad, A., Ball, M. ***Routing and scheduling of vehicles and crews - the state of the art***, Comput. Oper. Res. 10, 63-211, (1983).

[Carlos (2002)] Carlos, Luiz Amorim; ***Computação Evolutiva***, Mini-curso, ERMAC, (2002);

[Chistifides et.Al. (1979)] Christofides, N; Mingozzi, A. ; Toth, P. ***The vehicle routing problem***. In Christofides, Mingozzi, Toth, Sandi, editors. Combinatorial Optimization, p. 315-338. John Wiley & Sons, (1979);

[Cook (1971)] S. A. Cook; ***The Complexity of Theorem-Proving Procedures***, Symp. on Theory of computing, (1971);

[Cordeau et.Al. (2000)] J.-F. Cordeau, G. Laporte, and A. Mercier; ***A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows***, Working Paper CRT-00-03, Centre for Research on Transportation, Montreal, Canada, (2000).

[Cormen et.Al. (1999)] Thomas H. Cormen; Charles E. Leiserson; Ronald L. Rivest; ***Introduction to algorithms***, The MIT Press, (1999);

[Dantzig et.Al. (1959)] G.B.Dantzig, R.H. Ramser; ***The Truck Dispatching Problem Management Science***, 6, 80 (1959).

[Darwin (1859)] Charles Darwin; ***On The Origin of Species***, 1st edition, Harward University Press, MA, (1859).

[Dorigo et.Al. (1991)] Dorigo, M.; Maniezzo, V.; Colorni A; ***The Ant System: An Autocatalytic Optimizing Process***, Technical Report No. 91-016 Revised, Politecnico di Milano, Italy. (1991).

[Dorigo et.Al. (1999)] Dorigo, Di Caro, G.; Cambardella L. M.; ***Ant Algorithms for Discrete Optimization***, Artificial Life, 5(2):137-172. Also available as Technical Report No. 98-10 (IRIDIA), Université Libre de Bruxelles, Belgium (1999).

[Dror et.Al. (1986)] Dror, M. ; Levy, L. ***A vehicle routing improvement algorithm comparison of a “greedy"and a matching implementation for inventory routing***, Computers and Operations Research, 13. p. 33-45, (1986).

[Fisher et.Al. (1997)] Marshall L. Fisher; Kurt O. J"ornsteen; Oli B. G.Madsen ***Vehicle routing with time windows: Two optimization algorithms***, Operations Research, 45(3):488 - 492, (1979).

[Gendreau et.Al. (1992)] Gendreau, M; Hertz, A; Laporte, G. ***New insertion and post optimization procedures for the traveling salesman problem***. Operations Research, 40, p. 1086-1094. (1992).

[Gendreau et.Al. (1994)] Gendreau, M; Hertz, A; Laporte, G. ***A tabu search heuristic for the vehicle routing problem***. Management Science, 40, p. 1276-1290. (1994).

[Glover (1986)] Glover, Fred; ***Future Paths for Integer Programming and Link to Artificial Intelligence***, Computers and Operations Research, 13, 533-554, (1986);

[Gillet et.Al. (1974)] Gillet, B.E.; Miller, L.R. ***A heuristic algorithm for the vehicle-dispatch problem***, Operations Research, 22(2):p 340-349. (1974);

[Hjorring (1995)] Hjorring, C. A. ***The vehicle routing problem and local search metaheuristics***. Phd Thesis. The University of Aucland. (1995).

[Holland (1975)] J. H. Holland; ***Adaptation in natural and artificial systems***, (1975).

[Intances] Solomon's 100 customers Problems Instances (1987), ***URL: http://neo.lcc.uma.es/radiaeb/WebVRP/data/instances/solomon/solomon_100.zip***, 27 de Outubro de 2004.

[J2SE] Sun microsystems - Java Virtal Machine 5.0, ***URL: http://java.sun.com***, 19 de Novembro de 2004.

[JavaDoc] How to Write Doc Comments for the Javadoc Tool, ***URL: http://java.sun.com/j2se/javadoc/writingdoccomments/***, 28 de Novembro de 2004.

[JCC] Code Conventions for the Java Programming Language, ***URL: http://java.sun.com/docs/codeconv/***, 28 de Novembro de 2004.

[Johnson et.Al. (1979)] R. Michael Garey; David Johnson; ***Computers and intractably: A guide to the theory of $\mathcal{NP}$-completeness***, (1979).

[Juel et.Al. (1991)] San R. Thangiah; Kendall Nygard. Paul L. Juel. Gideon; ***A genetic algorithm for vehicle routing problem with time windows***, (1991).

[Kaan et.Al. (1987)] A. W. J. Kolen; A. H. G. R. Kaan; H. W. J. M. Trienekens. ***Vehicle Routing Problem with Time Windows***, (1987).

[Kirkpatrick et.Al. (1983)] Kirkpatrick, S., Gelatt, C. D. and Vecchi, P. M. ***Optimization by Simulated Annealing***, Science 220, 671-680, (1983);

[Larsen (1999)] J. Larsen; ***Parallelization of the vehicle routing problem with time windows***, Phd Thesis, Department of Mathematical Modelling, Technical University of Denmark, (1999);

[Larsen (2000)] J. Larsen; ***The Dynamic Vehicle Routing***, Phd Thesis, Department of Mathematical Modelling, Technical University of Denmark, (2000);

[Melo (2001)] MELO, V. A.; ***Metaheurísticas para o Problema do Caixeiro Viajante com Coleta de Prêmios***, Dissertação de Mestrado, Instituto de Computação, Universidade Federal Fluminense, Niterói, Rio de Janeiro, (2001);

[Metropolis et.Al. (1953)] Metropolis, N.; A. W, Rosenbluth; M. N. Rosenbluth, A. H. Teller and E. Teller ***Equation of State Calculation by Fast Computing Machines***, Journal of Physical Chemistry, 21, 1087-1092, (1953);

[Michalewicz (1992)] Michalewicz, Z; ***Genetic algorithms + Data Structures = Evolution Programs***, Springer-Verlag, (1994);

[Osman (1993)] Osman, I. ***Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem***, Annals of Operations Research, 41 p. 421-451. (1993).

[Qily (1999)] Kenny Zhu Qily; ***Heuristic methodos for vehicle routing problem with time windows***, Department of Electrical Engineering, National, University of Singapore, (1999).

[Papadimitriou et.Al. (1982)] PAPADIMITRIOU, C. H.; STEIGLITZ, K. ***Combinatorial Optimization: Algorithms and Complexity***; USA, Dover Publications Inc., (1982).

[Passos et.Al. (2003)] R. M. Passos; Evandrino Barros; ***Heurísticas para o Vehicle Routing Problem with Time Windows***, (2003);

[Potivin et.Al. (1996)] Potvin, J.Y.; Bengio, S. ***The vehicle routing problem with time windows***; part II: genetic search. INFORMS Journal on Computing 8 (2): p. 165-172. (1996).

[Pureza et.Al. (1991)] Pureza, V.M.; França, P.M. ***Vehicle routing problems via tabu seach metaheuristic***. Technical Report CRT-747, Centre de Recherche sur les transport, Montreal, (1991).

[Rouchat et.Al. (1995)] Y. Rochat and E.D. Taillard; ***Probabilistic Diversification and Intensification in Local Search for Vehicle Routing***, Journal of Heuristics 1, 147-167, (1995).

[Rousseau (1999)] L. M. Rousseau, M. Gendreau and G. Pesant; ***Using Constraint-based Operators with Variable Neighborhood Search to Solve the Vehicle Routing Problem with Time Windows***, Presented at the CP-AI-OR'99 Workshop, February 25.-26., University of Ferrara, Italy, (1999).

[Semet et.Al. (1993)] Semet, F; Taillard, E. ***Solving real-life vehicle routing problems efficiently using tabu search***. Annals of Operations Research, 41, p. 469-488, (1993).

[Schulze et.Al. (1999)] Jürgen Schlze; Torsten Fahle; ***A parallel algorithm for vehicle routing problem with time windows constraints***, Technical Report, University of Paderborn, (1999);

[Slackware] Slackware 10.0, kernel linux 2.4.26, ***URL: http://www.slackware.com***,
19 de Novembro de 2004.

[Solomon (1986)] Marius M. Solomon; ***On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time windows constraints***, (1986).

[Solomon (1987)] Marius M. Solomon; ***Algorithms for the Vehicle Routing Problem and Scheduling Problem with Time Window Contraints***, (1987).

[Srivinas et.Al. (1994)] Srivinas, M.; Patnaik, L.M. ***Genetic algorithmics: A survey***. IEEE Computer Society, p. 17- 26. (1994).

[San et.Al. (2001)] Sam R. Thangiah; Ibrahim H. Osman; Tong Sun; ***Hybrid Genetic Algorithm, Simulated Annealing and Tabu Search Methods for Vehicle Routing Problems with Time Windows***, (2001);

[Shaw (1998)] P. Shaw; ***Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems***, In Principles and Practice of Constraint Programming - CP98, Lecture Notes in Computer Science, 417-431, M. Maher and J.-F. Puget (eds), Springer-Verlag, New York, (1998).

[Taillard (1993)] Taillard, E. ***Parallel iterative search methods for vehicle routing problems*** Networks, 23(8), p. 661-674. (1993).

[Tan et.Al. (2001)] K,C. Tan, L.H. Lee, Q.L. Zhu and K. Ou ***Heuristic methods for vehicle routing problem with time windows***, Artificial Intelligence in Engineering, 15, 281-295, (2001).

[Thagiah et.Al. (1991)] Thangiah, S. R.; Nygard, K.E.; Juell, P. L.. ***GIDEON: A Genetic algorithm System for vehicle routing with time windows***. In: Proceedings of the 7th IEEE Conference on Artificial Intelligence Applications, p. 322-328, Miami Beach, FL. (1991).

[Whitley et.Al. (1989)] Whitley, D.; Starkweather, Fuquay, D. ***Scheduling problems and traveling salesman: The genetic edge recombination operator***. In Proceedings of the 3th International Conference on Genetic Algorithms. Schaffer, S. editor. (1989).

[Willard (1989)] Willard, J.A. G. ***Vehicle routing using r-optimal tabu search***, Masther's thesis. The Management School. Imperial College. London. (1989);

[Windows XP] Microsoft Windows XP Professional, ***URL: http://www.microsoft.com/windowsxp/pro/default.mspx/***, 28 de Novembro de 2004.