Universidade Federal de Juiz de Fora

RAFAEL DE SOUZA MARQUES

Um Algoritmo GRASP para uma Aplicação do Problema de Roteamento de Veículos.

JUIZ DE FORA

2010

RAFAEL DE SOUZA MARQUES

Um Algoritmo GRASP para uma Aplicação do Problema de Roteamento de Veículos.

Monografia submetida ao Departamento de Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador:

Stênio Sã Rosário F. Soares

Universidade Federal de Juiz de Fora

JUIZ DE FORA

2010

Um Algoritmo GRASP para uma Aplicação do Problema de Roteamento de Veículos.

Rafael de Souza Marques

Monografia submetida ao Departamento de Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Aprovada por:

Prof. Stênio Sã Rosário F. Soares / DCC-UFJF (Orientador)

Prof. Ilaim Costa Junior / DCC-UFJF

Profa. Luciana Brugiolo Gonçalves / IC-UFF

Juiz de Fora, Agosto de 2010.

Resumo

O Problema de Roteamento de Veículos (PRV) definido há mais de 40 anos, é um dos problemas mais estudados na literatura devido a sua importância em uma série de aplicações práticas que afetam principalmente a indústria, o comércio e o setor de serviços. O PRV Clássico consiste na determinação de rotas que atendam a um conjunto de clientes por uma frota de veículos com o menor custo possível. De acordo com as variáveis e restrições de cada problema, tem-se uma variação do Problema de Roteamento de Veículos. Neste trabalho, a variação do PRV abordada é caracterizada pelo fato de que os clientes possuem prioridades de atendimentos diferenciadas. Além disso, a frota de veículos é heterogênea e o objetivo é determinar rotas de forma que o maior número de clientes seja atendido, dando preferência a clientes de maior prioridade. Para isso, foram propostas e comparadas heurísticas construtivas e heurísticas de busca local que irão compor uma metaheurística baseada na abordagem GRASP (*Greedy Randomized Adaptive Search Procedure*) para solucionar o problema.

Palavras-chave

1. Problema de Roteamento de Veículos
2. GRASP
3. Heurísticas
4. Metaheurísticas

Glossário

GRASP	:	*Greedy Randomized Adaptive Search Procedure*
PRV	:	Problema de Roteamento de Veículos
LRC	:	Lista Restrita de Candidatos

Sumário

Lista de Figuras **vii**

Lista de Tabelas **viii**

1 Introdução **1**

2 O Problema de Roteamento de Veículos (PRV) **3**

2.1 Considerações e característica do problema 3

2.2 Variações do modelo clássico 6

2.2.1 Roteamento com janela de tempo 6

2.2.2 Roteamento com múltiplos depósitos 7

2.2.3 Roteamento com frota heterogênea 7

2.2.4 Roteamento de veículos capacitados 7

2.2.5 Roteamento com entrega e coleta 7

2.2.6 Roteamento com entrega prioritária 7

2.3 Definição formal do PRV 8

2.4 Variação do PRV abordado 9

3 Abordagens Heurísticas Proposta para o Problema **11**

3.1 Definições básicas sobre abordagens heurísticas 11

3.2 Heurísticas construtivas propostas para o PRV abordado 14

3.3 Heurísticas de busca local propostas 18

3.3.1 Busca local intra-rota 19

3.3.2 Busca local inter-rota 20

3.4 Abordagem GRASP para o PRV 21

4 Resultados Computacionais **28**

5 Conclusão e Trabalhos Futuros **37**

Referências **39**

Lista de Figuras

2.1 Entrada típica do PRV. 4

2.2 Uma possível solução para o PRV. 4

3.1 Ótimo local e global [Cordenonsi, 2008]. 13

3.2 Desvantagem de procedimentos puramente heurísticos [Cordenonsi, 2008]. . 14

3.3 Descrição de movimento 2-opt. 19

3.4 Funcionamento de busca local inter-rota. 21

3.5 Princípio do funcionamento do procedimento de viabilização. 24

4.1 Contribuição da Busca Local na instância I_1 com $\alpha = 0,4$ 32

4.2 Contribuição da Busca Local na instância I_9 com $\alpha = 0,4$ 33

4.3 Contribuição da busca local com a instância I_{19} e $\alpha = 0,4$ 35

4.4 Redução do número de veículos para a instância I_{19} com $\alpha = 0,4$ 36

4.5 Redução da distância percorrida para a instância I_{19} com $\alpha = 0,4$ 36

Lista de Tabelas

3.1 Análise das funções de avaliação 17

4.1 Combinações de heurísticas para composição das versões de GRASP 28

4.2 Primeiro conjunto de instâncias 29

4.3 Segundo conjunto de instâncias 29

4.4 Resumo dos resultados das heurísticas para as instâncias com 100 clientes. 31

4.5 Resumo dos resultados das heurísticas para as instâncias com 250 clientes. 31

4.6 Resumo dos resultados das heurísticas para as instâncias com 500 clientes. 32

4.7 Resumo dos resultados das heurísticas para o experimento de minimização da frota. 34

Capítulo 1

Introdução

Os elevados custos que demanda o transporte do produto final ou de matéria prima, bem como as despesas no deslocamento para prestações de serviços têm recebido grande atenção das equipes de planejamento e de produção das empresas nos mais diversos segmentos de mercado.

Os custos com deslocamento da frota correspondem em média a 20% do gasto total das empresas [Cruz and Oliveira, 2008] e cerca de 10 a 15% do valor final dos produtos tem influência direta do custo com o transporte [Fisher et al., 1997].

Definir manualmente um bom roteiro para atendimento de um conjunto de clientes que demandam serviços pode levar um tempo excessivo com análise de mapas e documentos. Porém, a utilização de técnicas adequadas de Pesquisa Operacional pode tornar mais confiável e eficiente esta tarefa.

Determinar as sequências de visitas que permitem atender um conjunto de clientes dispersos em uma região com o menor custo total de deslocamento é denotado na literatura como o Problema de Roteamento de Veículos (PRV) [Toth and Vigo, 2002]. O PRV possui um grande número de aplicações práticas, devido principalmente ao grande número de variações do problema, variações estas definidas por diferentes restrições e diferentes objetivos apresentados em cada caso. Dentre as diversas aplicações, pode-se citar a entrega de correspondência, o transporte escolar, o recolhimento de lixo, o patrulhamento policial e de segurança, distribuição de jornais, planejamento de transporte de carros por caminhões, entre outros [Goldbarg and Luna, 2005].

O PRV é um dos problemas mais complexos da área de otimização combinatória [Goldbarg and Luna, 2005]. Além do fato de que os algoritmos exatos conhecidos dificilmente conseguem solucionar problemas envolvendo mais do que 135 clientes/consumidores

[Fukasawa et al., 2006, Baldacci et al., 2007], a dificuldade em encontrar soluções ótimas para instâncias do mundo real traz o desafio da busca de novos métodos heurísticos, de forma que eles possam gerar soluções de boa qualidade em um tempo aceitável para a aplicação.

A finalidade deste trabalho é apresentar um algoritmo para uma variação do PRV, onde os clientes que devem ser atendidos possuem prioridades distintas. Três heurísticas construtivas e duas heurísticas de busca local são propostas. A partir destas heurísticas, uma abordagem baseada na metaheurística GRASP é apresentada para o problema.

Este trabalho está organizado como segue: no Capítulo 2 é apresentado o Problema de Roteamento de Veículos clássico, a sua formulação matemática, as principais variações deste problema encontrados na literatura e a descrição da variação do PRV abordado neste trabalho. As abordagens heurísticas construtivas e de busca local são apresentadas no Capítulo 3, além da metaheurística GRASP. Os resultados computacionais são mostrados no Capítulo 4. Por fim, as conclusões e sugestões de trabalhos futuros são apresentadas no Capítulo 5.

Capítulo 2

O Problema de Roteamento de Veículos (PRV)

O Problema de Roteamento de Veículos (PRV) foi primeiramente formulado no final da década de 50 por [Dantzig and Ramser, 1959] com a intenção de encontrar rotas ótimas para uma frota de caminhões que transportaria gasolina a partir de um terminal de granel até um conjunto de estações de venda de combustíveis.

2.1 Considerações e característica do problema

Atualmente, o PRV é uma denominação genérica dada a toda uma classe de problemas em que um conjunto de rotas para uma frota de veículos baseada em um ou mais depósitos (ou pátios) deve ser determinado para uma série de cidades ou clientes consumidores geograficamente dispersos [Goldbarg and Luna, 2005]. O objetivo é atender o conjunto de consumidores com um custo mínimo, através da escolha de rotas que iniciem e terminem no pátio, sendo que cada cliente deve ser atendido por um único veículo. O custo associado a cada rota pode variar conforme a aplicação.

Na figura 2.1, pode-se ver uma típica entrada para o PRV, onde tem-se o pátio onde estão todos os veículos da frota, e os nós dispersos, que representam aqueles clientes que podem ser atendidos, ou ainda, cidades que devem ser visitadas por algum veículo da frota.

Após a designação de rotas, pode-se ter um cenário como o apresentado na figura 2.2, de forma que cada rota corresponde ao atendimento prestado por um veículo da frota. No exemplo apresentado, não se tem a restrição que obriga que todo cliente seja atendido, mas pode-se observar que cada cliente atendido está presente em uma única rota.

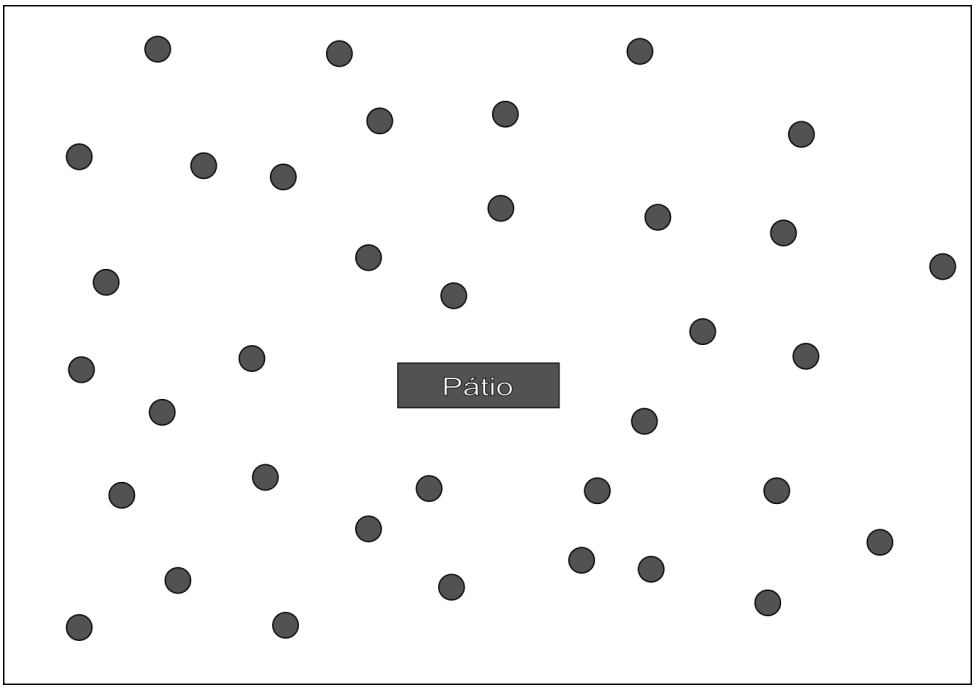

Figura 2.1: Entrada típica do PRV.

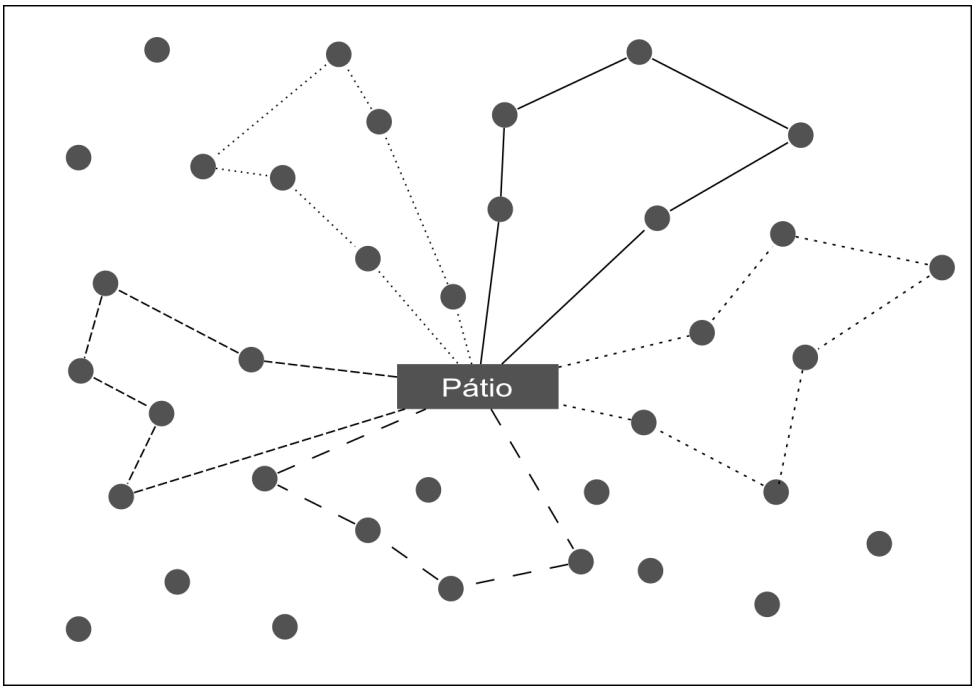

Figura 2.2: Uma possível solução para o PRV.

No mundo real, geralmente o PRV aparece definido sob diversas restrições e muitas variáveis. Para melhorar o entendimento, em [Golden et al., 1981] os autores propuseram uma taxonomia que permite caracterizar cada variação do problema, facilitando a compreensão dos aspectos mais importantes do PRV para a escolha da estratégia mais adequada para solucioná-lo. Segundo os autores, na fase de modelagem e análise do problema deve-se procurar identificar a existência de critérios como, por exemplo:

- Tempo para servir um determinado cliente:

  atendimento com tempo especificado e prefixado;

  atendimento dentro de uma janela de tempo (2.2.1).

- Número de depósitos existentes:

  único depósito;

  vários depósitos (2.2.2).

- Tamanho da frota disponível

  único veículo;

  diversos veículos.

- Tipo de frota disponível

  frota homogênea;

  frota heterogênea (2.2.3).

- Tempo de roteamento

  mesmo para todos os veículos;

  tempos diversos;

  sem restrições de tempo.

- Restrições na capacidade de veículos:

  todos os veículos têm a mesma capacidade (2.2.4);

  veículos com diferentes capacidades.

- Tipo de operação:

  atendimento;

  coleta;

entrega;

coleta e entrega (2.2.5).

- objetivo do problema:

  minimizar custos fixos;

  minimizar custos de operação na rota.

Na literatura, pode-se encontrar diversas abordagens utilizadas para solução do PRV, como Algoritmos Genéticos [Alba E., 2004, Bräysy and Gendreau, 2001] e Colônias de Formigas [B. Bullnheimer and Strauss, 1997, Reimann et al., 2003]. Uma abordagem baseada em *Simulated Annealing* é apresentada em [P. Czarnas and Gocyla, 2004], enquanto em [Montané, 2006] é apresentado um algoritmo de Busca Tabu Adaptativa. Abordagens baseadas no algoritmo *Greedy Randomized Adaptive Search Procedures* (GRASP) podem ser encontradas em [Festa and Resende, 2002].

2.2 Variações do modelo clássico

Tendo em vista o fato de que, em situações práticas, os problemas variam entre si pelos aspectos relacionados ao tipo de operação, tempo de roteamento, tipo de frota, dentre outros critérios citados, na literatura encontra-se um grande número de variações para o PRV clássico. Nesta seção são apresentadas as principais variações do problema.

2.2.1 Roteamento com janela de tempo

A variante chamada de Problema de Roteamento de Veículos com Janela de Tempo (PRVJT) consiste em um PRV clássico com uma restrição adicional de que uma janela de tempo está associada a cada cliente, definindo um intervalo de tempo em que o cliente pode ser atendido [Toth and Vigo, 2002].

Este problema pode ser encontrado, por exemplo, em situações em que, por razões de logística ou de legislação, um determinado tipo de veículo ou carga somente pode ser usado em determinados horários. Em algumas redes de supermercados, por exemplo, o recebimento de mercadorias se dá em horários previamente definidos conforme a carga. Isto evita o congestionamento de veículos com diferentes tipos de carga nas imediações do depósito, bem como permite uma melhor programação das diferentes equipes de trabalho que farão o recebimento das mercadorias.

2.2.2 Roteamento com múltiplos depósitos

Em situações reais, uma companhia pode ter vários depósitos a partir dos quais seus clientes podem ser atendidos. O Problema de Roteamento de Veículos com Múltiplos Depósitos (PRVMD) requer uma associação de clientes aos depósitos. Cada veículo que sai de um depósito atende aos clientes associados ao seu depósito. As viaturas podem sair e retornar ao mesmo depósito ou transitar entre os depósitos existentes.

2.2.3 Roteamento com frota heterogênea

O Roteamento com Frota Heterogênea (PRFH), também conhecido como Roteamento de Veículos Mistos [Renaud and Boctor, 2002], objetiva determinar, simultaneamente, a composição de uma frota heterogênea de veículos sediada em um depósito e o conjunto de rotas de atendimento.

2.2.4 Roteamento de veículos capacitados

No Problema de Roteamento de Veículos Capacitados (PRVC), os clientes correspondem a pontos de entrega e suas demandas são determinísticas, ou seja, conhecidas previamente. A frota de veículos possui capacidade uniforme e o objetivo é minimizar o custo total para servir a todos clientes [Toth and Vigo, 2002]. Atualmente, a maioria dos trabalhos científicos que envolvem o PRV se concentra no caso capacitado [Goldbarg and Luna, 2005].

2.2.5 Roteamento com entrega e coleta

O Problema de Roteamento com Entrega e Coleta (PREC) consiste no fato de que a cada cliente está associada uma quantidade de produtos que deve ser recolhida e outra que deve ser entregue pelo veículo transportador. Assume-se que em cada nó cliente, a entrega é realizada antes do recolhimento de um produto [Toth and Vigo, 2002].

2.2.6 Roteamento com entrega prioritária

O Problema de Roteamento com Entrega Prioritária é uma extensão do PRVC, pelo qual, o conjunto de clientes deve ser dividido em dois grupos. O primeiro grupo corresponde ao conjunto de clientes que demandam a entrega de um produto, e o segundo grupo corresponde ao conjunto de clientes que possuem mercadorias que devem ser recolhidas.

Há uma restrição de precedência entre esses grupos, onde todas as entregas devem ser realizadas antes de efetuar qualquer recolhimento [Toth and Vigo, 2002].

2.3 Definição formal do PRV

Dado um grafo direcionado ponderado $G = (V, A)$, onde $V = \{0, 1, \ldots, n\}$ é o conjunto de vértices (clientes), com 0 indicando o vértice referente ao pátio, A é o conjunto de arestas (i, j). Considere c_{ij} um custo não negativo associado a cada aresta $(i, j) \in A$. Considere ainda um conjunto de $K = \{1, \ldots m\}$ composto pelos m veículos idênticos da frota.

Sejam as variáveis binárias definidas por:

$$x_{ijk} = \begin{cases} 1, & \text{se o veículo } k \text{ visita o cliente } j \text{ imediatamente após o cliente } i. \\ 0, & \text{caso contrário.} \end{cases}$$

$$y_{ik} = \begin{cases} 1, & \text{se cliente } i \text{ é visitado pelo veículo } k. \\ 0, & \text{caso contrário.} \end{cases}$$

A formulação matemática apresentada por Fisher e Jaikumar em [L.Fisher and R.Jaikumar, 1981] para o Problema de Roteamento de Veículos é apresentada a seguir.

Função Objetivo:

$$Minimizar = \sum_{(i,j)\ \in\ A} \left(c_{ij} \sum_{k=1}^{m} x_{ijk} \right) \tag{2.1}$$

Sujeito a:

$$\sum_{k=1}^{m} y_{ik} = 1, \qquad i = 1, \ldots, n \tag{2.2}$$

$$\sum_{k=1}^{m} y_{ik} = m, \qquad i = 0 \tag{2.3}$$

$$\sum_{j \in V \setminus \{0\}} x_{ijk} = \sum_{j \in V \setminus \{0\}} x_{jik} = y_{ik}, \qquad \forall i \in V,\ k = 1, \ldots, m \tag{2.4}$$

$$\sum_{i,j \in S} x_{ijk} \leq |S| - 1, \qquad \forall S \subseteq \{1, ..., n\},\ k = 1, \ldots, m \tag{2.5}$$

$$y_{ik} \in \{0, 1\}, \qquad \forall i = 1, \ldots, n,\ k = 1, \ldots, m \tag{2.6}$$

$$x_{ijk} \in \{0, 1\}, \qquad \forall i, j = 1, \ldots, n,\ k = 1, \ldots, m \tag{2.7}$$

As restrições (2.2) asseguram que todo cliente é atendido, sendo cada cliente visitado por um único veículo. As restrições (2.3) asseguram que cada veículo da frota retorna ao pátio de veículos, enquanto as restrições (2.4) garantem que se um cliente é visitado por um veículo k, este veículo sairá deste cliente. As restrições (2.5) são as tradicionais restrições de eliminação de sub-rotas. Por este conjunto de restrições, para qualquer subconjunto de clientes, o número total de arestas que chegam aos clientes deste subconjunto é dado por, no máximo, o número de elementos do subconjunto menos uma unidade, ou seja, nenhum cliente é visitado mais que uma vez, impedindo que a rota seja fechada entre os clientes de S.

2.4 Variação do PRV abordado

O problema abordado neste trabalho originou-se de uma aplicação real, na qual uma empresa distribuidora de energia elétrica, objetivando melhorar a prestação de serviços ao cliente, requisitava um sistema que fosse capaz de gerar rotas, dado um conjunto de consumidores que demandavam atendimento, levando em consideração alguns aspectos particulares.

Em um primeiro instante, os clientes demandavam serviços distintos, para os quais, apenas viaturas com a capacidade de realizar aquela operação solicitada poderia ser designada para atender tal requisição. Como o cenário de demanda é muito grande, muitos clientes não conseguem atendimento no mesmo dia em que o serviço foi requisitado, devendo haver uma prioridade no atendimento de requisições mais antigas ou emergenciais. Sendo preferível, por exemplo, dados dois clientes que são equidistantes da posição atual de uma viatura, atender o cliente de maior prioridade.

A frota de veículos considerada é heterogênea, de forma que existem três tipos distintos de viaturas. O tipo de viatura está associado ao tipo de serviço que o cliente demanda, onde cada tipo de atendimento requer um tempo previsto que este atendimento demandará. Pode ocorrer, portanto, que um mesmo cliente deva ser atendido por mais de uma viatura, desde que os tipos de veículos sejam diferentes.

A cada tipo de veículo está associado um fator de deslocamento, que indicará a velocidade média esperada para aquele tipo de veículo. O tempo total de duração da rota deve considerar o tempo de deslocamento até os clientes, o tempo de atendimento e o tempo de retorno ao pátio.

Além disso, no cenário real descrito, há uma restrição quanto ao tempo de duração da

rota de qualquer viatura. Este tempo está associado à jornada de trabalho das equipes que prestam o atendimento por cada viatura.

Desta forma, na variação do PRV tratada neste trabalho, cada rota deverá iniciar e terminar no pátio. O objetivo é atender o número máximo de clientes, percorrendo a menor distância possível, dando preferência a clientes de maior prioridade.

Capítulo 3

Abordagens Heurísticas Proposta para o Problema

O grande desafio da Otimização Combinatória é produzir soluções com o menor custo possível em um tempo aceitável para a aplicação. Uma solução é dita ótima quando seu custo é o menor possível para resolver dado problema. Apesar de algoritmos exatos garantirem essa solução ótima (menor custo), este tipo de abordagem geralmente não garante que a solução seja alcançada em um tempo polinomial.

3.1 Definições básicas sobre abordagens heurísticas

Algoritmos heurísticos surgem como uma proposta para que soluções próximas da otimalidade (sub-ótimas) sejam encontradas em um tempo polinomial. Entretanto, heurísticas geralmente apresentam soluções para as quais não se pode afirmar que sejam ótimos globais [Cordenonsi, 2008].

As heurísticas podem ser divididas em heurísticas construtivas, heurísticas de melhoramento ou busca local e ainda metaheurísticas [Cordenonsi, 2008]. Nesta seção é apresentada uma breve descrição de cada um destes tipos de abordagens. Os algoritmos propostos serão detalhados na seção 3.2.

- Heurísticas construtivas

Um algoritmo construtivo consiste em um método iterativo capaz de construir uma solução a partir de um conjunto de elementos candidatos, que são selecionados para integrar a solução conforme uma função gulosa de avaliação. Em problemas de minimização,

a função de avaliação reflete a contribuição de cada elemento candidato no aumento do custo final da solução. A cada iteração, na escolha gulosa do próximo elemento a integrar a solução parcial, o elemento candidato escolhido é aquele que implica em menor aumento no custo da solução atual. O algoritmo pára quando uma solução viável é obtida ou quando é impossível a inclusão de qualquer elemento sem tornar a solução atual inviável.

Para o PRV, um algoritmo construtivo deve proceder de forma que cada rota comece com o elemento que indica o depósito e, a cada iteração, um novo cliente é inserido na solução, até que se obtenha uma rota completa que atenda o conjunto de restrições do problema. Os algoritmos construtivos não guardam informações sobre as últimas iterações, se concentrando apenas no próximo passo, que é a escolha de um novo elemento. A seleção de qual elemento será inserido é feita através de uma função gulosa, que varia de acordo com o objetivo que se procura alcançar.

- Heurísticas de busca local

Entende-se por espaço de busca ou espaço de soluções o conjunto de soluções viáveis de um problema para uma dada entrada de dados. Uma heurística de busca local tem como entrada uma solução inicial s, dita solução de partida ou incumbente, e procura obter soluções de melhor qualidade numa região do espaço de soluções, que é denotada por vizinhança da solução ou $\aleph(s)$, através de movimentos elementares na estrutura da solução de partida [Cordenonsi, 2008].

O espaço de soluções definido por $\aleph(s)$ depende da estrutura de vizinhança estabelecida, ou seja, depende do tipo de movimento a ser aplicado à solução s. Geralmente, um movimento elementar em uma heurística de busca local está associado à substituição, inclusão ou remoção de elementos da solução atual, ou ainda à permutações na ordem em que os elementos se encontram em s.

Para o PRV, uma vizinhança de uma solução s se refere às rotas que se encontram próximas, no espaço de busca do problema, que podem ser obtidas através de um movimento aplicado em s. Um movimento pode ser de inclusão ou de exclusão de arcos ou vértices, por exemplo.

Um algoritmo de busca local pode ter diferentes critérios de parada, como um número pré-fixado de iterações sem melhora, podendo parar apenas quando nenhuma solução de qualidade superior na vizinhança de s é encontrada, o que é considerado um ótimo local. Denota-se como ótimo local o menor valor de uma função em um intervalo considerado, neste caso, a vizinhança $\aleph(s)$. Eventualmente, o ótimo local pode ser o ótimo global, que

Figura 3.1: Ótimo local e global [Cordenonsi, 2008].

é a melhor solução para o problema, porém, não há garantias em relação a tal fato quando se emprega abordagens heurísticas ou qualquer abordagem não exaustiva.

A figura 3.1 apresenta um gráfico que exemplifica o funcionamento de uma heurística de busca local. A busca começa com uma solução inicial, possivelmente obtida a partir de um algoritmo construtivo. Após uma série de movimentos na solução incumbente, o algoritmo define o ponto de ótimo local como a melhor solução encontrada para o problema, já que qualquer solução dentro a vizinhança da solução atual apresenta qualidade inferior. Percebe-se neste exemplo que a melhor solução do problema (ótimo global) não foi encontrada.

- Metaheurísticas

Metaheurísticas são procedimentos que guiam outras heurísticas [Cordenonsi, 2008] através do espaço de soluções, buscando identificar boas características das soluções encontradas e explorar novas regiões promissoras. O objetivo é gerar procedimentos de busca que evitem uma parada prematura em ótimos locais, proporcionando soluções de melhor qualidade.

Na figura 3.2 pode-se ver um problema da aplicação de procedimentos heurísticos puramente construtivas e de busca local. Como pode ser observado na figura, dependendo

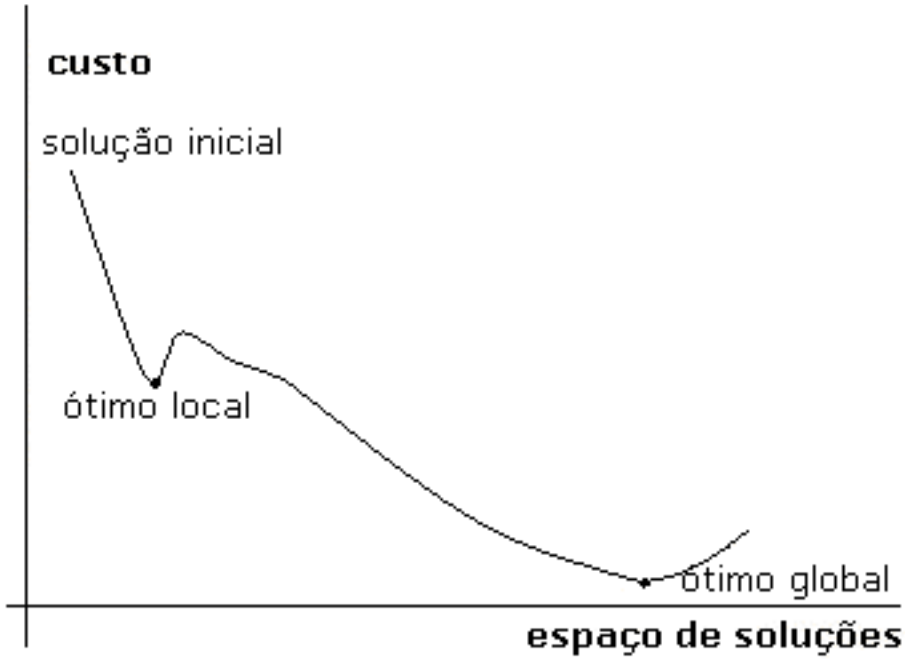

Figura 3.2: Desvantagem de procedimentos puramente heurísticos [Cordenonsi, 2008].

da estrutura de vizinhança utilizada, qualquer solução próxima ao ponto destacado como ótimo local apresenta custo maior que a solução atual.

O uso de metaheurísticas pode guiar as heurísticas de busca local no sentido de escapar de ótimos locais na tentativa de encontrar o ótimo global. Como não há garantias de que o ótimo global seja alcançado, os critérios de parada adotados pelos algoritmos devem ser escolhidos com cuidado no intuito de gerar soluções viáveis e de boa qualidade em um tempo computacional aceitável para a aplicação.

Nas próximas seções são descritas as heurísticas construtivas e de busca local, além do algoritmo baseado na metaheurística GRASP desenvolvido para resolver o problema proposto neste trabalho.

3.2 Heurísticas construtivas propostas para o PRV abordado

O método construtivo proposto é responsável por gerar um conjunto de rotas iniciais viáveis quanto as restrições do problema. Cada rota é construída de forma iterativa, inserindo um cliente por vez na rota até que não se possa inserir um novo cliente sem que a solução se torne inviável.

A escolha de qual cliente será inserido é feita em de duas etapas: primeiramente, uma Lista de Candidatos (LC) é criada, onde os clientes que não inviabilizam a solução são adicionados e ordenados segundo uma função gulosa $f : V \rightarrow R^+$ de avaliação de candidatos.

Uma segunda lista, denotada Lista Restrita de Candidatos (LRC), é construída a partir dos elementos de LC de forma a apresentar um subconjunto de elementos que possuem os melhores valores para a função de avaliação f. Seja max e min, respectivamente, o maior e menor valor obtido pela aplicação da função f a todos os elementos candidatos da LC. Seja ainda $\alpha \in [0, 1]$ um fator que define o percentual de elementos candidatos a serem considerados na LRC e S o conjunto de elementos já inseridos na solução. A cada iteração, os elementos que comporão a LRC são aqueles para os quais o valor da função f atende o critério definido pela Equação 3.1.

$$f(i) \leq \alpha \times max + (1 - \alpha) \times min \quad \forall i \in V \setminus S. \tag{3.1}$$

Como pode ser observado na Equação 3.1, quando $\alpha = 0$, apenas o elemento de menor custo associado à função f é inserido na LRC, o que configura a abordagem totalmente gulosa. Já quando $\alpha = 1$, tem-se a abordagem totalmente aleatória.

Como apresentado com Capítulo 2, o problema apresentado neste trabalho requer a definição de rotas que atendem critérios bem definidos de custo e ainda um critério relacionado à qualidade de serviço prestado pela concessionária de energia que solicitou a aplicação. Diante destas peculiaridades do problema, três funções gulosas de avaliação foram definidas de forma a atender diferentes requisitos.

Seja a solução s definida pelo conjunto de arcos inseridos em alguma rota e S o conjunto de nós clientes que já foram atendidos por alguma rota. A primeira função (f_1) leva em consideração apenas a distância entre o último cliente inserido i e o cliente candidato j, inserindo o cliente mais próximo, como pode ser observado na Equação 3.2.

$$f_1(j) = d_{ij} \;\; \forall j \in V \setminus S, \; i \in S \text{ e } \nexists \; (i, l) \in s \; \forall \; l \in V. \tag{3.2}$$

Pela Equação 3.2, pode-se observar que o conjunto de elementos candidatos em cada iteração deve ser recalculado com base no cliente inserido na iteração anterior. Além disso, embora a equação não indique, na implementação da heurística definida por esta função, são considerados apenas os clientes j para os quais é possível estabelecer a visita

e o atendimento sem que a restrição de tempo da jornada de trabalho da equipe seja violada.

A segunda função de avaliação, f_2, além de considerar a distância entre o último cliente i inserido e os clientes j candidatos, procura-se atender o requisito de prioridade de atendimento do cliente candidato (p_j). Desta forma, por f_2, espera-se que os clientes de maior prioridade tenham maior preferência de inserção na rota. A função f_2 é definida pela Equação 3.3. Pode-se observar que quanto maior a prioridade e quanto menor a distância entre os clientes candidatos e o último cliente inserido, menor o valor da função e, consequentemente, melhor avaliado é o cliente.

$$f_2(j) = \frac{d_{ij}}{p_j} \quad \forall j \in V \setminus S,\ i \in S \text{ e } \nexists\ (i,l) \in s\ \forall\ l \in V. \tag{3.3}$$

Antes de definir a terceira função de avaliação proposta, deve-se definir o conceito de cliente vizinho. Para tanto, seja o conjunto de nós clientes V. Como uma etapa de pré-processamento, utilizou-se um algoritmo de clusterização[1] que definiu diferentes grupos de clientes conforme a proximidade geográfica. A clusterização dos pontos geográficos onde os clientes estão localizados é realizada em uma etapa de pré-processamento, não sendo detalhada neste trabalho.

A última função proposta, f_3 leva-se em consideração, além da distância do cliente candidato e o último cliente inserido, o número de vizinhos do candidato j, denotado por $nV(j)$. O número de vizinhos é dado pela quantidade de clientes que pertencem ao mesmo *cluster* de j.

$$f_3(j) = \frac{d_{ij}}{nV(c_j)} \quad \forall j \in V \setminus S,\ i \in S \text{ e } \nexists\ (i,l) \in s\ \forall\ l \in V. \tag{3.4}$$

O principal objetivo ao se propor a função f_3 foi permitir que o algoritmo procure aumentar o número de clientes atendidos em uma mesma rota, possibilitando a redução do número de viaturas utilizadas.

Para facilitar o entendimento do impacto de cada uma das funções apresentadas na avaliação dos clientes candidatos, a Tabela 3.1 exemplifica a diferença entre as funções utilizadas. Para tanto, considere a solução parcial s composta apenas pelo nó zero (pátio ou depósito) e o primeiro cliente atendido i. Considere ainda as distâncias apresentadas

[1]Clusterização é o processo de tomar um conjunto de objetos e separá-los em grupos cujos objetos apresentem características similares entre si e que apresentam dissimilaridades em relação aos objetos de outros grupos [Soares, 2004].

Dados de cada cliente				**Funções de Avaliação**		
Candidato j	d_{ij}	p_j	$nV(c_j)$	f_1	f_2	f_3
c_1	50,00	2	4	50,00	25,00	12,50
c_2	48,00	4	3	48,00	12,00	16,00
c_3	30,00	1	2	30,00	30,00	15,00
c_4	80,00	5	2	80,00	16,00	40,00
c_5	40,00	2	6	40,00	20,00	6,67

Tabela 3.1: Análise das funções de avaliação

na coluna d_{ij} e os valores da coluna $nV(c_j)$ definidos para o número de nós pertencentes a cada cluster dos elementos c_j com $j \in \{1, \ldots, 5\}$ avaliados.

De acordo com os dados apresentados, a Lista de Candidatos definida pela função f_1 é dada por $\{c_3, c_5, c_2, c_1, c_4\}$, enquanto que a função f_2 define uma $LC = \{c_2, c_4, c_5, c_1, c_3\}$. A função f_3, que leva em consideração o número de vizinhos, geraria $LC = \{c_5, c_1, c_3, c_2, c_4\}$. Como pode ser visto, os valores apresentados para cada função de avaliação levam a diferentes regiões do espaço de soluções.

Dados V o conjunto de nós clientes, e K o conjunto de veículos da frota, no Algoritmo 1 é apresentado o pseudocódigo da heurísticas de construção. Como parâmetros, são passados o tipo de função de avaliação, f_a, e o valor do fator α que define o tamanho da LRC.

Na linha 1, o conjunto de rotas que define a solução do problema é inicializada como vazio, enquanto na linha 2, todas as rotas para o conjunto de veículos da frota são inicializadas com o nó zero, que indica o pátio. Na linha 3, o conjunto S de clientes que já foram atendidos também é inicializado com o nó pátio.

O laço compreendido entre as linhas 4 e 18 define que, para cada veículo disponível no pátio, uma rota será construída. No início da construção de cada rota, a variável que indica qual o último cliente atendido é inicializado na linha 5 como zero, já que inicialmente apenas o índice zero, referente ao pátio, encontra-se inserido em cada rota. O tempo da rota é definido como zero, na linha 7 é criada a lista de candidatos conforme o conjunto de nós ainda não atendidos, a função gulosa de avaliação e o tempo da rota, que inicialmente é zero.

No laço compreendido entre as linhas 8 e 17, a lista restrita de candidatos é criada de acordo com a lista de candidatos e o valor do parâmetro α na linha 9. Os clientes são inseridos na rota do veículo k corrente, o tempo transcorrido da rota é atualizado na linha 12 para que, na linha 15, a lista de candidatos seja montada levando em consideração

somente os clientes que, caso inseridos, não ultrapassem a jornada de trabalho. Caso não exista nenhum cliente que possa ser inserido sem inviabilizar a solução, a LC terá zero clientes e a construção da rota de um novo veículo k é iniciada.

Algoritmo 1 Construtivo(f_a, α)

```
1: Solucao ← ∅;
2: Rota_k ← 0 ∀k ∈ {1, ..., |K|};
3: S ← {0};
4: para k = 1 até |K| faça
5:    ultimo ← 0;
6:    tempo_k ← 0;
7:    LC ← montaListaCandidatos(V \ S, f_a, ultimo, tempo_k);
8:    enquanto (|LC| > 0) faça
9:       LRC ← montaListaRestrita(LC, α);
10:      atribua a c um elemento da LRC, selecionado aleatoramente;
11:      Rota_k ← c;
12:      tempo_k ← tempo_Gasto_Rota(k);
13:      ultimo ← c;
14:      S ← S ∪ {c};
15:      LC ← montaListaCandidatos(V \ S, f_a, ultimo, tempo_k);
16:   fim enquanto
17:   Solucao ← Solucao ∪ Rota_k;
18: fim para
19: retorne Solucao;
```

Para uma melhor compreensão, no restante deste trabalho a heurística construtiva definida pela uma função f_1 é denotada $C1$, aquela associada à função f_2 é referenciada como $C2$ e a heurística construtiva baseada na função f_3 é chamada $C3$.

3.3 Heurísticas de busca local propostas

Como descrito na seção anterior, uma heurísticas de Busca Local visa intensificar a qualidade de uma solução inicial através de movimentos elementares na estrutura desta solução, na tentativa de encontrar soluções de melhor qualidade que a solução de partida.

Duas heurísticas de busca local para a variação do PRV tratada neste trabalho foram implementadas. A estrutura de vizinhança da primeira busca local é definida através de movimentos intra-rota, ou seja, somente movimentos com elementos da mesma rota. A segunda busca local é baseada em movimento inter-rota, de forma que a vizinhança é estabelecida através de movimentos com elementos de rotas diferentes.

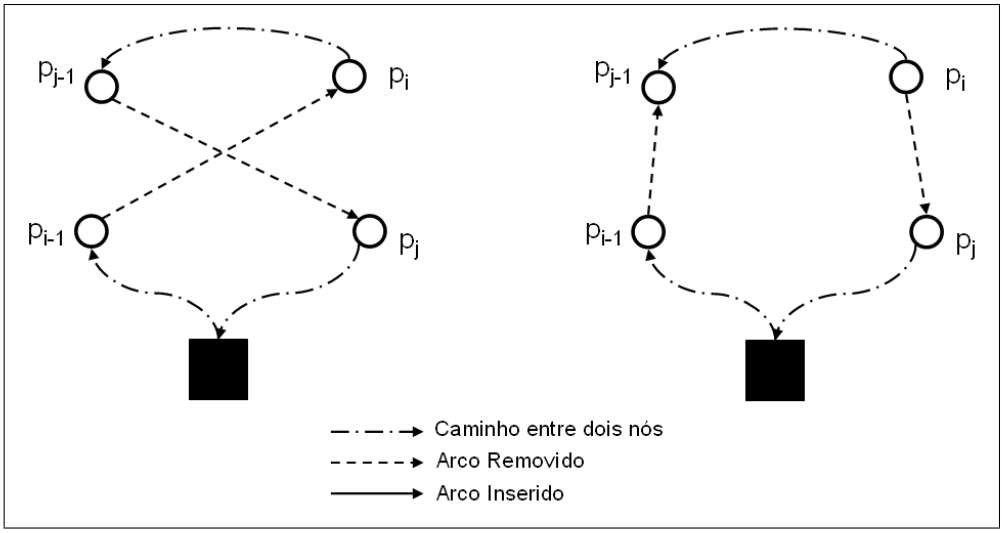

Figura 3.3: Descrição de movimento 2-opt.

3.3.1 Busca local intra-rota

A primeira busca local implementada neste trabalho corresponde a heurística conhecida como *k-optimal* [Lin and Kernighan, 1973], pela qual se removem *k-arcos* de uma rota e os substitui por outros *k-arcos* de modo a formar uma nova rota viável. Com este movimento, é possível que se consiga reduzir o custo total de uma rota. Assim, os movimentos *k-opt* somente são realizados caso o custo da rota seja reduzido. Uma rota é dita *k-ótima* quando não é mais possível efetuar trocas que melhorem o custo da rota.

A Figura (3.3) exemplifica o funcionamento da busca local *k-opt* com $k = 2$, onde os arcos (p_{i-1}, p_i) e (p_{j-1}, p_j) são removidos e substituídos pelos arcos (p_{i-1}, p_{j-1}) e (p_i, p_j). O que se espera é que se consiga rearranjar a rota visando diminuir o tempo total de duração da mesma, de forma que novos clientes possam ser inseridos nesta rota.

À medida em que k cresce, o método *k-optimal* aproxima-se da enumeração total [Goldbarg and Luna, 2005]. Por isso, foi adotado neste trabalho o uso do procedimento *k-optimal* com $k = 2$, onde o tamanho da vizinhança é de $n(n-1)/2$.

O Algoritmo 2 descreve o funcionamento da heurística de busca local do tipo *2-optimal*. Inicialmente, todas as combinações possíveis de arcos presentes na rota incumbente é separada em um conjunto, denotado por C (linha 1). A partir desse conjunto, um arco deve ser escolhido (linha 3) e removido da rota atual (linha 4), gerando uma rota auxiliar $Rota'$ temporariamente desconexa. Na linha 5, o conjunto de todos os arcos que podem reconectar a rota a partir de nós que estão na solução é construído. O procedimento $Melhor\_Reconexao()$ (linha 6) identifica a rota de menor custo dentre todas as rotas

possíveis de se obter a partir da reconexão de $Rota'$.

O custo da nova rota é comparado com o custo da rota inicial (linha 7), caso o custo seja menor, atualiza-se a rota incumbente e recomeça-se o processo (linha 9). Caso o custo da última rota obtida seja maior que o da rota inicial, remove-se o arco do conjunto C.

Algoritmo 2 Busca Local 2-opt($Rota$)

```
1:  C ← Conjunto_Arcos_Possiveis(Rota);
2:  enquanto (|C| > 0) faça
3:     arco ← arco ∈ C;
4:     Rota' ← Remove(arco, Rota);
5:     C' ← Conjunto_Arcos_Possiveis(Rota');
6:     Rota' ← Melhor_Reconexao(arco, C');
7:     se (custo(Rota') < custo(Rota)) então
8:        Rota ← Rota';
9:        C ← Conjunto_Arcos_Possiveis(Rota);
10:    senão
11:       Retira_Arco(arco, C);
12:    fim se
13: fim enquanto
```

Para efeitos de comparação, esta busca local é denotada por BL1 ao logo deste trabalho.

3.3.2 Busca local inter-rota

A segunda busca local implementada neste trabalho, aqui denotada BL2, procura realizar uma realocação dos clientes que já estão inseridos em alguma rota. Basicamente, o movimento realizado é o de remoção de um cliente de uma rota, e inserção do mesmo em outra. Caso o custo total, considerando-se a duas rotas, diminua, a realocação das rotas é efetivada.

A Figura 3.4 mostra o funcionamento da busca local por realocação. A intenção é que, com as modificações efetuadas nas rotas, possa-se inserir novos clientes nas rotas para as quais foi possível remover clientes. Como pode ser visto, o cliente p_i, que era atendido pela rota p após o cliente p_{i-1}, passou a ser atendido pela rota q após o cliente q_{j-1}.

O Algoritmo 3 mostra o pseudocódigo da busca BL2. Primeiramente, seleciona-se uma rota presente na solução (linha 1). Em seguida, iterativamente, cada cliente da rota incumbente é selecionado(linha 2), removido de sua rota e inserido em outra rota (linhas 4 e 5).

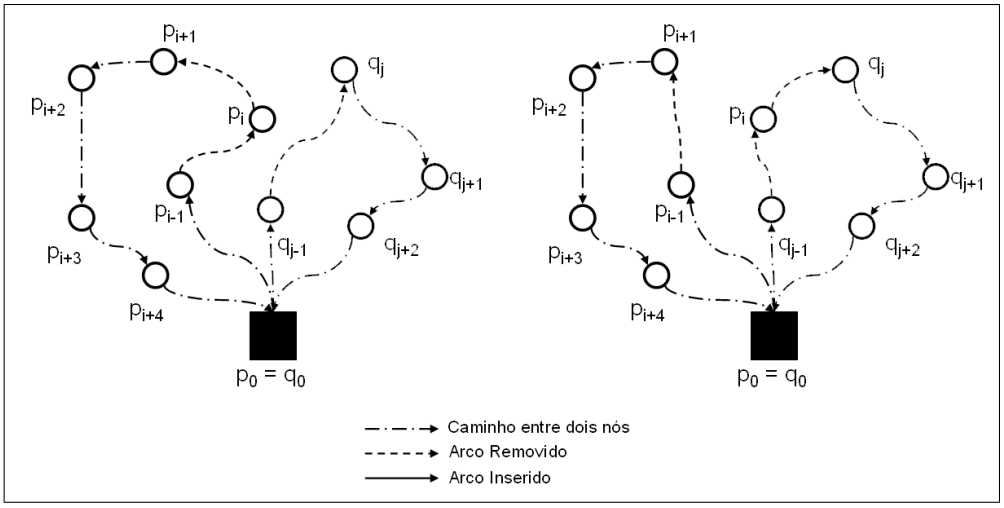

Figura 3.4: Funcionamento de busca local inter-rota.

Caso um movimento implique na redução do somatório dos custos das rotas envolvidas na troca (linha 6), a realocação do cliente é efetivada e o processo é reiniciado considerando-se o novo conjunto de rotas.

Algoritmo 3 Realocação($Solucao$)

1: **para** *cada* $Rota_i \in Solucao$ **faça**
2: **para** *cada cliente* $\in Rota_i$ **faça**
3: **para** *cada* $Rota_j \neq Rota_i$ **faça**
4: $Rota'_i \leftarrow Remove(cliente, Rota_i)$;
5: $Rota'_j \leftarrow Insere(cliente, Rota_j)$;
6: **se** $(custo(Rota'_i + Rota'_j) < custo(Rota_i + Rota_j))$ **então**
7: $Rota_i \leftarrow Rota'_i$;
8: $Rota_j \leftarrow Rota'_j$;
9: ***vá para*** linha 1;
10: **fim se**
11: **fim para**
12: **fim para**
13: **fim para**

3.4 Abordagem GRASP para o PRV

A metaheurística GRASP (Greedy Randomized Adaptative Search Procedure) foi proposta por [Feo and Resende, 1989] e consiste em um processo iterativo, onde cada iteração é dividida em duas fases. Na primeira fase, uma solução inicial para o problema é construída, enquanto na fase de busca local, a vizinhança da solução construída é explorada objetivando-se melhorar a qualidade da solução inicial.

Uma importante característica do GRASP é que as soluções obtidas ao longo de sua

execução são independentes de uma iteração para outra [Festa and Resende, 2002], ou seja, nenhuma informação das iterações anteriores é guardada. O critério de parada, normalmente, é estabelecido através do número máximo de iterações ou de um limite de tempo de processamento.

A cada iteração, o custo da solução encontrada é comparado, com o custo da melhor solução gerada até a iteração anterior, caso seja melhor, atualiza-se a melhor solução. No algoritmo 4 é exibido um pseudocódigo do GRASP básico para uma versão que tem como critério de parada um número máximo de iterações, passado como parâmetro.

Nas linhas 1 e 2, o contador de iterações e a variável que armazena o menor custo conhecido são inicializados. A cada iteração definida pelo laço entre as linhas 3 e 10, uma solução é construída e submetida a um algoritmo de busca local (linhas 4 e 5). O custo da solução após a busca local é comparado com o menor custo conhecido e, conforme o resultado da comparação, a solução é atualizada (linha 7). Na linha 11, após o conjunto de iterações terem sido executadas, a melhor solução obtida é retornada.

Algoritmo 4 GRASP($max\_Iteracoes$)

```
1:  i ← 0;
2:  menorCusto ← ∞;
3:  enquanto (i < max_Iteracoes) faça
4:     Solucao ← Construtivo();
5:     Solucao' ← Busca_Local(Solucao);
6:     se (custo(Solucao') < menorCusto) então
7:        menorCusto ← custo(Solucao');
8:        s ← Solucao';
9:     fim se
10:    i ← i + 1;
11: fim enquanto
12: retorne s;
```

No GRASP desenvolvido para o problema apresentado neste trabalho, foram utilizadas as combinações entre as heurísticas construtivas C1, C2 e C3 com as heurísticas de buscas locais BL1 e BL2 adaptadas, denotadas por BL1' e BL2'. Estas adaptações se referem à adição de dois procedimentos executados após a realização das buscas locais, no sentido de melhorar a solução obtida.

Para que o funcionamento das buscas locais fique mais claro, antes da descrição das buscas locais BL1' e BL2', os procedimentos adicionais de *Inserção*, *Aprimoramento* e *Viabilização* são detalhados nos Algoritmos 5, 6 e 7, respectivamente.

O método *Inserção* implementado procura efetuar o máximo possível de inserções de

clientes para uma dada rota. Convém ressaltar que este procedimento não é utilizado durante a construção de uma solução. Trata-se de um procedimento que visa inserir clientes em rotas das quais foi possível ganhar tempo após a fase de busca local, possibilitando inserir novos clientes.

Como pode ser observado pelo o Algoritmo 5, para uma dada rota definida pelo parâmetro do algoritmo, inicialmente deve-se selecionar os nós clientes que irão compor a lista de candidatos LC (linhas 1 e 2) que possam ser inseridos na rota, conforme o tipo de atendimento desta rota, já que o procedimento foi desenvolvido para atender também os casos em que não se tem a base de dados separada por tipo de atendimento.

Algoritmo 5 Inserção($rota$)

```
1:  ta ← rota.tipo_atendimento;
2:  LC ← Candidatos(N, ta);
3:  rota' ← rota;
4:  melhorRota ← rota;
5:  inseriu ← falso;
6:  para cada cliente ∈ LC faça
7:     Adiciona_Melhor(cliente, rota');
8:     se ( (custo(rota') < custo(melhorRota)) & Viavel(rota') ) então
9:        melhorRota ← rota';
10:       inseriu ← verdadeiro;
11:    fim se
12:    rota' ← rota;
13: fim para
14: se (inseriu) então
15:    rota ← melhorRota;
16:    rota ←Inserção(rota);
17: fim se
```

Cada cliente presente na LC é adicionado no arco que implicar em menor custo para rota (linha 7). O procedimento será repetido para todos os clientes, de forma que apenas a inserção que implicou em menor custo para a rota é efetivada. Ao final do procedimento, sempre que uma inserção for realizada (linha 14), uma nova tentativa de inserção será feita na linha 16. O procedimento é chamado até que não se consiga inserir novos clientes.

O método denotado como *Aprimoramento*, descrito sucintamente no algoritmo 6, consiste em remover o cliente mais custoso da rota (linha 2) para que novos clientes sejam inseridos (linha 3). No melhor caso, um cliente ruim será retirado e, um ou mais clientes serão inseridos, melhorando o custo da solução, ou ainda, aumentando o número de clientes atendidos sem acréscimo demasiado no custo da solução.

A função de viabilização, apresentada no Algoritmo 7, tem como objetivo tornar

Algoritmo 6 Aprimoramento($Rota$)

1: $rota' \leftarrow rota$;
2: $Remove\_Nó\_Mais\_Custoso(rota')$;
3: $Inserção(rota')$;
4: **se** $(custo(rota') < custo(rota))$ **então**
5: $\quad rota \leftarrow rota'$;
6: **fim se**

viáveis as rotas que tenham extrapolado o limite de tempo de uma jornada de trabalho a partir da remoção de algum nó cliente.

Na escolha de qual nó será removido (linha 3), busca-se o cliente que possui a maior soma das distâncias a nós adjacentes. Logo após a remoção do cliente selecionado, uma chamada ao procedimento de inserção apresentado no Algoritmo 5 é feita na linha 5.

Algoritmo 7 Viabilização($Solucao$)

1: **para cada** $rota \in Solucao$ **faça**
2: $\quad$ **enquanto** $(tempo(rota) > JORNADA)$ **faça**
3: $\quad\quad Remove\_Pior\_Nó\ (rota)$;
4: $\quad$ **fim enquanto**
5: $\quad Inserção(rota)$;
6: **fim para**

No procedimento *Viabilização*, a inserção de um novo cliente somente é realizada caso a rota não se torne inviável. A vantagem de se tentar inserir um novo cliente após a remoção pode ser vista no exemplo apresentado na Figura 3.5, onde um cliente relativamente distante de um dado nó pertencente à rota é removido, e assim, pode-se integrar um outro cliente a rota sem que a mesma torne-se inviável.

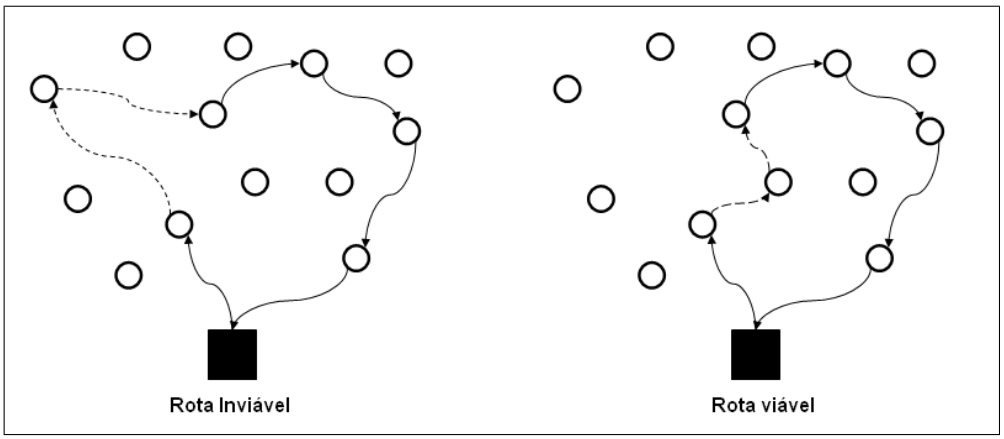

Figura 3.5: Princípio do funcionamento do procedimento de viabilização.

As buscas locais apresentadas neste trabalho fazem uso direto dos procedimentos *Inserção*, *Aprimoramento* e *Viabilização*. A busca BL1' é apresentada no Algoritmo 8

Algoritmo 8 BL1'($Solucao$)

```
1: Solucao' ← Solucao;
2: para cada rota ∈ Solucao' faça
3:    rota ←BL1(rota);
4: fim para
5: enquanto (custo(Solucao') < custo(Solucao)) faça
6:    para cada rota ∈ Solucao' faça
7:       se Tempo_rota < 0,85 × JORNADA então
8:          Esvazia_Rota(rota);
9:       fim se
10:   fim para
11:   para cada rota ∈ Solucao' faça
12:      Inserção(rota');
13:   fim para
14:   para cada rota ∈ Solucao' faça
15:      Aprimoramento(rota');
16:   fim para
17:   Solucao ← Solucao';
18:   para cada rota ∈ Solucao' faça
19:      rota ←BL1(rota);
20:   fim para
21: fim enquanto
22: retorna Solucao';
```

e visa trabalhar em cima de cada rota, melhorando-as ao máximo através do rearranjo proporcionado pela execução da busca local BL1, como indicado na linha 3.

Em seguida, para cada rota da solução será verificado se o tempo total de duração é menor que 85% do tempo da Jornada de Trabalho (linha 7) para que a rota seja esvaziada na linha 8, com o intuito de inserir esses clientes em uma outra rota. O valor de 85% foi adotado empiricamente por apresentar os melhores resultados durante os testes com a busca local BL1.

Na linha 12 o procedimento de *Inserção* é chamado para cada rota. Já na linha 15 tenta-se melhorar ainda mais a qualidade da solução realizando o *Aprimoramento* de cada rota. Após a sequência de tentativas de melhoria na qualidade da solução, caso a solução sido modificada, realiza-se uma nova busca local com BL1, como indicado na linha 19, na tentativa rearranjar as rotas, recomeçando o processo.

A busca local BL2' apresentada no Algoritmo 9 tem o funcionamento semelhante ao da busca local BL1'. A principal diferença consiste na heurística de busca local BL2 ser chamada ao invés da busca local BL1 e o valor percentual do tempo de jornada de trabalho igual a 67% ter obtido melhores resultados para a busca local de *Realocação* nos

testes efetuados.

Pela busca local BL2', enquanto a solução inicial for melhorada através de BL2 (linhas 1 e 19), os procedimentos de Inserção, Aprimoramento e Viabilização são executados repetidamente até que se encontre a melhor solução com base na heurística de BL2.

Algoritmo 9 BL2'($Solucao$)

```
1:  Solucao' ←BL2(Solucao);
2:  Viabilização(Solucao');
3:  enquanto (custo(Solucao') < custo(Solucao)) faça
4:     para cada rota ∈ Solucao' faça
5:        rota ←BL1(rota);
6:     fim para
7:     para cada rota ∈ Solucao' faça
8:        se Tempo_rota < 0,67 × JORNADA então
9:           Esvazia_Rota(rota);
10:       fim se
11:    fim para
12:    para cada rota ∈ Solucao' faça
13:       Inserção(rota');
14:    fim para
15:    para cada rota ∈ Solucao' faça
16:       Aprimoramento(rota');
17:    fim para
18:    Solucao ← Solucao';
19:    Solucao' ←BL2(Solucao);
20:    Viabilização(Solucao');
21: fim enquanto
22: retorna Solucao';
```

A abordagem GRASP proposta (GRASP_PRV) consiste, portanto, no uso das heurísticas de construção e busca local, descritas neste capítulo, durante a execução das iterações definidas no GRASP padrão já apresentado. O Algoritmo 10 descreve o GRASP_PRV desenvolvido neste trabalho. O critério de parada adotado consiste em um número máximo de iterações e o aspecto guloso do algoritmo é definido pelo parâmetro α. Os parâmetros TC e TBL indicam o tipo de heurística construtiva e o tipo de heurística de busca local empregados.

Como pode ser visto, uma vez que tem-se três heurísticas de construção e duas de busca local, um total de seis versões de GRASP foram utilizadas para resolver o problema. No Capítulo 4 são apresentados os resultados computacionais obtidos para o conjunto de abordagens desenvolvidas.

Algoritmo 10 GRASP_PRV($TC, TBL, \alpha, maxIteracoes$)

1: $i \leftarrow 0$;
2: $Solucao \leftarrow \emptyset$;
3: **enquanto** ($i < max\_Iteracoes$) **faça**
4: $Solucao \leftarrow Construtivo(TC, \alpha)$;
5: $Solucao' \leftarrow Busca\_Local(TBL, Solucao)$;
6: **se** ($custo(Solucao') < custo(Melhor\_Solucao)$) **então**
7: $Melhor\_Solucao \leftarrow Solucao'$;
8: **fim se**
9: $i \leftarrow i + 1$;
10: **fim enquanto**
11: **retorna** $Melhor\_Solucao$

Capítulo 4

Resultados Computacionais

Todos os algoritmos foram implementados em linguagem C++ e compilados através do GNU GCC *Compiler* versão 4.3.3 no sistema operacional Linux Ubuntu versão 9.04. Os experimentos foram realizados em um computador com processador Intel Core2Duo 2.2 Ghz rodando em um único núcleo, com 1 GB de memória RAM.

Para facilitar a análise das heurísticas propostas, a Tabela 4.1 apresenta cada versão das heurísticas GRASP desenvolvidas para o problema proposto.

Dois experimentos foram realizados para a avaliação das versões GRASP apresentadas. O primeiro experimento consiste em uma simulação da aplicação real, onde se almeja encontrar rotas para um conjunto de clientes através de uma frota limitada de viaturas, levando-se em consideração a prioridade de atendimento de cada cliente. No segundo experimento, buscam-se soluções que minimizem o número de veículos necessários para atender a todos os clientes, não levando em consideração a prioridade de atendimento dos clientes.

Por não se conhecer instâncias da literatura com características semelhantes as do problema abordado, foram criadas instâncias de forma que seus atributos foram definidos

GRASP	Construtivo	Busca Local
G1	C1	BL1'
G2	C1	BL2'
G3	C2	BL1'
G4	C2	BL2'
G5	C3	BL1'
G6	C3	BL2'

Tabela 4.1: Combinações de heurísticas para composição das versões de GRASP

Instância	**Número de Clientes**	**Número de Veículos**
I_1	100	5
I_2	100	10
I_3	100	15
I_4	250	5
I_5	250	10
I_6	250	15
I_7	500	5
I_8	500	10
I_9	500	15

Tabela 4.2: Primeiro conjunto de instâncias

Instância	**Número de Clientes**
I_{10}	60
I_{11}	70
I_{12}	80
I_{13}	90
I_{14}	100
I_{15}	105
I_{16}	110
I_{17}	120
I_{18}	125
I_{19}	150

Tabela 4.3: Segundo conjunto de instâncias

de forma aleatória. Na Tabela 4.2 é apresentada o conjunto de instâncias utilizadas no primeiro experimento. Três conjuntos de clientes foram combinados com três conjuntos de viaturas.

Para o experimento de minimização da frota de veículos, a Tabela 4.3 mostra as caracterísitcas das dez instâncias utilizadas.

Cada versão GRASP foi executada cem vezes para cada valor do parâmetro α (0,10; 0,20; 0,30 e 0,40). Para a heurística gulosa ($\alpha = 0$), o algoritmo foi executado uma vez. Para avaliação e comparação das versões GRASP propostas foram utilizadas as métricas descritas em [Resende et al., 2008, Ribeiro et al., 2002].

Seja *Best* de cada instância, o valor da melhor solução encontrada dentre todas as execuções dos algoritmos considerados. A métrica *#Best* indica o número de vezes que cada algoritmo avaliado encontrou uma solução de igual valor a *Best* considerando o conjunto total de instâncias.

Para cada algoritmo e instância, seja *Dif* a diferença percentual entre o *Best* da instância e a solução obtida em cada execução do algoritmo. A média de *Dif* sobre todas as execuções do algoritmo para o conjunto de instância define a métrica *MDif*.

Para cada instância, a métrica *NScore* de um algoritmo A indica o número de algoritmos que encontraram uma solução melhor que a solução obtida por A. A soma de todos os valores de *NScore* para todas as instâncias é representado por *Score*, de forma que, quanto menor o valor de *Score*, melhor o algoritmo.

- Experimento: Simulação de aplicação real

Neste experimento, o custo da solução deve levar em consideração tanto a distância como o número de clientes e a prioridade de atendimento dos clientes. Com o objetivo de minimizar a distância, maximizar o número de clientes atendidos e maximizar o número de clientes com prioridades mais altas, a função de custo a ser minimizada foi definida como:

$$\text{Custo} = \frac{\text{Distância total percorrida}}{\text{Número de Clientes + Soma das Prioridades}}$$

A partir da Tabela 4.2, três conjuntos de instâncias foram definidos conforme o número de clientes. Os resultados apresentados na Tabela 4.4 mostram o comportamento dos algoritmos para o grupo de instâncias com 100 clientes. Como pode ser observado pelos valores destacados em negrito, o algoritmo que apresentou melhor resultado quanto às métricas *#Best* e *Score* foi o *G4*, que combina a heurística construtiva que leva em consideração a prioridade dos clientes, com a busca local de realocação.

Percebe-se, pela métrica *#Best*, que foram encontradas poucas soluções com os mesmos valores de *Best*. Devido aos três fatores que determinam o custo da solução (distância, número de clientes e prioridade total), além do elevado número de combinações possíveis entre os clientes presentes na solução e os que clientes que não estão presentes na solução, ocorre uma grande variedade nos resultados (mesmo que próximos ao valor de *Best*) e poucas reocorrências de valores iguais a de *Best*.

Outro dado importante refere-se aos valores de *Mdif*. Nota-se que os valores das diferenças médias são relativamente altos. Isto sucede da comparação da melhor solução encontrada para cada instância com os diversos valores estabelecidos para α (0; 0,10; 0,20; 0,30 e 0,40). Observa-se também que não há uma grande divergência entre os valores de *MDif* obtidos pelas diferentes versões do GRASP que possuem métodos construtivos

Instâncias	Métricas	Versões GRASP					
		G1	G2	G3	G4	G5	G6
$\{I_1, I_2, I_3\}$	*Mdif (%)*	6,54	**6,44**	6,75	6,82	7,10	7,14
	#Best	**2**	0	1	**2**	0	1
	Score	5	7	**3**	**3**	12	8
	Tempo (s)	3,087	3,140	3,117	3,170	**3,085**	3,149

Tabela 4.4: Resumo dos resultados das heurísticas para as instâncias com 100 clientes.

iguais neste experimento.

Além do fato de se intensificar a busca local sempre que houver uma melhora no custo da solução, a função de *Aprimoramento* está sempre promovendo, não somente um movimento de inter-rota, como também um movimento de intra-rota, desempenhando um papel marcante nas duas buscas locais, conduzindo a resultados semelhantes na métrica *MDif*.

As Tabelas 4.5 e 4.6 apresentam os resultados para as instâncias de 250 (I_4, I_5 e I_6) e 500 clientes (I_7, I_8 e I_9), respectivamente. Observa-se que o algoritmo *G1* apresentou os melhores resultados tanto em relação a métrica *#Best*, quanto para a métrica *Score* para estes dois conjuntos de instâncias.

Instâncias	Métricas	Versões GRASP					
		G1	G2	G3	G4	G5	G6
$\{I_4, I_5, I_6\}$	*Mdif (%)*	7,35	**7,18**	7,44	7,22	8,36	8,22
	#Best	**2**	0	1	0	0	1
	Score	**4**	8	6	5	6	7
	Tempo(s)	7,743	7,728	7,594	7,616	**7,521**	7,554

Tabela 4.5: Resumo dos resultados das heurísticas para as instâncias com 250 clientes.

A partir dos resultados obtidos em ambas as tabelas, pode-se inferir que a versão *G1*, que combina a heurística construtiva de inserção de clientes de menor distância com a heurística de busca local *2-optimal*, apresentam resultados mais satisfatórios para instâncias com um número maior de clientes. O que pode explicar este comportamento é o fato de que, durante a fase construtiva, um número maior de clientes é inserido na solução, mesmo que possuam prioridades mais baixas. Com a fase de busca local, ocorre uma re-organização da rota possibilitando novas inserções. Como a fase de inserção na busca local visa diminuir o custo da solução, acaba-se por inserir clientes de maior prioridade, contribuindo assim tanto no número de clientes atendidos como também na prioridade total dos clientes atendidos.

Como forma de avaliar a contribuição de cada fase do GRASP na redução do custo

Instâncias	Métricas	Versões GRASP					
		G1	G2	G3	G4	G5	G6
$\{I_7, I_8, I_9\}$	*Mdif (%)*	**9,63**	9,64	11,97	12,08	17,03	16,87
	#Best	**2**	0	**2**	0	0	1
	Score	**2**	5	3	9	14	8
	Tempo(s)	21,750	21,809	20,873	20,868	**19,599**	19,632

Tabela 4.6: Resumo dos resultados das heurísticas para as instâncias com 500 clientes.

final da solução, um dos valores de α foi escolhido aleatoriamente e o conjunto de soluções obtidas por cada algoritmo foi considerado de forma a se extrair o custo médio da fase de construção e o custo médio da solução final para as 100 execuções. O valor do custo médio das duas fases foi comparado com o valor da pior solução obtida em todo o experimento.

As Figuras 4.1 e 4.2 ilustram a contribuição dada pela fase de busca local em cada versão GRASP quando o parâmetro de aleatoriedade α é igual a $0,4$. O valor máximo (igual a $1,0$) do gráfico corresponde ao custo da pior solução obtida na fase construtiva. Cada coluna apresenta a diferença percentual em relação ao valor da solução média do algoritmo avaliado.

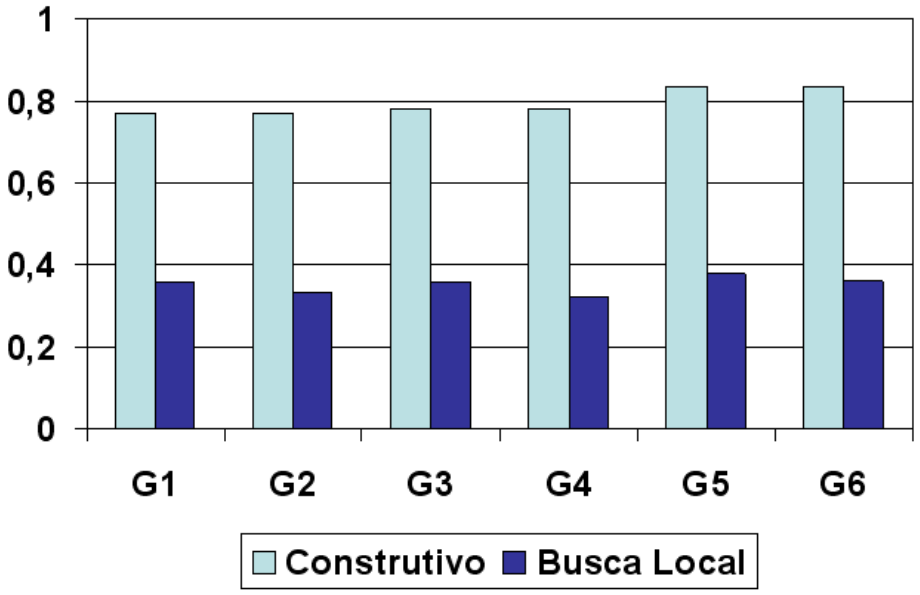

Figura 4.1: Contribuição da Busca Local na instância I_1 com $\alpha = 0,4$

Pela Figura 4.1, observa-se que as versões que utilizam o construtivo *C3* (*G5* e *G6*) apresentaram o pior resultado na fase de construção, com custos médios próximos a 83% da pior solução para a instância I_1, contra 77% do construtivo das versões que usam *C1* (*G1* e *G2*). Com a fase de busca local, os algoritmos *G2* e *G4* apresentaram as maiores reduções nos custos médios da soluções obtidas na fase de construção, caindo a valores

próximos de 35% e 34%, respectivamente. No geral, todas as buscas locais implicaram na diminuição do custo da solução obtida na fase construtiva em até 40%.

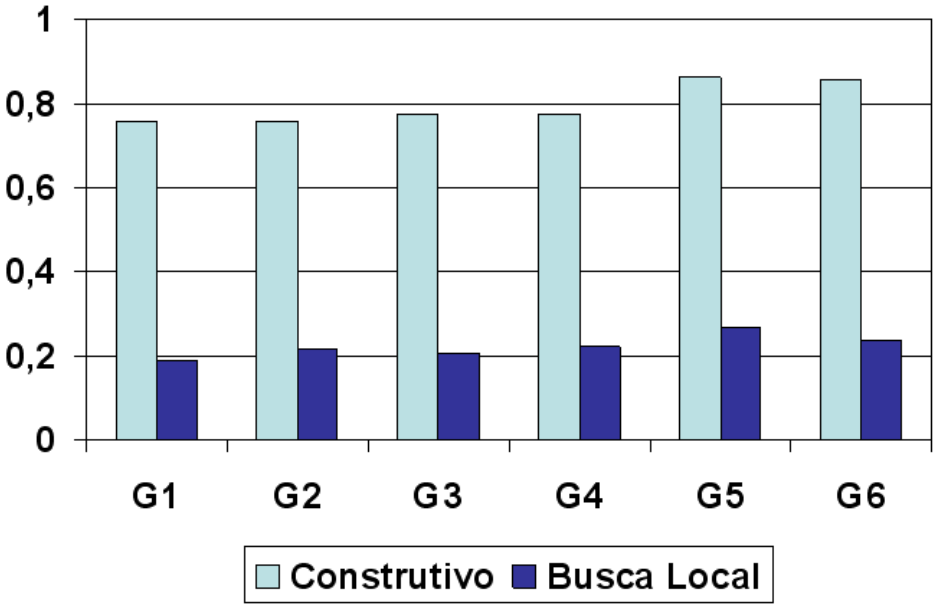

Figura 4.2: Contribuição da Busca Local na instância I_9 com $\alpha = 0,4$

Percebe-se pela Figura 4.2 que a contribuição dada pela busca local na instância I_9 foi maior do que a obtida na instância I_1, sugerindo que quanto maior a instância, mais se consegue intensificar a busca local para melhorar a solução, implicando em uma redução maior do custo. Para esta instância, em todas as execuções, quando o parâmetro de aleatoriedade α foi igual a $0,4$, a versão $G1$ apresentou o melhor resultado $(0,19)$, em contrapartida $G5$ apresentou o pior resultado $(0,26)$. A maior redução no custo da solução foi dada pela versão $G1$, totalizando uma redução de 56% da fase construtiva para a fase de busca local.

- Experimento: Minimização da frota

Neste experimento as versões GRASP apresentadas foram avaliadas e comparadas quanto ao objetivo de minimização do número de veículos necessários ao atendimento de todos os clientes que demandam serviço. A frota foi considerada ilimitada, ou seja, enquanto existirem clientes atendidos, uma nova viatura foi designada para atendimento.

Outro ponto importante do experimento é que não foram consideradas as prioridades dos clientes. Assim, o objetivo foi diminuir a distância entre os clientes visitados, o que influencia diretamente no número de veículos necessários para atender a todos os clientes.

Instâncias	Métricas	Versões GRASP			
		G1	G2	G5	G6
$\{I_{10}, .., I_{19}\}$	*Mdif (%)*	20,72	**11,94**	14,40	12,19
	#Best	1	3	2	**4**
	Score	24	14	13	**7**
	Tempo(s)	**2,237**	6,280	3,128	5,613

Tabela 4.7: Resumo dos resultados das heurísticas para o experimento de minimização da frota.

Para o experimento foram utilizadas dez instâncias ($I_{10}, ..., I_{19}$), e como a prioridade dos clientes não foi considerada, as versões GRASP que utilizam a heurística construtiva *C2*, que correspondem às versões *G3* e *G4*, não foram avaliadas.

Os resultados da Tabela 4.7 mostram a superioridade, quanto a *#Best* e *Score*, da versão *G6*, que combina a fase construtiva *C3*, que leva em consideração o número de vizinhos, com a busca local de realocação *BL2'*. Entretanto, pela métrica *MDif*, observa-se que *G2* configurou-se uma abordagem interessante. Quanto ao tempo médio, porém, *G2* e *G6* registraram os tempos de execuções mais elevados, o que pode ser justificado pelo fato de *G2* e *G6* conseguiram intensificar mais a fase de busca local.

Observou-se ainda que neste experimento, diferentemente do experimento da aplicação real, as versões GRASP com a busca local de realocação obtiveram melhores resultados do que as heurísticas baseadas na busca local *2-optimal*.

Como a frota é ilimitada, após a fase construtiva todos os nós clientes pertencem a solução. Assim, a função de *Aprimoramento* não contribui de maneira expressiva para a redução dos custos das versões baseadas na busca local *BL1'*, pois o algoritmo não consegue fazer trocas de elementos que implicariam em uma melhora no custo da solução. Já as versões baseadas na busca local *BL2'* obtém vantagem por realizar movimentos inter-rota e também intra-rota (através da função de *Aprimoramento*).

A Figura 4.3 apresenta o gráfico da redução do custo da solução na fase construtiva para fase de busca local da instância I_{19}. Neste experimento, a fase construtiva *C3* (construtivo de *G5* e *G6*) apresentou melhores resultados (0,88), contra 0,95 de *C1* (construtivo de *G1* e *G2*). A busca local que obteve melhores resultados foi a *BL2'*. Observa-se que na versão *G6* a busca local conseguiu reduzir o custo para 0,31, enquanto que na versão *G2* a redução ficou próxima ao resultado de *G6* com 0,32.

A análise da redução do número de veículos deve ser feita cuidadosamente, pois mesmo que duas soluções consigam reduzir o número de viaturas para um mesmo valor, só o

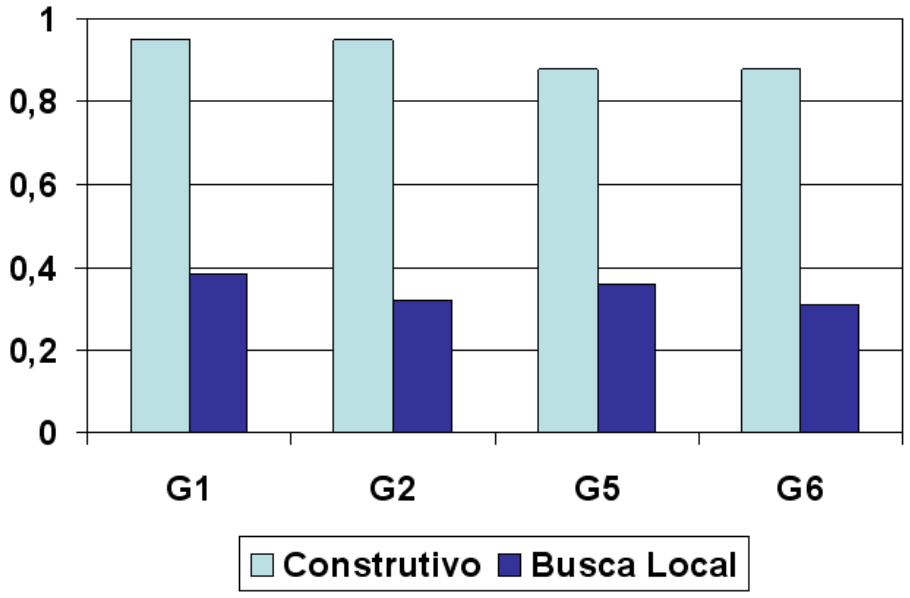

Figura 4.3: Contribuição da busca local com a instância I_{19} e $\alpha = 0,4$

tamanho da frota não é suficiente para dizer se uma solução é melhor que a outra. Por exemplo, na figura 4.4 tem-se que *G1* e *G2* começam a fase construtiva com 27 viaturas e *G5* e *G6* com 26. Após a busca local *G1* e *G5* possuem um total de 17 viaturas e *G2* e *G6* possuem 16 veículos.

Mesmo que as versões possuam o mesmo número de viaturas, a qualidade da solução deve ser dada também em função da distância total percorrida. Pois assim, encontra-se a solução que representa um menor custo real.

Na Figura 4.5, observa-se que apesar da solução apresentada por *G5* possuir uma viatura a mais que a solução apresentada por *G2* após a fase de busca local, pelo que se observa na Figura 4.5 mostra que a distância total percorrida na solução de *G5* aproxima-se das distâncias de *G2* e *G6*. Isto ocorre porque, mesmo que apenas um cliente não tenha sido atendido, uma viatura sempre será designada a atender este cliente, aumentando o número de viaturas, porém, aumentando relativamente pouco a distância total percorrida.

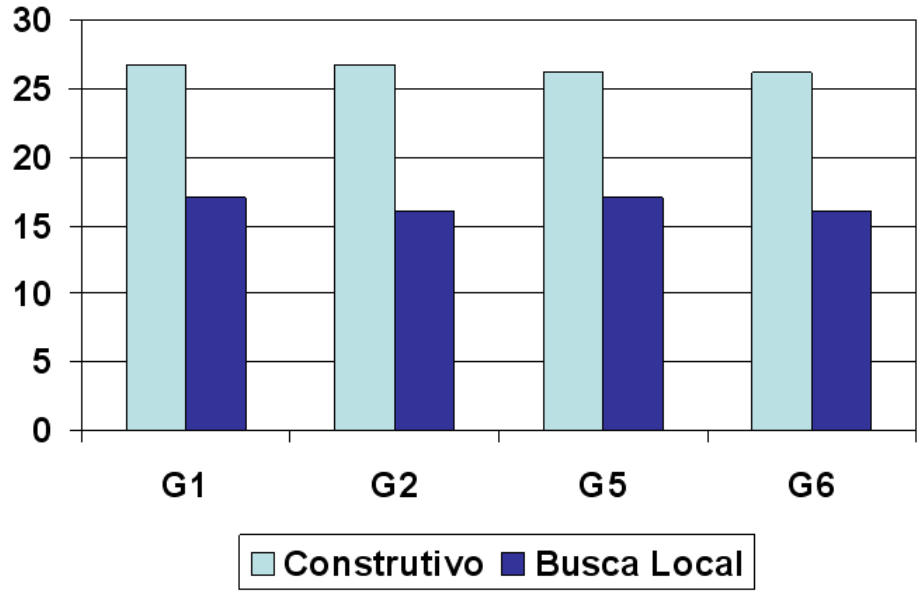

Figura 4.4: Redução do número de veículos para a instância I_{19} com $\alpha = 0,4$

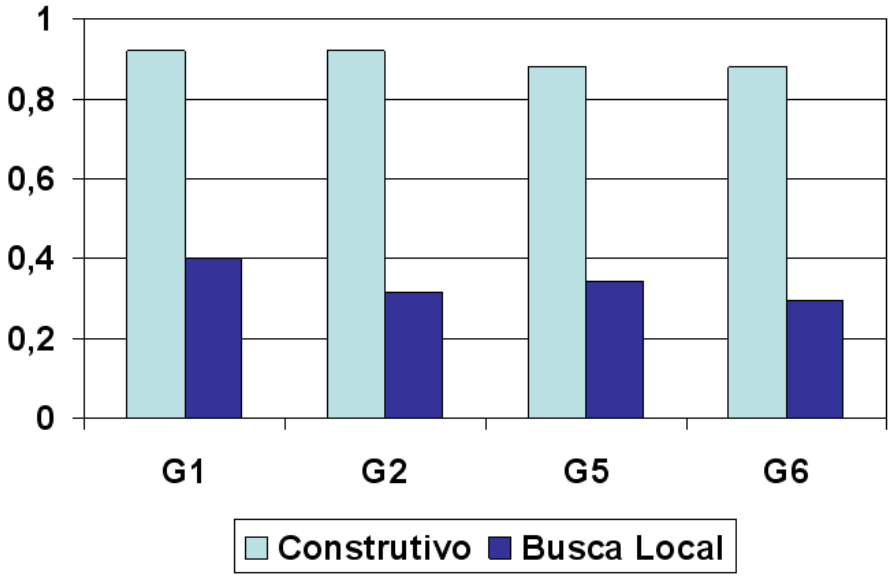

Figura 4.5: Redução da distância percorrida para a instância I_{19} com $\alpha = 0,4$

Capítulo 5

Conclusão e Trabalhos Futuros

Neste trabalho foi apresentada uma variação do Problema Clássico de Roteamento de Veículos originada de uma aplicação real, para a qual foram desenvolvidas seis versões heurísticas baseadas na metaheurística GRASP.

Os resultados mostraram que a fase de busca local apresentou reduções expressivas na minimização do custo das soluções para instâncias maiores, o que é um bom diferencial em termos de aplicação real. Percebeu-se também que para cada experimento, uma versão GRASP se apresentou mais eficaz dentre as seis versões propostas, mostrando a importância no desenvolvimento de heurísticas diferentes para avaliar e comparar qual se comporta melhor conforme as características do problema e das instâncias.

Além disso, este trabalho serviu como ferramenta inicial de estudo para outros alunos no sentido de compreender, não somente o Problema de Roteamento de Veículos, como também a importância do uso de heurísticas na solução de problemas de otimização combinatória.

Em termos de trabalhos futuros, pode-se citar tanto o desenvolvimento de novos módulos que ajudem a perturbar a solução, buscando com isso obter resultados de melhor qualidade do que as versões originais, como também uma avaliação das heurísticas propostas para um número maior de instâncias. Por fim, o desenvolvimento de metaheurísticas com abordagens diferentes da heurística GRASP seria fundamental para se comparar o desempenho entre as versões aqui apresentadas.

Um outro trabalho interessante a ser encaminhado refere-se a consideração da prioridade de atendimento do cliente como uma janela de tempo, que variaria conforme o valor da prioridade. Na aplicação real deste problema, um modelo de tempo real é requerido, onde permite-se alteração e reconfiguração de rotas ao longo do horizonte de

planejamento.

Referências

[Alba E., 2004] Alba E., D. B. (2004). Solving the vehicle routing problem by using cellular genetic algorithms. *Conference on Evolutionary Computation in Combinatorial Optimization*, 3004:11–20.

[B. Bullnheimer and Strauss, 1997] B. Bullnheimer, R. F. H. and Strauss, C. (1997). Applying the ant system to the vehicle routing problem. *2nd International Conference on Metaheuristics*.

[Baldacci et al., 2007] Baldacci, R., Toth, P., and Vigo, D. (2007). Recent advances in vehicle routing exact algorithms. *4OR: A Quarterly Journal of Operations Research*, 5(4):269–298.

[Bräysy and Gendreau, 2001] Bräysy, O. and Gendreau, M. (2001). Genetic algorithms for the vehicle routing problem with time windows. Technical report, Department of Optimization, Oslo, Norway.

[Cordenonsi, 2008] Cordenonsi, A. (2008). *Ambientes, Objetos e Dialogicidade: Uma Estratégia de Ensino Superior em Heurísticas e Metaheurísticas*. PhD thesis, Universidade Federal do Rio Grande do Sul.

[Cruz and Oliveira, 2008] Cruz, E. P. and Oliveira, T. T. (2008). Redução de custos em transportes rodoviários: o estudo de caso de uma distribuidora multinacional de combustíveis líquidos. *Revista Pensamento Contemporâneo em Administração*, (2):64–75.

[Dantzig and Ramser, 1959] Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91.

[Feo and Resende, 1989] Feo, T. and Resende, M. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71.

[Festa and Resende, 2002] Festa, P. and Resende, M. (2002). GRASP: An annotated bibliography. In Ribeiro, C. and Hansen, P., editors, *Essays and surveys in metaheuristics*, pages 325–367. Kluwer Academic Publishers.

[Fisher et al., 1997] Fisher, M. L., Jörnsten, K. O., and Madsen, O. B. G. (1997). Vehicle routing with time windows: Two optimization algorithms. *Operations Research*, 45(3):488–492.

[Fukasawa et al., 2006] Fukasawa, R., Longo, H., Lysgaard, J., Poggi, D., Reis, M., Uchoa, E., and Werneck, R. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Math Programming*, 106:491511.

[Goldbarg and Luna, 2005] Goldbarg, M. C. and Luna, H. P. L. (2005). *Otimização Combinatória e Programação Linear - Modelos e Algoritmos*. Elsevier Editora Ltda, Rio de Janeiro - RJ, Brazil.

[Golden et al., 1981] Golden, B., Ball, M., and Bodin, L. (1981). Current and future research directions in network optimization. *Computers & Operations Research*, 8(2):71 – 81.

[L.Fisher and R.Jaikumar, 1981] L.Fisher, M. and R.Jaikumar (1981). A generalized assignment heuristic for vehicle routing. *Networks*, 11:109–124.

[Lin and Kernighan, 1973] Lin, S. and Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516.

[Montané, 2006] Montané, F. A. T. (2006). Um algoritmo de busca tabu adaptativa para o prv com frota mista. *XXVI ENEGEP*.

[P. Czarnas and Gocyla, 2004] P. Czarnas, Z. C. and Gocyla, P. (2004). Parallel simulated annealing for bicriterion optimization problems. *PPAM*, pages 233–240.

[Reimann et al., 2003] Reimann, M., Doerner, K., and Hartl, R. (2003). Analyzing a unified ant system for the vrp and some of its variants. *EvoWorkshops*, pages 300–310.

[Renaud and Boctor, 2002] Renaud, J. and Boctor, F. F. (2002). A sweep-based algorithm for the fleet size and mix vehicle routing problem. *European Journal of Operational Research*, 140(3):618 – 628.

[Resende et al., 2008] Resende, M., Martí, R., Gallego, M., and Duarte, A. (2008). GRASP and path-relinking for the max-min diversity problem. Technical Report, AT&T Labs Research, Florham Park, New Jersey.

[Ribeiro et al., 2002] Ribeiro, C., Uchoa, E., and Werneck, R. (2002). A hybrid GRASP with perturbations for the steiner problem in graphs. *INFORMS J. on Computing*, 14(3):228–246.

[Soares, 2004] Soares, S. S. R. F. (2004). Metaheurísticas para o problema de clusterização automática. Master's thesis, Instituto de Computação - Universidade Federal Fluminense.

[Toth and Vigo, 2002] Toth, P. and Vigo, D. (2002). *The Vehicle Routing Problem*. SIAM - Monographs on Discrete Mathematics and Applications.