UNIVERSIDADE FEDERAL FLUMINENSE

ROGÉRIO DA SILVA

Metaheurística aplicada ao Problema de Recobrimento de Rotas com Coleta de Prêmios

NITERÓI-RJ

2014

UNIVERSIDADE FEDERAL FLUMINENSE

ROGÉRIO DA SILVA

Metaheurística aplicada ao Problema de Recobrimento de Rotas com Coleta de Prêmios

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Algoritmos e Otimização

Orientador:
Isabel Cristina Mello Rosseti

NITERÓI-RJ

2014

ROGÉRIO DA SILVA

Metaheurística Aplicada ao Problema de Recobrimento de Rotas com Coleta de Prêmios

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Algoritmos e Otimização

Aprovada em Abril de 2014.

BANCA EXAMINADORA

Profª. Isabel Cristina Mello Rosseti - Orientadora, UFF
(Presidente)

Prof. Luiz Satoru Ochi, UFF

Prof. Haroldo Gambini Santos, UFOP

Niterói-RJ
2014

À minha família.

Agradecimentos

Ao meus pais, Luiz (*In Memoriam*) e Francinete, por tudo que sou.

Aos meus irmãos Jorge e Francineide pela força e incentivo.

À minha esposa Francineide e ao meu filho Erick pelo amor, apoio e compreensão, principalmente, nos momentos difíceis em que ficamos distantes durante esta jornada.

A todos os colegas, professores e técnicos-administrativos, do IFPI participantes do convênio IC-UFF/IFPI.

À minha orientadora, Profa. Isabel Rosseti, por ter me guiado com muita dedicação durante a construção deste trabalho, mesmo diante de tantas adversidades.

Aos demais professores e técnicos do IFPI que participaram de alguma forma deste projeto.

Ao IFPI, pela oportunidade de participação nesse programa.

Aos demais professores e técnicos do IC/UFF envolvidos no convênio com IFPI: Aura, Helena, Loana, Satoru, Christiano, Viviane, Martinhon, Esteban e Murta.

Em fim, a todos aquele que de alguma forma, em qualquer momento da minha vida, tenha contribuído para que eu pudesse ter condições para chegar a esta conquista.

Muito obrigado!

Resumo

O problema de recobrimento de rotas com coletas de prêmios (PRRCP) pode ser definido em um grafo não direcionado e simétrico $G = (V \cup W, E)$, onde $V \cup W$ é o conjunto dos vértices e E é o conjunto das arestas. O conjunto V é composto de nós obrigatórios T e opcionais $V \backslash T$. A cada vértice de V tem-se um prêmio não negativo associado. O conjunto W representa os vértices que devem ser cobertos por algum vértice do subconjunto V. Assim, o PRRCP consiste em encontrar um ciclo hamiltoniano de custo mínimo que contenha todos os vértices obrigatórios de T e que todos os vértices de W estejam cobertos, realizando uma coleta mínima de prêmios. Nesse trabalho propõe-se um algoritmo heurístico híbrido para solucionar o PRRCP, baseado na metaheurística GRASP hibridizada com as heurísticas GENIUS, na fase construtiva, e VND na fase de busca local, denominado GRASP-GENIUS-VND. Para validar o algoritmo proposto foram realizados testes com dois grupos de problemas-teste da literatura. Os resultados desses experimentos comprovaram que a nova abordagem desenvolvida neste trabalho mostrou-se competitiva em relação a heurística considerada estado da arte, onde das 144 instâncias, o GRASP-GENIUS-VND conseguiu alcançar, ou superar, os melhores resultados em 129 delas. A melhoria em qualidade das soluções foi, em média, de 57.95%. Já a redução do tempo computacional médio alcançou 49.32%.

Palavras-chave: Problema de Recobrimento de Rotas com Coletas de Prêmios. Metaheurísticas. GRASP. GENIUS.

Abstract

The prize collecting covering tour problem (PCCTP) can be defined in a non-directed and symmetric graph $G = (V \cup W, E)$, where $V \cup W$ is the set of nodes and E the set of edges. T is the set of mandatory vertices in V, that must be present in every solution. The optional vertices belong to the set $V \backslash T$. Each node of V has a non-negative associated prize. The set W represents the vertices that might be covered by some vertex of the subset V. Therefore, the PCCTP consists in finding a minimal cost hamiltonian cycle that contains all required vertices of T, also all vertices of W are covered, minimizing the collected prize. In this work, we proposed a hybrid GRASP heuristic (called GRASP-VND-GENIUS), based on GENIUS approach in the construction phase, and the improvement strategy VND in local search step. Computational experiments, conducted on a set of 144 instances from the literature, showed that this new hybrid method outperformed the state-of-the-art heuristic both in mean solution values and execution time, in 129 of them. The improved solution costs obtained were, on average, equal to 57.95%, reducing the computational efforts in 49.32%.

Keywords: Prize Collecting Covering Tour Problem, Metaheuristic, GRASP, GENIUS .

Sumário

Lista de Abreviaturas e Siglas viii

Lista de Figuras ix

Lista de Tabelas x

1 Introdução 1

2 Problema de Recobrimento de Rota com Coletas de Prêmios - PRRCP 4

2.1 Descrição do Problema Abordado 4

2.2 Revisão de Literatura 7

2.2.1 Formulações Matemáticas 9

2.2.2 Regras de Redução 12

3 Heurísticas e Metaheurísticas 15

3.1 Heurísticas 15

3.1.1 Heurísticas Construtivas 16

3.1.1.1 Inserção Mais Barata 16

3.1.1.2 GENIUS 17

3.1.2 Heurística de Refinamento 22

3.1.2.1 Método de Descida em Vizinhança Variável (VND) 22

3.2 Metaheurísticas 24

3.2.1 GRASP 24

4 Metodologia Aplicada **28**

4.1 Avaliação de uma solução do PPRCP 28

4.2 Estruturas de Vizinhança 29

4.3 Algoritmo GRASP aplicado ao PRRCP 31

4.3.1 GRASP-GENIUS-VND 31

4.3.1.1 Método de Construção de Solução Inicial 33

4.3.1.2 GENIUS adaptada ao PRRCP 33

4.3.1.3 Método de Descida em Vizinhança Variável (VND) para PRRCP 35

5 Resultados Computacionais **38**

5.1 Ambiente Computacional 38

5.2 Instâncias do PRRCP 39

5.3 Definição do valor de alfa 40

5.4 Resultados e Comparações 48

5.5 Significância Estatística 58

5.6 Análise do Comportamento das Estratégias 59

5.7 Considerações Finais 61

6 Conclusões e Trabalhos Futuros **63**

Referências **65**

Lista de Abreviaturas e Siglas

GRASP	:	*Greedy Randomized Adaptive Search Procedure*;
GENI	:	*Generalized Insertion*;
ILS	:	*Iterated Local Search*;
LC	:	Lista de Candidatos;
LRC	:	Lista Restrita de Candidatos;
PCV	:	Problema do Caixeiro Viajante;
PRR	:	Problema de Recobrimento de Rotas;
PRRCP	:	Problema de Recobrimento de Rotas com Coleta de Prêmios;
RVND	:	*Random Variable Neighborhood Descent*;
VND	:	*Variable Neighborhood Descent*;
US	:	*Unstringing and Stringing.*

Lista de Figuras

2.1 Rota construída sobre uma instância do PRR. 5

2.2 Rota construída sobre um grafo instância do PRRCP. 7

3.1 GENI - Inserção Tipo 1 de vértice v entre v_i e v_j 19

3.2 GENI - Inserção Tipo 2 de vértice v entre v_i e v_j 19

3.3 US - Remoção Tipo 1 de vértice v_i da rota 21

3.4 US - Remoção Tipo 2 de vértice v_i da rota 22

5.1 Distribuição da probabilidade acumulada para instância *gr96_ VT18_ T58_ W20_ 25*, do Grupo 1, com alvo 42746. 60

5.2 Distribuição da probabilidade acumulada para instância *pr264_ VT52_ T53_ W159_ 25*, do Grupo 2, com alvo 27693. 61

Lista de Tabelas

5.1 Definição do valor de alfa - Grupo 1. 41

5.1 Definição do valor de alfa - Grupo 1. 42

5.1 Definição do valor de alfa - Grupo 1. 43

5.1 Definição do valor de alfa - Grupo 1. 44

5.1 Definição do valor de alfa - Grupo 1. 45

5.2 Definição do valor de alfa - Grupo 2. 46

5.2 Definição do valor de alfa - Grupo 2. 47

5.3 Comparação dos resultados entre o GRASP-GENIUS-VND e o ILS-MRD_AD [35] - Grupo 1. 49

5.3 Comparação dos resultados entre o GRASP-GENIUS-VND e o ILS-MRD_AD [35] - Grupo 1. 50

5.3 Comparação dos resultados entre o GRASP-GENIUS-VND e o ILS-MRD_AD [35] - Grupo 1. 51

5.3 Comparação dos resultados entre o GRASP-GENIUS-VND e o ILS-MRD_AD [35] - Grupo 1. 52

5.3 Comparação dos resultados entre o GRASP-GENIUS-VND e o ILS-MRD_AD [35] - Grupo 1. 53

5.4 Comparação dos resultados entre o GRASP-GENIUS-VND e o ILS-MRD_IB [35] - Grupo 2. 54

5.4 Comparação dos resultados entre o GRASP-GENIUS-VND e o ILS-MRD_IB [35] - Grupo 2. 55

5.5 Resumo da comparação entre as abordagens testadas - Grupo 1. 56

5.6 Resumo da comparação entre as abordagens testadas - Grupo 2. 57

5.7 Análise de Significância Estatística 58

Capítulo 1

Introdução

A prestação de serviços de assistência social, médica ou jurídica a comunidades geograficamente distribuídas e distantes dos grandes centros urbanos tem sido uma preocupação das diversas entidades públicas, filantrópicas ou de iniciativa privada. Implantar unidades fixas nessas localidades envolvem altos investimentos e ainda existe a carência de profissionais com disponibilidade para fixação nessas regiões.

Uma das soluções mais praticadas para atender essas pessoas, que residem em locais distantes dos centros de serviços com profissionais e equipamentos devidamente adequados, tem sido a disponibilização de equipes itinerantes. Como exemplo, o Tribunal de Justiça do Piauí (TJ-PI) tem um projeto chamado Justiça Itinerante, no qual um ônibus adaptado com uma estrutura de um fórum, apto a realizar atendimentos jurídicos, foi adquirido. Esse ônibus periodicamente percorre cidades distantes das unidades do TJ-PI, ou bairros da periferia da capital, abrangendo, assim, uma maior quantidade de cidadãos.

A situação relatada acima é uma das aplicações de problemas de otimização combinatória que envolvem escalonamento de recursos. Outras aplicações reais de problemas de otimização combinatória têm sido objetos de pesquisa, tais como, roteamento diversos, planejamento de produção, localização de máquinas e instalações de manufaturas, biologia computacional, entre outros. Geralmente esses problemas são generalização do clássico problema do caixeiro viajante (PCV) [13, 14]. Nesta dissertação aborda-se o problema de recobrimento de rotas com coletas de prêmios (PRRCP) [16, 35], cujo o objetivo consiste em construir rotas de comprimento mínimo atendendo a um conjunto de restrições previamente estabelecido.

Por pertencer a classe dos problemas NP-difíceis, assim como o PCV, propõem-se neste trabalho um algoritmo heurístico que utiliza a metaheurística *Greedy Randomi-*

zed Adaptive Search Procedures (GRASP) [7, 8] hibridizada com as heurísticas GENIUS [10] e a busca local *Variable Neighborhood Descent* (VND) [18] para resolver de maneira aproximada o PRRCP.

O PRRCP representa uma situação muito presente e de grande relevância na área de planejamento logístico de deslocamentos, principalmente, quando tem-se disponível uma única equipe para atender a um conjunto de pontos de atendimento. Ele envolve diretamente otimização de recursos materiais e humanos em empresas privadas ou em órgãos governamentais.

Estudos que levem a resolvedores eficientes para este problema, otimizando os recursos envolvidos, bem como consumindo o mínimo de recursos computacionais e tempo de processamento, são de grande importância para administração de empresas e órgãos. A utilização da metaheurística GRASP hibridizada com as heurísticas GENIUS [10] e *Variable Neighborhood Descent* (VND) [18] para resolver o PRRCP é, mediante levantamento bibliográfico, academicamente inédito.

O PRRCP foi introduzido na Dissertação de Lyra [16]. Ele é uma generalização do problema de recobrimento de rotas (PRR) [4] que, por sua vez, é uma especialização do PCV.

O PRRCP pode ser definido em um grafo não direcionado $G = (N, E)$, onde N é o conjunto de vértices e E é o conjunto de arestas. Os vértices N são divididos em dois subconjuntos de vértices, $N = (V \cup W)$. O conjunto dos vértices V é composto por vértices obrigatórios T e vértices opcionais $V \backslash T$. A cada vértice $i \in V$ há um prêmio p_i associado, onde $p_i > 0$. Já o conjunto W é composto pelos vértices que devem ser cobertos por elementos do conjunto V presentes na solução. Dada uma distância D, entende-se por cobertura a existência na solução de pelo menos um vértice V, a uma distância menor ou igual a D, de cada vértice de W. Uma solução viável para o problema PRRCP, além de conter todos os vértices T e cobrir os elementos do conjunto W, deve ainda coletar um valor mínimo de prêmios dado por $PRIZE$, obtido pelo somatório dos prêmios dos vértices de V que compõem a solução.

Este trabalho tem como objetivo geral a elaboração de um novo algoritmo heurístico híbrido, baseado na metaheurística GRASP e em técnicas construtivas e de refinamento, eficiente na resolução do PRRCP. Além deste objetivo geral, tem-se ainda os seguintes objetivos específicos:

- realizar uma revisão de literatura sobre o PRRCP, problemas correlatos, e técnicas

de solução já propostas;

- investigar sobre métodos heurísticos de construção e de melhoria de soluções, assim como sobre metaheurísticas;
- desenvolver um algoritmo baseado na metaheurística GRASP, hibridizado com procedimentos de Busca Local com múltiplas vizinhanças para construir soluções de boa qualidade em tempo computacional viável para o PRRCP; e
- comparar o desempenho do algoritmo desenvolvido com as abordagens disponíveis na literatura.

Esta dissertação está organizada em seis capítulos, incluindo esta seção de introdução.

O Capítulo 2 faz uma revisão detalhada do problema abordado neste trabalho. São apresentadas também aplicações, formulações matemáticas e regras de redução encontradas na literatura relacionados ao PRRCP.

No Capítulo 3, é realizado um levantamento bibliográfico a respeito das técnicas utilizadas no algoritmo proposto neste trabalho, incluindo-se heurísticas construtivas e de refinamento e a metaheurística GRASP.

A metodologia proposta é detalhada no Capítulo 4. Neste capítulo é explicada a estrutura geral do algoritmo GRASP proposto, assim como os métodos de geração de solução inicial e de busca local. Estruturas de vizinhança utilizadas também são apresentadas.

No Capítulo 5 os experimentos computacionais realizados são descritos. Posteriormente é feita a análise dos resultados obtidos, comparando-os com dados disponíveis na literatura. Por fim, no Capítulo 6, conclusões do trabalho realizado e sugestões de trabalhos futuros são apresentados.

Capítulo 2

Problema de Recobrimento de Rota com Coletas de Prêmios - PRRCP

Este capítulo descreve em detalhes o problema abordado nesta dissertação, o PRRCP. É feito um levantamento bibliográfico do PRRCP. Trabalhos relacionados, formulações matemáticas e regras de reduções disponíveis para esse problema são apresentados e discutidos.

2.1 Descrição do Problema Abordado

O PRRCP, introduzido em [16], é uma variação do PRR [4]. Esse último é uma especialiação do PCV (mais detalhes sobre PCV em [13, 14]). O PRR consiste em encontrar um caminho de menor custo, em um grafo não direcionado e simétrico, $G = (N, E)$, dado um conjunto de elementos e restrições, a saber:

- N é o conjunto de todos os vértices disponíveis, onde $N = (V \cup W)$;
- $E = \{(i, j) \mid i, j \in N, i \neq j\}$ é o conjunto de arestas;
- V é o conjunto do vértices que podem fazer parte de uma solução;
- W é o conjunto de vértices que deve ser coberto por algum vértice V da solução. Entende-se por cobertura a existência da solução de pelo menos um vértice V que esteja a uma distância igual ou inferior a distância D. A distância de cobertura D também é uma entrada do problema;
- Vértices pertencentes ao conjunto W não podem compor a rota;

- T é o subconjunto dos vértices V que são definidos como obrigatórios, e, devem, portanto, ser necessariamente visitados. Os demais $(V \backslash T)$ são chamados de opcionais; e

- Existe uma matriz simétrica de distâncias c_{ij}, com $i, j \in (V \cup W)$, definida sobre o conjunto de arestas E, satisfazendo a desigualdade triangular.

Dadas essas restrições, o PRR consiste em encontrar uma rota ou ciclo hamiltoniano de comprimento mínimo utilizando um subconjunto dos vértices V. Todos os vértices do subconjunto $T \subseteq V$ devem ser visitados e nenhum vértice W poderá ficar descoberto. Para isso, se necessário, vértices opcionais $(V \backslash T)$ podem pertencer à rota.

Pode-se reduzir o PRR ao PCV. Para isso, basta que:

1. Todos os vértices V sejam obrigatórios, $V = T$; e,

2. Não existam vértices a serem cobertos, $W = \emptyset$.

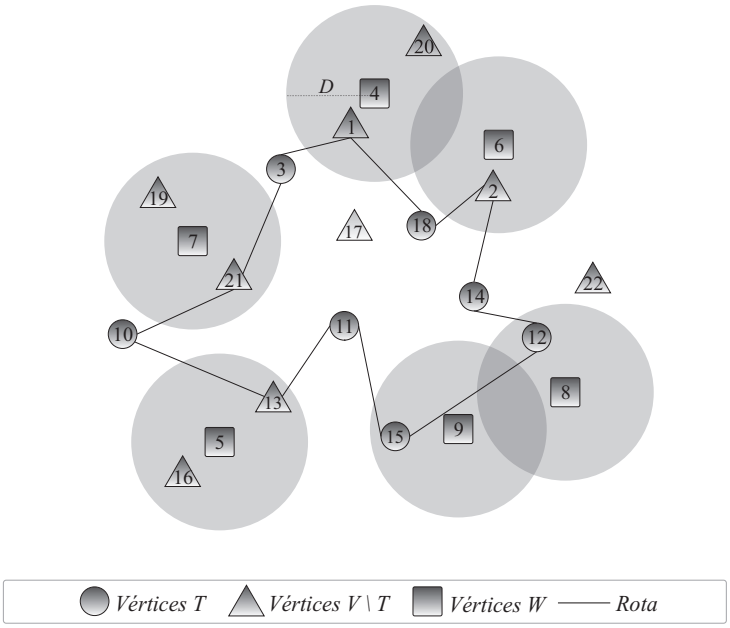

Figura 2.1: Rota construída sobre uma instância do PRR.

Como PCV é NP-difícil, é possível provar PRR também o é. Dessa forma, abordagens exatas podem ser muito custosas, devido à explosão combinatória das variáveis e restrições envolvidas.

Na Figura 2.1, os vértices obrigatórios T, os opcionais $(V \backslash T)$ e os de cobertura W, estão representados, respectivamente, por círculos, triângulos e quadrados. Pode-se ainda

observar uma área circular sombreada ao redor dos vértices W de raio D, onde D é a distância de cobertura, conforme é demonstrado no vértice 4. De acordo com as restrições do PRR, pelo menos um vértice de V, que compõe a rota, deve pertencer a áreas sombreadas. Observe ainda que, de acordo as restrições do PRR, só faz sentido a entrada de um vértice $i \in (V \backslash T)$ na rota se i fizer cobertura exclusiva de algum vértice $j \in W$. Dessa forma, os vértices 17 e 22, jamais farão parte de uma solução ótima. Já os vértices $4, 5$ e 7, pertecentes a W, podem ser cobertos por mais de um vértice $i \in (V \backslash T)$ cada um. Nesses casos é necessária a escolha do vértice de melhor custo para solução.

O problema de recobrimento de rotas com coletas de prêmios é uma variação do PRR e, devido a isso, contém todas as suas restrições. Além dessas, uma nova restrição relacionada ao $PRIZE$ é incluída. Assim, cada vértice $i \in V$ tem um valor-prêmio não-negativo, $p_i \geq 0$, associado. Em qualquer instância do PRRCP, o somatório dos prêmios dos vértices V é obrigatoriamente maior ou igual a $PRIZE$ [16].

O PRRCP consiste em encontrar uma rota-solução S sobre o conjunto vértices V que seja um ciclo hamiltoniano de comprimento mínimo que:

- Todos os vértices $T \subseteq V$ estejam contidos em S;
- Exista em S pelo um vértice $i \in V$, cuja distância $c_{ij} \leq D$, para todo vértice $j \in W$; e
- O somatório dos prêmios coletados deve ser maior ou igual ao $PRIZE$, ou seja, $\sum_{i \in S} p_i \geq PRIZE$.

Observa-se na Figura 2.2, uma solução para o PRRCP. Nela, além das figuras geométricas ilustrando os tipos de vértices, foram incluídos os prêmios $p_i \geq 0$ em cada vértice $i \in V$, posicionado ao lado de cada vértice (em sublinhado). Na instância apresentada no exemplo da Figura 2.2, o somatório dos prêmios p_i de todos os $i \in V$ é de 145 unidades. No entanto, exige-se a coleta de prêmios, dado por $PRIZE$, de 110 unidades. A construção de um rota atendendo somente as restrições do PRR, isto é, vértices obrigatórios e cobertura dos vértices W, alcança 84 unidades. Entretanto, para o PRRCP, essa rota ainda é inviável, pois a nova restrição de prêmio não foi atendida. Sendo assim, é necessário coletar, pelo menos, mais 26 unidades. Por causa disso, foi necessário incluir mais três vértices do conjunto $V \backslash T$, os vértices $17, 22$ e 24, totalizando exatamente o valor requerido pelo $PRIZE$ de 110 unidades.

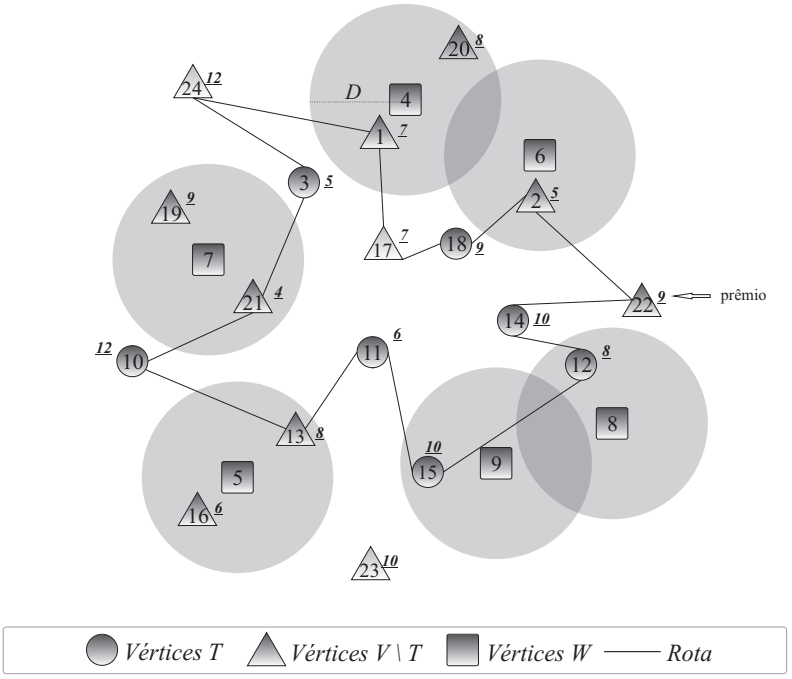

Figura 2.2: Rota construída sobre um grafo instância do PRRCP.

Observa-se, portanto, que além dos vértices obrigatórios T, alguns vértices opcionais $V \backslash T$ foram necessariamente incluídos na solução. Os vértices $1, 2, 13$ e 21 por cobrirem vértices W ainda descobertos, mesmo com todos os vértices obrigatórios já visitados e, os vértices $17, 22$ e 24, também do subconjunto $V \backslash T$, pois seus prêmios são necessários para se atingir a meta de coleta de prêmios prefixada.

2.2 Revisão de Literatura

Nesta seção são relatados os trabalhos encontrados na literatura que tiveram o PRRCP como objeto de estudo.

Embora exista uma grande variedade de possíveis aplicações em problemas reais da sociedade, o PRRCP só foi investigado em duas pesquisas desde que fora proposto. O primeiro é a dissertação de Lyra [16], trabalho no qual o PRRCP foi proposto como uma variação do PRR. Formulação matemática, regras de redução e um algoritmo aproximado para resolução do problema foram apresentados no trabalho. O segundo é a dissertação de Silva [35], onde foram propostos seis algoritmos heurísticos, cinco baseados na metaheurística *Iterated Local Search* (ILS) [15] e outro em algoritmos evolutivos [36] para resolução do PRRCP, além de uma nova formulação matemática e três novas regras de redução.

Na dissertação de Lyra [16] uma revisão da literatura sobre PRR foi realizada, desde que o problema foi introduzido em 1981 [4]. É nessa dissertação que o PRRCP foi proposto como variação do PRR, por meio da inclusão da restrição relacionada ao prêmio. Baseado nas formulações matemáticas encontradas em [11], Lyra propôs uma formulação matemática para resolução exata do PRRCP.

Em [16] as regras de redução aplicáveis ao PRR disponíveis na literatura foram apresentadas e analisadas, assim como adaptações dessas regras para que elas pudessem ser aplicadas a instâncias do PRRCP. Uma nova regra de redução foi proposta. Para resolução aproximada do PRRCP foi descrito um algoritmo heurístico baseado no método GRASP combinado com a uma variação da metaheurística *Variable Neighbourhood Search* (VNS) [18]. Foi incorporada também a técnica de reconexão de caminhos [12], como elemento intensificador de busca de soluções no conjunto das melhores soluções, chamado de conjunto de elite, encontradas ao longo das iterações da heurística GRASP.

Os experimentos foram realizados sobre instâncias geradas aleatoriamente em um plano cartesiano 100 x 100, e foram executados em um processador de 2Ghz e 1GB de memória RAM. O algoritmo encontrou a solução ótima para a maioria dos casos onde o ótimo era conhecido [16].

Na dissertação de Silva [35] uma nova formulação matemática, três novas regras de redução e a definição de uma ordem de aplicação dessas regras, combinadas com mais quatro outras já introduzidas em [16], foram apresentadas.

Seis algoritmos heurísticos para encontrar soluções de boa qualidade para o PRRCP foram propostos. Cinco deles foram versões de algoritmos baseados na metaheurística *Iterated Local Search* (ILS), combinadas com cinco diferentes métodos de geração de solução inicial (*ADD*, *DROP*, inserção mais próxima, inserção mais barata e GENIUS[10]) e o sexto algoritmo utilizou as técnicas populacionais de algoritmos evolutivos. A técnica de reconexão de caminhos foi aplicada como estratégia de intensificação do algoritmo populacional.

Para realização dos experimentos computacionais em [35], dois conjuntos de problemas-testes foram gerados para o PRRCP, a partir das instâncias do repositório TSPLIB [28]. O primeiro, denominado Grupo 1, contém instâncias com até 200 vértices. Já o segundo, conhecido como Grupo 2, contém instâncias de 201 até, no máximo, 400 vértices. Esses problemas-teste também foram utilizados para avaliar a qualidade do algoritmo proposto nesta dissertação.

Tanto os algoritmos baseados em ILS quanto o algoritmo evolutivo encontraram o melhor valor conhecido para a maioria das instâncias criadas [35]. Para os problemas-testes do Grupo 1, os algoritmos ILS encontraram o melhor valor conhecido em mais de 95%, em média, do total de instâncias. Já para as instâncias do Grupo 2, a melhor versão do algoritmo ILS obteve taxa de sucesso de 76% dos casos. Enquanto o algoritmo evolutivo obteve taxa de sucesso de 99,1% para os problemas-teste do Grupo 1 e de 91% para as instâncias do Grupo 2. No entanto, para as instâncias do Grupo 2, o algoritmo evolutivo utilizou, em média, o dobro do tempo para alcançar o mesmo valor obtido pelo ILS. De acordo com [35], esse fato foi devido à aplicação do método de reconexão de caminhos a soluções de baixa qualidade oriundas da busca local, que ainda encontravam-se distantes das soluções usadas como guias do conjunto elite, exigindo-se, portanto, maior tempo de execução.

2.2.1 Formulações Matemáticas

Nesta seção as formulações matemáticas propostas em [16, 35] para resolver o PRRCP são descritas.

Em [16] um modelo de programação linear inteira para o PRRCP foi proposto com uso de variáveis de fluxo, a fim de evitar a formação de ciclos desconexos do vértice de origem.

Seja $G = (N, E)$ um grafo completo e não direcionado, onde $N = (V \cup W)$. Para cada aresta (i, j) uma variável inteira não-negativa z_{ij} é associada, a fim de representar a quantidade de fluxo no arco (i, j). Para cada vértice $k \in V$, tem-se uma variável binária y_k, para indicar se o vértice k está na solução. Os vértices que não pertencem a solução assumem valor zero, isto é, $y_k = 0$. Caso contrário, $y_k = 1$. Para representar as arestas (i, j) que pertencem a rota, foi definida uma variável binária x_{ij} a qual terá o valor um, se e somente se, a aresta (i, j) pertence a rota e assumirá o valor zero, caso contrário. O custo de deslocamento do vértice i até j é dado por c_{ij}. O prêmio de cada $k \in V$ é dado por p_k. E, finalmente, $PRIZE$ é o somatório mínimo de prêmios exigido como coleta na rota.

Essa formulação pode ser descrita como um problema de programação linear inteira da seguinte forma [16]:

$$Minimizar \sum_{i<j|i,j\in V} c_{ij}x_{ij}, \tag{2.1}$$

Sujeito a:

$$\sum_{k \in V} p_k y_k \geq PRIZE, \tag{2.2}$$

$$\sum_{k \in S_l} y_k \geq 1 \quad (\forall l \in W), \tag{2.3}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2y_k \quad (\forall k \in V), \tag{2.4}$$

$$\sum_{i \in V} z_{kj} = \sum_{i \in V} z_{ik} + y_k \quad (\forall k \in V - \{s\}), \tag{2.5}$$

$$\sum_{j \in V} z_{sj} = 1, \tag{2.6}$$

$$\sum_{j \in V} z_{js} = \sum_{j \in V} y_j, \tag{2.7}$$

$$x_{ij} \leq z_{ij} \quad (\forall i, j \in V), \tag{2.8}$$

$$x_{ij} \geq z_{ij}/(|V| + 1) \quad (\forall i, k \in V), \tag{2.9}$$

$$y_k = 1 \quad (\forall k \in T), \tag{2.10}$$

$$y_k \in \{0, 1\} \quad (\forall k \in V \backslash T), \tag{2.11}$$

$$x_{ij} \in \{0, 1\} \quad (\forall i, j \in V), \tag{2.12}$$

$$z_{ij} \in Inteiros \quad (\forall i, j \in V). \tag{2.13}$$

A função objetivo da formulação para o PRRCP, que é minimizar o custo da rota gerada, é apresentada em (2.1). Na inequação (2.2), existe a garantia da coleta de prêmio, ou seja, o somatório dos prêmios p_k dos $k \in S$ alcance o $PRIZE$ estabelecido. A cobertura de todos os vértices $l \in W$ é assegurada pela restrição (2.3), onde S_l é o conjunto de todos vértices $k \in V$ que cobrem $l \in W$. A restrição (2.4) garante a conectividade da rota, ou seja, todo vértice da rota deve ter grau dois. As restrições (2.5) a (2.7) têm por objetivo evitar ciclos desconexos. As restrições (2.8) e (2.9) asseguram que as rotas geradas a partir das variáveis de fluxo z_{ij} e das variáveis binárias x_{ij} sejam as mesmas. A restrição (2.10) tem por objetivo garantir que todos os vértices $k \in T$ estejam na rota. As demais restrições, (2.11), (2.12) e (2.13), informam sobre o domínio das variáveis y_k, x_{ij} e z_{ij}, respectivamente, ou seja, os valores possíveis que essas variáveis podem assumir.

Se a restrição (2.2) for excluída dessa formulação, então obtém-se uma formulação para o PRR. Considerando $n = |V \cup W|$, $p = |V|$ e $m = |E|$, a quantidade restrições dessa formulação é da ordem de $O(n^2)$ e o número de variáveis geradas é $(2m + p)$.

Já o modelo de programação de linear inteira proposto em [35] usa variáveis de fluxo

multiproduto para evitar a criação de ciclos desconexos na solução. Esse tipo de variável, usada nessa formulação proposta, foi utilizado por [3, 37] para resolução do problema do caixeiro viajante.

As variáveis dessa formulação são as seguintes: (a) x_{ij}, variável binária, que assume valor um, se o arco (i, j) estiver presente na rota, e valor zero, caso contrário; (b) z_k, também variável binária, para indicar se o vértice k está na solução, isto é, para todo vértice $k \in T$, $z_k = 1$; e (c), y_{ij}^k, variável inteira não-negativa que representa a quantidade de fluxo do produto k escoado pela aresta $(i, j) \in E$. Além disso, o custo de deslocamento do vértice i até j é dado por c_{ij}. O prêmio de cada $k \in V$ é dado por p_k. $PRIZE$ é o somatório mínimo de prêmio requerido na instância-problema.

Essa formulação matemática pode ser descrita da seguinte forma [35]:

$$Minimizar \sum_{i,j \in V, i \neq j} c_{ij} x_{ij} \tag{2.14}$$

Sujeito a:

$$\sum_{j \in V, j \neq i} x_{ij} = z_i \quad (\forall i \in V), \tag{2.15}$$

$$\sum_{i \in V, i \neq j} x_{ji} = z_j \quad (\forall j \in V), \tag{2.16}$$

$$y_{ij}^k \leq x_{ij} \quad (\forall i, j, k \in V), \tag{2.17}$$

$$\sum_{i \in V} y_{1i}^k = z_k \quad (\forall k \in V \backslash \{s\}), \tag{2.18}$$

$$\sum_{i \in V} y_{i1}^k = 0 \quad (\forall k \in V \backslash \{s\}), \tag{2.19}$$

$$\sum_{i \in V} y_{ik}^k = z_k \quad (\forall k \in V \backslash \{s\}), \tag{2.20}$$

$$\sum_{j \in V} y_{kj}^k = 0 \quad (\forall k \in V \backslash \{s\}), \tag{2.21}$$

$$\sum_{i \in V} y_{ij}^k - \sum_{i \in V} y_{ji}^k = 0 \quad (\forall k, j \in V \backslash \{s\}, j \neq k), \tag{2.22}$$

$$\sum_{i \in V} p_i z_i \geq PRIZE, \tag{2.23}$$

$$\sum_{i \in S_i} z_i \geq 1 \quad (\forall l \in W), \tag{2.24}$$

$$z_i = 1 \quad (\forall i \in T), \tag{2.25}$$

$$z_i \in \{0,1\} \quad (\forall i \in (V \backslash T)), \tag{2.26}$$

$$x_{ij} \in \{0,1\} \quad (\forall i,j \in V), \tag{2.27}$$

$$y_{ij}^k \in InteirosPositivos \quad (\forall i,j,k \in V). \tag{2.28}$$

No modelo supracitado, a função objetivo dada em (2.14) minimiza o custo total da rota. As restrições (2.14) e (2.16) garantem a consistência entre os vértices incluídos na rota e a existência de arestas entre eles. A restrição dada em (2.17) limita a passagem de fluxo de produtos a arestas da solução e as restrições (2.18) informam que somente será enviado produto da origem 1 para clientes que estejam na rota, e as (2.19) impedem o retorno de produtos a origem. As restrições (2.20) e (2.21) asseguram que todo vértice receberá seu produto correspondente, e esse produto não deverá ser enviado a nenhum outro vértice. Já as restrições (2.22) garantem a conservação do fluxo dos produtos que ainda não alcançaram seus vértices de destino. A restrição $PRIZE$ está expressa em (2.23). As restrições (2.24), garantem que pelo menos um vértice $i \in S_l, \forall l \in W$, esteja na rota, onde S_l é o subconjunto de vértices V que cobrem l, garantindo, assim, a premissa de cobertura do PRRCP. Em (2.25) há a imposição de que todos vértices obrigatórios T façam parte da rota. As demais restrições, de (2.26) a (2.28), representam os valores que cada variável da formulação poderá assumir.

O número de restrições e de variáveis dessa formulação é $O(n^3)$ [35].

2.2.2 Regras de Redução

Um dos principais gargalos para a resolução de problemas de otimização combinatória é o tamanho de suas entradas [16]. Quando se reduz espaço de busca por soluções, consequentemente, o esforço computacional requerido para se chegar a um ótimo global também é minimizado. Dessa forma, regras de redução têm por objetivo retirar das instâncias elementos que possam diminuir o espaço de busca, porém sem prejudicar a obtenção de soluções ótimas.

Esta seção apresenta as regras de redução existentes na literatura para o PRRCP.

Devido à natureza do PRRCP, as regras de redução podem atuar das seguintes formas sobre as instâncias:

- transformando vértices opcionais $(V \backslash T)$ em vértices obrigatórios $(T \subseteq V)$, pelo

fato de cobrir exclusivamente algum vértice W, ou pelo fato de seu prêmio ser estritamente necessário para coleta do $PRIZE$ requerido;

- excluindo vértices do conjunto do vértices opcionais ($V \backslash T$), caso seu prêmio e sua cobertura sejam facilmente atendidas pelos demais vértices opcionais; e
- removendo vértices do conjunto W, se sua cobertura já é garantida de alguma forma, ou seja por vértices obrigatórios T, ou seja por vértices opcionais ($V \backslash T$) com prêmios indispensáveis.

Considerando as regras de redução introduzidas por [11], posteriormente aplicado ao PRR por [17], Motta em [21] consolidou quatro regras de redução para o PRR. Das quatro regras resultantes de [21], Lyra em [16] adaptou, ou usou diretamente, três regras de redução e propôs uma regra específica para o PRRCP, considerando a nova restrição do problema.

A investigação realizada em [35] utiliza o conjunto de regras consolidado por [16] e o complementa com mais três novas regras de redução aplicáveis ao PRRCP. Os dois trabalhos mostraram ainda que a ordem de aplicação das regras pode ter impacto na redução do espaço de busca.

Considere que cada vértice $i \in W$ tem um conjunto de vértices V que o cobre, ou seja, $S_i = \{j \in V : d_{ij} \leq D, i \in W\}$ e que para cada vértice $j \in V$, existe um conjunto de vértices W que ele pode cobrir caso esteja na rota, dado por $Cob_j = \{i \in W : d_{ij} \leq D, j \in V\}$. Seja uma variável binária, $flag$, que indica a necessidade de algum vértice $V \backslash T$ estar presente na solução e uma outra variável binária ε_{ij} que representa se um dado vértice i, pertencente a W, é coberto pelo $j \in V$, isto é, $\varepsilon_{ij} = 1$ indica que i está coberto por j, e $\varepsilon_{ij} = 0$, caso contrário.

O conjunto de regras associadas ao PRRCP por ordem de aplicação [35], contém as seguintes regras:

- **R1** - para todo vértice $j \in V \backslash T$, para todo vértice $i \in W$, se $|S_i| = 1$ e $\varepsilon_{ij} = 1$, converter j a um vértice obrigatório, isto é, $j \in T$ [16];
- **R2** - se ao remover j, para todo vértice $j \in V \backslash T$, do conjunto, e a soma dos prêmios do demais vértices V não atingir o $PRIZE$ requerido, converter j em um vértice obrigatório de T [35];

- **R3** - Se o somatório dos prêmios dos vértices T for suficiente para atingir o $PRIZE$, ou seja, $\sum_{i \in T} p_i \geq PRIZE$, remover todo vértice $j \in V \backslash T$ em que $|Cob_j| = 0$;
- **R4** - Se $\sum_{i \in T} p_i \geq PRIZE$, para todo $i \in W$, e se existir um vértice $j \in V \backslash T$ e um vértice $k \in T$ tal que $\varepsilon_{ij} = 1$ e $\varepsilon_{ik} = 1$ e se o vértice j cobrir somente o vértice i ou todos os vértices cobertos por j também forem cobertos por um vértice de T, remover j de $V \backslash T$ [35];
- **R5** - Para todo $i, j \in V \backslash T, i \neq j$, se $\sum_{k \in V - i} p_k \geq PRIZE$, e $Cob_i \subseteq Cob_j$ e $d_{ik} > d_{jk}$, $\forall k \in V, k \neq i$, remover i de $V \backslash T$ [35];
- **R6** - Remover $i \in W$, se existir $j \in T$, tal que $\varepsilon_{ij} = 1$ [11]; e
- **R7** - Remover $i \in W$, se para todo $j \in V \backslash T, \varepsilon_{ij} = 1$ e atribuir valor um à variável $flag$ [16];

A ordem de aplicação das regras de redução considera que primeiramente é necessário definir o conjunto dos vértices obrigatórios T, isto é, verificar quais os vértices $V \backslash T$ devem ser convertidos em obrigatórios T (Regras R1 e R2), visto que as demais regras utilizam o conjunto de vértices T como parâmetro para reduzir o espaço de busca do problema. O passo seguinte é verificar dentre os $V \backslash T$ restantes quais são dispensáveis para a resolução do problema, ou seja, quais vértices $i \in V \backslash T$ podem ser descartados sem prejudicar a obtenção de uma solução ótima, seja pelo prêmio ou pela sua função de cobertura (Regras R3, R4 e R5).

Por fim, com os conjuntos T e $V \backslash T$ devidamente definidos, é possível verificar quais vértices do conjunto W já tem sua cobertura garantida. Ou seja, para qualquer solução obtida, algum vértice V necessariamente satisfará a sua restrição de cobertura e, portanto, esses vértices de Wpodem ser também desconsiderados, reduzindo, assim, o esforço computacional para atendimento da premissa de cobertura (Regras R6 e R7).

No Capítulo 3 é apresentada uma revisão bibliográfica sobre técnicas usadas na resolução aproximada de problemas de otimização combinatória que são utilizadas neste trabalho para encontrar ótimos locais para o PRRCP.

Capítulo 3

Heurísticas e Metaheurísticas

Neste capítulo é feita uma revisão de literatura a cerca das técnicas utilizadas neste trabalho para atingir o objetivo geral estabelecido. Heurísticas construtivas e de refinamento e a metaheurística GRASP são apresentadas ao longo deste capítulo.

3.1 Heurísticas

Heurísticas são procedimentos, alternativos aos método exatos, para a resolução aproximada de problemas de otimização combinatória. Ou seja, elas obtêm soluções de boa qualidade em um tempo computacional aceitável, em contraste com abordagens exatas que podem requerer tempos de computação elevados. Entretanto, as heurísticas não têm o compromisso com a otimalidade da solução.

Heurísticas são projetadas para tratar problemas específicos, diferentemente das metaheurísticas, que são procedimentos genéricos que podem aplicar a cada passo heurísiticas subordinadas e podem ser utilizadas para propor algoritmos específicos. De acordo com [32], adaptação de uma determinada metaheurística a um problema resulta em uma heurística resolver o último.

As heurísticas são comumente classificadas em dois grupos, a saber: heurísticas construtivas e heurísticas de refinamento. As construtivas têm por objetivo gerar uma solução viável, comumente, elemento a elemento, para um problema específico. A cada elemento candidato a entrar na solução é atribuído um valor benefício (ou prejuízo). Assim, a escolha de cada novo item é avaliada por uma função objetivo levando-se em consideração as especificidades do problema em questão e o valor benefício de cada candidato.

Por outro lado, as heurísticas de refinamento, também chamadas de heurísticas de

busca local, realizam movimentos sobre uma solução inicial S_0 já existente, provida por uma heurística construtiva. A cada movimento realizado usando-se uma vizinhança bem definida em S_0, um outra solução S' é gerada. Dessa forma, heurísticas de busca local são baseadas no conceito de vizinhança. De acordo com [32], a vizinhança N_i de uma solução S pode ser definida como um conjunto de soluções que diferem de S por i atributos. No caso do PRRCP, por exemplo, uma solução $S^{'}$ pertence à vizinhança de uma solução S se dois de seus vértices têm suas posições invertidas em relação a rota original S.

Em alguns casos, heurísticas podem percorrer vizinhanças gerando vizinhos inviáveis. Nessas situações, movimentos que gerem inviabilidade de soluções são admitidos, por meio do relaxamento de algumas restrições que compõem o problema. Contudo, é acrescentado o conceito de penalidade à função objetivo, onde a cada violação a uma restrição relaxada, é atribuído uma valor penalidade.

3.1.1 Heurísticas Construtivas

3.1.1.1 Inserção Mais Barata

A heurística construtiva inserção mais barata [29] gera uma solução a partir de um conjunto de vértices disponíveis, uma matriz de custos, ou distâncias, c_{ij}, e uma rota parcial contendo um par de vértices escolhidos aleatoriamente. Uma função de avaliação de custo C é usada para encontrar, dentre todos os vértices candidatos, ou seja, os que ainda não pertencem a solução, qual é o vértice que possui o menor custo de inserção na rota e em que posição ele deve ser incluído na solução. A inserção mais barata considera a inclusão somente entre pares vértices que sejam adjacentes.

Para um problema de minimização, a função de avaliação de custo, utiliza a rota parcial e a matriz c_{ij} para identificar o vértice que incrementará o custo dessa rota com um menor valor. Por exemplo, considerem i e j vértices adjacentes na rota e k o vértice candidato. Assim, o custo C_{ij}^{k}, o custo de inserir k entre i e j, é dado pelo somatório dos custos das arestas que deverão ser inseridas na solução que conectarão o vértice i ao k e o vértice k ao j, subtraído o custo da aresta que não pertencerá mais a rota, ou seja, a aresta (i, j). Assim, a função de avaliação de custo é dada por $C_{ij}^{k} = (c_{ik} + c_{kj}) - c_{ij}$.

O Algoritmo 1 contém o pseudocódigo da heurística construtiva inserção mais barata para um problema de criação de rota genérico. Na linha 1, seleciona-se aleatóriamente dois vértices da lista de candidatos (LC). Em seguida, na linha 2, constrói-se uma sub-rota com os dois vértices, obtidos já obtidos. Inicialmente, LC é formada por todos os

Algoritmo 1 Inserção Mais Barata

1: $i, j \leftarrow SelecaoAleatoria(LC)$;
2: $S \leftarrow \{i - j\}$;
3: $LC \leftarrow LC - \{i, j\}$
4: **enquanto** *(condição de parada não estiver satisfeita)* **faça**
5: $C \leftarrow DefinirCustos(LC)$;
6: $k \leftarrow SelecaoMenorCusto(C)$;
7: $S \leftarrow S \cup k$;
8: $LC \leftarrow LC \backslash k$
9: **fim enquanto**
10: **retorne** S;

vértices aptos a fazer parte da rota S. De acordo com o problema em questão, enquanto a rota não estiver completa, é calculado para cada vértice $k \in LC$, seu melhor custo de inserção, bem como a posição, ou seja, entre quais vértice i e j adjacentes, k deve ser inserido (Linha 5). A cada iteração, obrigatoriamente, um novo vértice k é inserido na rota S usando-se essa estratégia puramente gulosa (Linhas 6 a 8).

No Capitulo 4 são descritas as estruturas de vizinhança que utilizam o algoritmo de inserção mais barata adaptada ao PRRCP, usadas durante o processo de busca local.

3.1.1.2 GENIUS

Proposta em 1992 por [10] para solucionar o problema do caixeiro viajante, a heurística GENIUS é composta de duas etapas, a saber: um procedimento de construção, a fase GENI *(Generalized Insertion)* e um procedimento de melhoria, a fase US *(Unstringing and Stringing)*. A fase GENI possui dois métodos de inserção de vértices. Analogamente, na fase US existem dois métodos de remoção de vértices da rota. Em [10] foi ponderado que no momento da construção, isto é, na fase GENI, ainda não existia uma noção de otimização global, que resultava em uma sucessão decisões desfavoráveis que uma fase de pós-otimização, a fase US, podia desfazê-las ou corrigí-las. O Algoritmo 2 mostra o pseudocódigo do algoritmo da heurística GENIUS.

GENIUS, diferentemente da heurística inserção mais barata, parte de uma rota parcial contendo três vértices da lista de candidatos. Com isso, enquanto não for construída a rota S, ou seja, enquanto o critério de parada não for atendido, de acordo com o problema que está sendo abordado, a cada iteração um novo vértice k qualquer em LC é selecionado e enviado ao procedimento GENI para inserção na melhor posição, isto por que o vértice k não será necessariamente inserido entre vértices adjacentes. Além disso, ao finalizar a construção da rota, a fase de pós-otimização US faz uma busca na solução excluindo cada

Algoritmo 2 GENIUS

1: $i, j, k \leftarrow SelecaoAleatoria(LC)$;
2: $S \leftarrow \{i - j - k\}$;
3: $LC \leftarrow LC - \{i, j, k\}$
4: **enquanto** *(condição de parada não estiver satisfeita)* **faça**
5: $v \leftarrow SelecaoSimples(LC)$;
6: $S \leftarrow GENI(S, v)$; *//insere v em S*
7: $LC \leftarrow LC \backslash v$;
8: **fim enquanto**
9: $S^* \leftarrow US(S)$; *//melhoria da rota*
10: **retorne** S^*;

vértice $k \in S$, e o reinserindo em uma nova posição, se possível, a fim de melhorar o custo da solução construída.

Generalized Insertion Procedure (GENI)

GENI inicia com uma rota contendo três vértices selecionados aleatoriamente entre os candidatos. Em seguida, a cada iteração, um novo vértice v é inserido na rota parcial por meio de uma re-otimização local na rota. Essa otimização é realizada usando a melhor combinação entre v e os p vértices na rota S mais próximos de v, denotados por $N_p(v)$.

De acordo com [10], a principal característica de GENI é que a inserção de um vértice v na rota não acontece, necessariamente, entre dois vértices adjacentes. Entretanto, após a inserção esses dois vértices passam a ser adjacentes a v.

Para um melhor entendimento dos dois tipos de inserção de GENI, suponha que se deseja inserir o vértice v entre quaisquer dois vértices v_i e v_j da rota. Dada uma orientação da rota, isto é, sentidos horário e anti-horário, considere v_k um vértice no caminho de v_j até v_i, e v_l um vértice no caminho v_i até v_j. Considere ainda, que dado um vértice v_h, em qualquer da rota, v_{h-1} é seu antecessor e v_{h+1} o sucessor. Com essas definições dadas, a inserção do vértice v entre v_i e v_j poderá ser feita de duas formas, como explicado a seguir em inserção Tipo 1 e Tipo 2, devidamente ilustradas nas Figuras 3.1 e 3.2.

GENI - Inserir Tipo 1: Antes de inserir v, retiram-se as arestas (v_i, v_{i+1}), (v_j, v_{j+1}) e (v_k, v_{k+1}) que serão substituídas por quatro novas arestas (v_i, v), (v, v_j), (v_{i+1}, v_k) e (v_{j+1}, v_{k+1}), onde o vértice k deve ser diferente de i e j. Devido ao fato de que em GENIUS a rota tem uma orientação, essas modificações implicarão a mudança de sentido dos caminhos $(v_{i+1}, ..., v_j)$ e $(v_{j+1}, ..., vk)$.

GENI - Inserir Tipo 2: Inicialmente, é necessário verificar que $v_k \neq v_j$, $v_k \neq v_{j+1}$, $v_l \neq v_i$ e $v_l \neq v_{i+1}$. A inserção de v na rota resulta da remoção das arestas (v_i, v_{i+1}),

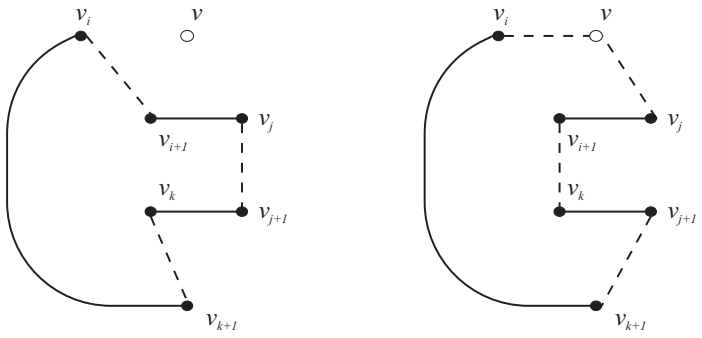

Figura 3.1: GENI - Inserção Tipo 1 de vértice v entre v_i e v_j

(v_{l-1}, v_l), (v_j, v_{j+1}) e (v_{k-1}, v_k) seguido da inserção das arestas (v_i, v), (v, v_j), (v_l, v_{j+1}), (v_{k-1}, v_{l-1}) e (v_{i+1}, v_k). Analogamente ao que ocorre em *Inserir Tipo 1*, os caminhos $(v_{i+1}, ..., v_{l-1})$ e $(v_l, ..., v_j)$ devem ser invertidos para preservação do sentido rota.

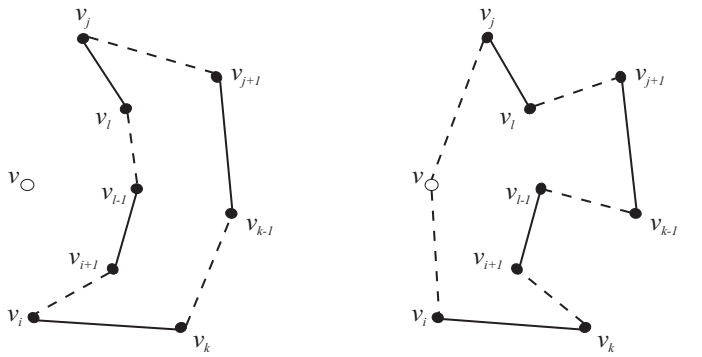

Figura 3.2: GENI - Inserção Tipo 2 de vértice v entre v_i e v_j

No algoritmo da fase GENI, considerando-se as duas orientações, as combinações de v_i, v_j, v_k e v_l é de $O(n^4)$. Para limitar as possibilidades de cada uma dessas variáveis, para cada vértice $v \in V$, definem-se p vizinhos de v, dado por $N_p(v)$, onde $N_p(v)$ representa os p vértices que já pertencem a rota que estão mais próximos de v, de acordo com a matriz de distância c_{ij}. Assim, para um dado valor de p, primeiro seleciona-se $v_i, v_j \in N_p(v)$, depois $v_k \in N_p(v_{i+1})$ e $v_l \in N_p(v_{j+1})$. A inserção do vértice v pode ocorrer entre todos os vértices adjacentes do conjunto $N_p(v)$. De acordo com [10], p é, na prática, um número relativamente pequeno.

O Algoritmo 3 ilustra o pseudocódigo da fase GENI, onde dada uma rota parcial S e um vértice v aleatório a ser inserido, o primeiro procedimento é criar o conjunto N_p para cada um dos vértices do problema, esteja ou não em S (Linha 1). Esse conjunto é

Algoritmo 3 GENIUS: Fase GENI

Entrada: S, v;
1: $N_p \leftarrow DefinirVerticesProximos(V)$;
2: $f^* \leftarrow \infty$;
3: **para** $v_i, v_j \in N_p(v), v_i \neq v_j$ **faça**
4: **para** $sentido \in \{horario, anti-horario\}$ **faça**
5: $S' \leftarrow inserir\_tipo1(v_i, v_j, S, v, sentido)$
6: **se** $f(S') < f^*$ **então**
7: $S^* \leftarrow S'$;
8: $f^* \leftarrow f(S^*)$
9: **fim se**
10: $S'' \leftarrow inserir\_tipo2(v_i, v_j, S, v, sentido)$
11: **se** $f(S'') < f^*$ **então**
12: $S^* \leftarrow S''$;
13: $f^* \leftarrow f(S^*)$
14: **fim se**
15: **fim para**
16: **fim para**
17: **retorne** S^*;

utilizado na reorganização da solução após a inserção do vértice v, pois conforme pode ser observado nas Figuras 3.1 e 3.2 e nos procedimentos *Inserir Tipo 1 e Tipo 2*, cada vértice v_i, v_j, v_k e v_l (e seus antecessores e/ou sucessores), v é sempre conectado a um dos vértices da rota que estão em seu conjunto N_p. Em seguida, para cada orientação (sentido horário/anti-horário) as duas formas de inserção GENI são testadas. Finalmente, retorna-se a rota de melhor custo.

Unstringing and String (US)

A fase US, conforme já descrito, tem por objetivo refinar a rota recém construída pela fase GENI. Para isso, remove-se, iterativamente, cada um dos vértice de S de duas formas diferentes e cada um deles é reinserido por meio do uso dos métodos de inserção de GENI. As Figuras 3.3 e 3.4 ilustram essas duas maneiras de remoção.

No Algoritmo 4, o pseudocódigo da fase US é ilustrado. Para cada vértice $v_i \in S$ e os sentidos da rota, os dois tipos de remoção são testados. Em cada tipo de remoção de US, após a exclusão do vértice v_i, os dois procedimentos de inserção da fase GENI são usados para reinserir o vértice v_i na rota S. Dentre todas as rotas parciais geradas com custo inferior ao de S, resultantes das combinações de vértice v_i, a rota de menor custo é retornada como resultado do processo de pós-otimização US. A seguir os dois tipos de remoção de US são descritos e ilustrados nas Figuras 3.3 e 3.4.

US - Remover Tipo 1: Sejam $v_j \in N_p(v_{i+1})$, e $v_k \in N_p(v_{i-1})$ um vértice no caminho

Algoritmo 4 GENIUS: Fase US

Entrada: S;

```
1: S* ← S;
2: f* ← f(S*);
3: para (v_i ∈ S) faça
4:     para (sentido ∈ {horario, antihorario}) faça
5:         S' ← remover_tipo1(v_i, S, sentido)
6:         se (f(S') < f*) então
7:             S* ← S';
8:             f* ← f(S*)
9:         fim se
10:        S'' ← remover_tipo2(v_i, S, sentido)
11:        se (f(S'') < f*) então
12:            S* ← S'';
13:            f* ← f(S*)
14:        fim se
15:    fim para
16: fim para
17: retorne S*;
```

$(v_{i+1}, .., v_{j-1})$ para uma dada orientação. Para se excluir o vértice v_i da rota, deve-se então remover as arestas (v_{i-1}, v_i), (v_i, v_{i+1}), (v_k, v_{k+1}) e (v_j, v_{j+1}). Logo em seguida, a rota é reconectada por meio das seguintes arestas, (v_{i-1}, v_k), (v_{i+1}, v_j) e (v_{k+1}, v_{j+1}). Ainda devem-se inverter os caminhos $(v_{i+1}, ..., v_k)$ e $(v_{k+1}, ..., v_j)$.

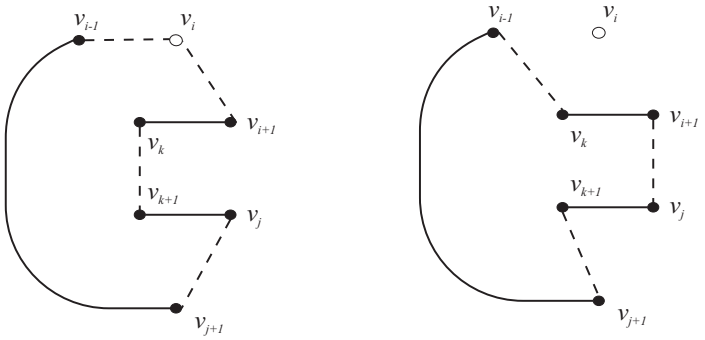

Figura 3.3: US - Remoção Tipo 1 de vértice v_i da rota

US: Remover Tipo 2: Na remoção Tipo 2 de US, considere $v_j \in N_p(v_{i+1})$ e v_k um vértice do caminho $(v_{j+1}, ..., v_{i-2})$, tal que $v_k \in N_p(v_{i-1})$, e $v_l \in N_p(v_{k+1})$ pertence ao caminho $(v_j, ..., v_{k-1})$, dada uma orientação. Com os vértices j, k e l definidos, o próximo passo é excluir o vértice i. Para retirar o vértice i da rota, as arestas (v_{i-1}, v_i), (v_i, v_{i+1}), (v_{j-1}, v_j), (v_l, v_{l+1}) e (v_k, v_{k+1}) são removidas, e, em seguida, a conectividade da rota é reestabelecida pela inclusão das arestas (v_{i-1}, v_k), (v_{l+1}, v_{j-1}), (v_{i+1}, v_j) e (v_l, v_{k+1}),

observando-se que os caminhos $(v_{i+1}, ..., v_{j-1})$ e $(v_{l+1}, ..., v_k)$ são invertidos para garantir o sentido da rota.

Nas Figuras 3.3 e 3.4, os métodos de remoção de US utilizam exatamente os mesmos movimentos de GENI (ver Figuras 3.1 e 3.2), porém de forma inversa. Em [10] foi observado que US pode ser aplicado como mecanismo de otimização em quaisquer rotas, e não exclusivamente às geradas com o algoritmo GENI.

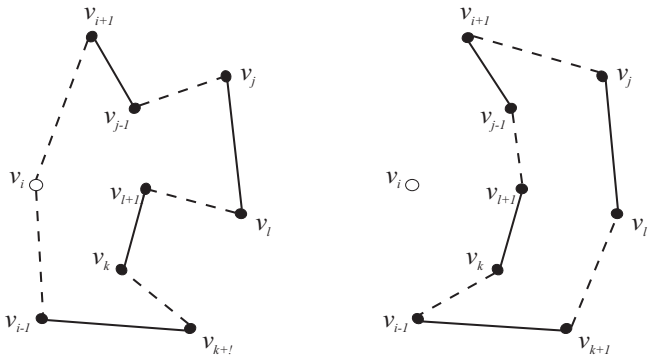

Figura 3.4: US - Remoção Tipo 2 de vértice v_i da rota

3.1.2 Heurística de Refinamento

3.1.2.1 Método de Descida em Vizinhança Variável (VND)

O método de Descida em Vizinhança Variável, do termo em inglês *Variable Neighborhood Descente* (VND) proposto em [18], é uma heurística de busca local que tem como fundamento básico a exploração do espaço de soluções em diversas estruturas de vizinhança. VND emprega a troca sistemática de vizinhanças e, em cada uma delas, somente soluções que apresentem melhora em relação a solução S atual são aceitas. Sempre que houver melhoria em uma solução obtida em uma estrutura de vizinhança $N^{(x)}(S)$, o procedimento retorna para a primeira estrutura de vizinhança, reiniciando, assim, o processo de busca local. Senão, altera-se a estrutura de vizinhança para $N^{(x+1)}(S)$. Esse algoritmo para quando todas as estruturas de vizinhanças tiverem sido exploradas, retornando a melhor solução encontrada.

No Algoritmo 5, o pseudocódigo do VND é mostrado. Nas linhas 1 e 2 definem-se a

Algoritmo 5 VND

Entrada: S, $N^{(t)}$
1: $k \leftarrow 1$;
2: $k\_max \leftarrow t$,
3: **enquanto** $k \leq k\_max$ **faça**
4: $S^{'} \leftarrow MelhorVizinho(N^{(k)}(S))$
5: **se** $(f(S^{'}) < f(S))$ **então**
6: $S \leftarrow S^{'}$;
7: $k \leftarrow 1$;
8: **senão**
9: $k \leftarrow k + 1$;
10: **fim se**
11: **fim enquanto**
12: **retorne** S;

quantidade de vizinhanças que serão exploradas. Em seguida, iterativamente, seleciona-se o melhor vizinho $S^{'}$ em cada vizinhança de S (linha 4). Na linha 5 é avaliado se a solução vizinha $S^{'}$ é melhor que S, em caso positivo $S^{'}$ passa a ser a solução corrente do VND (linha 6) e retorna-se à primeira vizinhança (linha 7), caso contrário, na linha 9, avança-se para a próxima vizinhança. Após percorrer todas as vizinhanças sem encontrar vizinho melhor que a solução corrente S, retorna-se essa como resultado da busca local (linha 12).

O método VND permite que sejam feitos uma série de ajustes. Por exemplo, realizar a exploração em toda vizinhança a cada iteração em busca do melhor vizinho, método conhecido como *Best Improvement* [20], pode ser muito exaustiva. Alternativamente, pode-se utilizar a técnica da primeira melhora, *First Improvement* [20], na qual a exploração da vizinhança é interrompida assim que se encontra uma solução vizinha melhor que a atual. Pode-se ainda, ao invés de explorar toda a vizinhança, percorrer somente um percentual dessa última. Ou ainda, fazer a exploração de forma aleatória, obtendo vizinhos quaisquer dentro do espaço de soluções de cada vizinhança, limitado a um valor máximo de iterações.

Usar a mesma ordem das estrutura de vizinhança, $(N^{(1)}(S), N^{(2)}(S), ..., N^{(t)}(S))$ em VND, pode levar a caminhos onde os elementos explorados pelos movimentos da estrutura $N^{(1)}$ sempre serão privilegiados. A alternativa é, a cada iteração ou execução do método, organizar as vizinhanças aleatoriamente. Essa variação é chamada de *Randomized Variable Neighborhood Descente* (RVND) [34].

De acordo com [18], o VND baseia-se nos seguintes princípios básicos:

- um ótimo local com relação a uma dada estrutura de vizinhança não corresponde,

necessariamente, a um ótimo local com relação a uma outra estrutura de vizinhança;

- um ótimo global corresponde a um ótimo local para todas as estruturas de vizinhança; e
- ótimos locais com relação a uma ou mais estruturas de vizinhança são relativamente próximos, para muitos problemas de otimização combinatória.

Neste trabalho optou-se por usar o VND com as estruturas de vizinhança aleatorizadas a cada execução da busca local, seguindo o critério *First Improvement* aceitando, assim, o primeiro vizinho $S' \in N^{(k)}(S)$ que melhora a solução S.

3.2 Metaheurísticas

Metaheurísticas são procedimentos genéricos para a resolução aproximada de problemas de otimização combinatória computacionalmente difíceis [32]. Diferentemente das heurísticas, que geralmente são específicas, as metaheurísticas são de finalidade geral e são providas de mecanismo capazes de escapar de ótimos locais.

Resende e Ribeiro em [30] afirmaram que as metaheurísticas são classificadas de acordo com a maneira usada para explorar o espaço de soluções, a saber: busca local e busca populacional. As metaheurísticas de busca local utilizam movimentos m sobre a solução atual S para guiar a busca em sua vizinhança chegando uma solução vizinha S'. Portanto, as metaheurísticas de busca local exploram o espaço de soluções usando o conceito de vizinhança. Já as metaheurísticas populacionais utilizam conceitos da evolução natural das espécies por meio de algoritmos evolutivos nos quais a busca ocorre em um ou mais conjunto de soluções, denominados populações, e não em uma única solução. Busca Tabu [9], (GRASP) [7, 8, 30], *Iterated Local Search* (ILS) [15] e *Variable Neighborhood Search* (VNS) [18] são exemplos de metaheurísticas de busca local. Algoritmos Genéticos [27] e Colônias de Formigas [6] são exemplos de metaheurísticas populacionais.

Na próxima seção é apresentada em detalhes a metaheurística GRASP, que foi utilizada no algoritmo proposto nesta dissertação para resolução aproximada do PRRCP.

3.2.1 GRASP

A metaheurística GRASP é um procedimento multipartida e iterativo para problemas de otimização combinatória [7, 8, 30], onde cada iteração é composta de duas fases: cons-

trução e busca local. Na fase de construção uma solução viável S_0 para o problema é construída, geralmente elemento a elemento. Uma vez obtida uma solução viável, essa tem sua vizinhança $N(S)$ explorada até chegar ao ótimo local S' na fase de busca local. A melhor solução encontrada S^* após todas as iterações é retornada como resultado desse algoritmo.

A cada iteração, a fase de construção gera um solução viável S_0, elemento a elemento, tendo com base uma lista de candidatos LC. Em LC os itens estão ordenados seguindo-se o critério de incremento de custo na solução parcial. LC é, então, construída seguindo uma função adaptativa gulosa. Essa função é adaptativa porque, a cada iteração, é refeito o cálculo de custo de cada candidato. Em problemas de minimização, ocupa a primeira posição o candidato que incrementar menos o custo em S. Porém, em GRASP, um aspecto probabilístico é requerido. Assim, a partir de LC, é construído um subconjunto restrito dos melhores candidatos, chamado de lista restrita de candidatos (LRC), do qual é aleatoriamente selecionado um elemento para fazer parte da solução.

O Algoritmo 6 apresenta o pseudocódigo da estrutura básica de um algoritmo GRASP para um problema de minimização. Conforme proposto em [8], $max\_iteracoes$ é a quantidade de iterações que serão realizadas, *seed* é um valor usado para inicializar gerador de números pseudo-aleatórios na fase de construção e α é o componente que delimita o tamanho de LRC.

Algoritmo 6 GRASP

Entrada: $max\_iteracoes$, *seed*, α

1: $S^* \leftarrow \emptyset$; $f^* \leftarrow \infty$;
2: **para** $1, 2, ..., max\_iteracoes$ **faça**
3: $S \leftarrow solucao\_inicial(semente, \alpha)$;
4: $S' \leftarrow busca\_local(S)$;
5: **se** $(f(S') < f^*)$ **então**
6: $S^* \leftarrow S'$;
7: $f^* \leftarrow f(S^*)$;
8: **fim se**
9: **fim para**
10: **retorne** S^*;

Conforme descrito anteriormente, a construção da LRC é feita entre os melhores elementos de LC. Para isso, pode-se seguir o critério de cardinalidade, no qual os p melhores elementos de LC são inseridos em LRC, ou utilizar o critério de valor associado a cada elemento candidato. Para entender o critério de valor associado, considere c^{max} e c^{min}, respectivamente, o maior e o menor custo dentre os candidatos. Assim, LRC será composta por todos os elementos $e \in LC$ cujo o custo $c(e) \in [c^{min}, c^{min} + \alpha(c^{max} - c^{min})]$. Dessa

forma, α com valores próximos a 0, leva a construções mais gulosas, e valores próximos a 1, gerarão soluções mais aleatórias, e consequentemente, mais diversas.

Em GRASP, a diversidade de soluções e tempo de execução estão relacionados ao valor do parâmetro α, que determina o tamanho de LRC, ajustando portando, o aspecto probabilístico das soluções iniciais [30]. Em [30], um estudo foi feito sobre esses elementos, no qual concluiu-se que a probabilidade de um algoritmo GRASP encontrar uma solução ótima é maior quando a variância das soluções construídas também é maior. Porém, os autores afirmaram, que a busca local levará, em média, um maior tempo de execução. Portanto, eles observaram que a escolha adequada dos valores de α é determinante para o equilíbrio entre tempo computacional e qualidade das soluções encontradas.

O Algoritmo 7 traz o pseudocódigo da fase de construção GRASP usando o critério de valor associado para elaboração da LRC.

Algoritmo 7 Fase de Construção GRASP

Entrada: *seed*, α

1: $S \leftarrow \emptyset$
2: $LC \leftarrow E$ *//inicializa lista dos candidatos*
3: $C \leftarrow c(e), \forall e \in LC$ *//matriz de custo de inserção*
4: **enquanto** (solução S não estiver completa) **faça**
5: $c^{min} \leftarrow min\{c(e) \mid e \in C\}$
6: $c^{max} \leftarrow max\{c(e) \mid e \in C\}$
7: $LRC \leftarrow \{e \in C \mid c(e) \leq c^{min} + \alpha(c^{max} - c^{min})\}$
8: $k \leftarrow SelecaoAleatoria(LRC)$
9: $S \leftarrow S \cup k$
10: $LC \leftarrow LC \backslash k$
11: $C \leftarrow c(e), \forall e \in LC$
12: **fim enquanto**
13: **retorne** S;

Definir um valor fixo para α, no GRASP, pode impedir esse algoritmo de encontrar soluções de alta qualidade ou até o ótimo global, o qual poderia eventualmente ser alcançado com outro valor [25]. Para contornar esse problema, foram propostas em [23, 24] técnicas nas quais o valor de α varia durante as iterações.

De acordo com [30], já a segunda fase do GRASP, responsável pela busca local, tem seu desempenho diretamente relacionado à qualidade da solução inicial, à estrutura de vizinhança definida, e à técnica de busca local de exploração de vizinhança adotada.

Uma possível deficiência da metaheurística GRASP padrão proposta em [7, 8], é o completo isolamento entre as suas iterações [30]. Conforme relatado anteriormente, para melhorar o desempenho da busca local, um dos fatores influenciadores é a qualidade

da solução inicial. Diversas técnicas foram incorporadas ao GRASP básico, tornando-o capaz de usar informações sobre sua memória, no momento da construção da solução, atenuando essa deficiência levantada. Em [30, 32] foram analisadas diversas extensões e técnicas alternativas para melhorar o desempenho da metaheurística GRASP, incluindo GRASP reativo, perturbação nos custos, memória e aprendizado, filtragem, reconexão por caminhos, paralelismo, entre outros.

O Algoritmo 8 apresenta um pseudocódigo da fase de busca local da metaheurística GRASP. O processo tem como ponto de partida a solução inicial S_0 (linha 1), gerada pela primeira fase da iteração GRASP. Iterativamente, o algoritmo procura por soluções cujo valor da função objetivo seja inferior ao da solução corrente. Assim, ao encontrar uma solução melhor, essa torna-se a solução corrente (linha 3). A busca continua até não mais existir nenhuma solução melhor que a solução corrente. A melhor solução, nesse caso denominada de ótimo local, é retornada como resultado da fase de busca local GRASP [31].

Algoritmo 8 Fase de Busca Local GRASP

1: $S \leftarrow S_0$
2: **enquanto** existe $S^{'} \in N(S)$ tal que $f(S^{'}) < f(S)$ **faça**
3: $\quad S \leftarrow S^{'}$;
4: **fim enquanto**
5: **retorne** S;

De acordo com [30], em relação às demais metaheurísticas, GRASP leva vantagem devido à sua facilidade de implementação, visto que poucos parâmetros são necessários. GRASP, originalmente, tem apenas dois parâmetros que necessitam sofrer ajustes: um é o critério de parada, geralmente dado pela quantidade máxima de iterações, e outro, o α que determina a qualidade dos elementos que pertencerão a LRC.

No Capítulo 4 é apresentado o algoritmo heurístico proposto neste trabalho a partir da metaheurística GRASP. Para isso, mostram-se as adaptações realizadas nas fases da metaheurística GRASP, por meio das heurísticas descritas neste capítulo, considerando as especificidades do PRRCP.

Capítulo 4

Metodologia Aplicada

Este capítulo mostra a metodologia utilizada neste trabalho para resolver o problema de recobrimento de rotas com coletas de prêmios (PRRCP), apresentando, como contribuição principal, um novo algoritmo para resolução aproximada do PRRCP.

Esta abordagem é baseada em GRASP, tendo como módulo de geração de solução inicial a heurística GENIUS e o método de exploração de busca local guiado pelo VND. A estratégia de busca usa um conjunto de estruturas de vizinhanças aleatoriamente ordenadas a cada invocação deste procedimento.

A primeira seção deste capítulo descreve a função de avaliação aplicada para verificar a qualidade das soluções obtidas pelos algoritmos heurísticos propostos. Em seguida, as estruturas de vizinhança utilizadas no processo de busca local com VND são apresentadas. E, finalmente, explica-se, em mais detalhes, a estrutura principal do algoritmo proposto, onde é mostrado como cada uma das heurísticas e técnicas descritas no Capítulo 3 foram combinadas e aplicadas na resolução do PRRCP.

4.1 Avaliação de uma solução do PPRCP

As soluções do PRRCP são avaliadas pela função f descrita pela Equação 4.1, na qual contabilizam-se os somatórios dos custos de deslocamento total na rota.

$$f(S) = \sum_{i,j \in V} c_{ij} x_{ij} \tag{4.1}$$

Onde,

- S é a solução PRRCP encontrada pelo algoritmo;
- V é o conjunto de todos os vértices que podem entrar na rota, os quais formam as arestas (i, j) da rota;
- c_{ij} é o custo de deslocamento de vértice i até j, com $i, j \in V$; e
- x_{ij} é a variável binária que indica se a aresta (i, j) aparece na rota.

O algoritmo heurístico proposto admite somente soluções viáveis tanto no processo de geração de solução inicial, quanto na exploração das vizinhanças no processo de busca local.

4.2 Estruturas de Vizinhança

Nesta seção as estruturas de vizinhanças utilizadas durante a fase de busca local para explorar o espaço de soluções são descritas. Devido às especificidades do PRRCP, dentre elas a existência de pontos (ou vértices) opcionais, ou seja, que podem não pertencer à rota, as estruturas de vizinhança são divididas em dois tipos: intra-rota e extra-rota.

Estruturas de vizinhanças cujo movimentos envolvam exclusivamente vértices que já fazem parte da rota, alterando portanto, apenas as suas posições dentro da solução, são definidas como intra-rota. Por outro lado, classificam-se como estruturas de vizinhanças extra-rotas, aquelas em que os movimentos utilizam, para inclusão ou substituição, o subconjunto de vértices opcionais que não pertecem à rota, $(V \backslash T \notin S)$, que encontra-se em processo de melhoria por meio da busca local.

O método de busca local do algoritmo desenvolvido neste trabalho tem à sua disposição quinze estruturas de vizinhanças, baseadas em movimentos clássicos do PCV, como o *SHIFT* e o *SWAP*, e em outros movimentos compostos baseados nas heurísticas GENIUS e na inserção mais barata, utilizados para realocação ou substituição de vértices na rota. Estruturas de vizinhança com procedimentos compostos foram utilizadas com sucesso em [22, 35]. Diferentemente das estruturas de vizinhanças de [35], que permitiam que seus movimentos obtivessem, também, soluções inviáveis, neste trabalho as estruturas de vizinhança foram propostas para que se considerassem todas as características do PRRCP para estabelecer os conjuntos de vértices e movimentos disponíveis. Dessa forma, somente movimentos que mantêm a solução viável são admitidos na fase de busca local. Considera-se como solução viável PRRCP aquela que: a) contenha todos os vértices obrigatórios; b)

a restrição de cobertura de vértices seja atendida; e, c) o somatório dos prêmios de seus vértices seja maior ou igual ao *PRIZE*.

Neste presente trabalho, além das estruturas de vizinhanças encontradas na literatura, propõe uma nova estrutura de vizinhança, denominada *dupla remocao simples inserir mais barata*, na qual dois vértices opcionais $(V \backslash T)$, $(i, j \in S)$, podem ser substituídos por um vértice $(v \notin S)$, desde que o vértice v, satisfaça às seguintes condições: a) forneça a cobertura dos vértices do conjunto W, que tenham ficado descobertos com a exclusão de i e j da rota; e b) que seu prêmio seja suficiente para garantir que prêmio total da rota não seja inferior ao $PRIZE$ requerido.

Das quinze estruturas de vizinhanças utilizadas, dez são intra-rota:

1. *shift*: troca um vértice de posição dentro da rota;
2. *swap*: consiste na inversão de posição de um vértice i com outro vértice j dentro da rota;
3. *or-opt*: movimento similar ao *shift*, porém é feita a troca de posição de n vértices consecutivos dentro da rota;
4. *2-opt*: remove duas arestas não adjacentes da solução e insere duas novas para manter o ciclo único;
5. *3-opt*: remove três arestas não adjacentes e insere três novas rotas, semelhante ao *2-opt*;
6. *remover_simples_re_inserir_mais_barata*: remove um vértice e o reinsere via inserção mais barata;
7. *remover_simples_re_inserir_genius*: remove um vértice e o reinsere via um dos métodos de GENIUS;
8. *remover_genius_re_inserir_mais_barata*: remove um vértice usando um dos métodos de remoção de GENIUS e faz a inserção usando critério de menor custo entre vértices adjacentes.
9. *remover_genius_re_genius*: remove e reinsere um vértice de S usando as técnicas de GENIUS; e
10. *remover_mais_barata_re_inserir_genius*: remove usando critério de inserção mais barata e reinsere via GENIUS.

E cinco são do tipo extra-rota:

1. *dupla_ remocao_ simples_ inserir_ mais_ barata*: substitui dois vértices ($V \backslash T$) por um único que não pertença à solução;
2. *remover_ simples_ inserir_ mais_ barata*: substituir um vértice ($V \backslash T$) da rota, inserindo via inserção mais barata;
3. *remover_ simples_ inserir_ genius*: substituir um vértice ($V \backslash T$) da rota, inserindo via GENIUS;
4. *remover_ genius_ inserir_ genius*: substituir um vértice ($V \backslash T$) da rota removendo e inserindo via GENIUS; e
5. *remover_ genius_ inserir_ mais_ barata*: retira um vértice ($V \backslash T$) da solução via GENIUS e insere um novo via inserção mais barata.

4.3 Algoritmo GRASP aplicado ao PRRCP

A metaheurística GRASP é um método no qual cada iteração é dividida em duas fases: uma de construção e outra de refinamento ou busca local [7, 8, 30]. Na fase de construção, uma solução viável para problema é construída e, em seguida, tem sua vizinhança explorada durante a fase de busca local. Esse procedimento é repetido até que se atinja o critério de parada estabelecido. A melhor solução encontrada dentre todas as iterações é retornada como resultado. GRASP é, portanto, um método multi-partida e iterativo. A seguir uma versão de GRASP híbrido proposto para resolução do PRRCP é apresentada.

4.3.1 GRASP-GENIUS-VND

Foi proposto para a resolução do PRRCP o algoritmo heurístico identificado como GRASP-GENIUS-VND que visa encontrar soluções viáveis de boa qualidade (ou até ótimas) em tempo computacional aceitável e inferior aos encontrados na literatura. Hibridizou-se a metaheurística GRASP com as heurísticas GENIUS, como método de geração de solução inicial, e com o método VND como mecanismo de exploração da vizinhança das soluções iniciais geradas. Dessa forma, buscou-se unir a já conhecida simplicidade da implementação de GRASP, ao elaborado método de construção de GENIUS e a flexibilidade de VND em explorar diversas estruturas de vizinhanças sistematicamente.

Nesta nova heurística o critério de parada usado é o encontro de uma solução prefixada ou alcançar um tempo máximo de execução préviamente estabelecido. No Algoritmo 9, a estrutura geral do GRASP proposto é apresentada.

Algoritmo 9 GRASP-GENIUS-VND

Entrada: $tempo\_max$; $alvo$;

```
1:  S* ← ∅; f* ← ∞; tempo_execucao ← 0;
2:  enquanto tempo_execucao < tempo_max faça
3:      S ← solucao_inicial_GENIUS(seed, α);
4:      S' ← busca_local_VND(S);
5:      se (f(S') < f*) então
6:          S* ← S';
7:          f* ← f(S*);
8:          se (f* <= alvo) então
9:              retorne S*;
10:         fim se
11:     fim se
12:     tempo_execucao ← obter_tempo_execucao();
13: fim enquanto
14: retorne S*;
```

Pode-se observar que no Algoritmo 9 são necessários apenas três parâmetros de entrada: $tempo\_max$, que define a tempo máximo a ser executado pelo algoritmo caso este não encontre a solução $alvo$ prefixada, ou uma melhor que esta, que é segundo parâmetro do algoritmo; e, α que tem um valor fixo previamente estabelecido.

O algoritmo executa iterativamente os procedimentos de construção inicial e busca local (linhas 3 e 4). A solução inicial S é construída pelo algoritmo GENIUS, modificado para atender às restrições do PRRCP e aos princípios de GRASP, e em seguida a rota inicial PRRCP obtida é passada, como parâmetro de entrada, para a busca local, que é composta de um VND e de um conjunto de estruturas de vizinhanças. O VND retorna uma solução vizinha S', que é comparada à melhor solução encontrada pelas iterações anteriores (linha 5). Se houver melhoria de custo, atualiza-se a melhor solução atual. Além disso, na linha 8 é verificado se alcançou-se a solução alvo estabelecida ou uma melhor. Em caso positivo, o algoritmo é interrompido retornando essa solução (linha 9). Na linha 10 atualiza-se o tempo de execução.

Na seção seguinte a heurística GENIUS é adaptada para realizar a construção de soluções iniciais PRRCP.

4.3.1.1 Método de Construção de Solução Inicial

Uma instância do PRRCP é uma matriz multidimensional que contém:

- A distância euclidiana entre todos os vértices;
- O tipo de cada um dos vértices: V, $V \backslash T$ ou W;
- O prêmio (p_i) dos vértices V, $p_i > 0, \forall i \in V$;
- A distância de cobertura para os vértices W; e
- O $PRIZE$ a ser coletado.

A partir das informações acima, que fazem parte da caracterização do problema, as técnicas da heurística GENIUS (Seção 3.1.1.2) foram adaptadas para que essa heurística seja usada como mecanismo de construção de solução inicial para o GRASP proposto, conforme descrito nas próximas seções.

4.3.1.2 GENIUS adaptada ao PRRCP

A construção de soluções iniciais em algoritmos baseados em GRASP geralmente é feita elemento a elemento e deve seguir a dois critérios concomitantemente, devidamente ponderados pelo valor do parâmetro α: gulosidade e randomicidade. Algoritmos gulosos escolhem sempre a melhor opção atual gerando caminhos determinísticos. Com o objetivo de gerar soluções diversificadas, o GRASP exige a inclusão de uma componente probabilística, onde são escolhidos n melhores elementos da lista de candidatos (LC) seguindo algum critério guloso, e desse subconjunto, formado por um intervalo de valores calculados usando-se α diretamente, somente um será escolhido, aleatoriamente, para fazer parte da rota. Esse subconjunto é conhecido como lista restrita de candidatos (LRC), e é responsável pelo aspecto probabilístico da metaheurística. Em seguida, após a inclusão de cada vértice, os valores que determinam o benefício dos demais elementos que ainda não pertencem a rota são atualizados.

As principais mudanças no GENIUS, para torná-lo um gerador de soluções iniciais, devem ser feitas na fase GENI, que é de fato responsável pela construção da rota, especificamente na definição dos vértices candidatos e nas condições de integralidade da solução. Assim, inicialmente restringe-se a LC, e consequentemente a LCR, ao subconjunto dos

vértices obrigatórios T. Dessa forma, inicialmente, atende-se à primeira restrição do PRRCP que é conter todos os vértices obrigatórios. Uma vez inseridos todos os vértices de T, faz-se um levantamento dos vértices de W que continuam sem cobertura, ou seja, não existe na rota algum vértice cuja distância euclidiana seja igual ou inferior a distância de cobertura definida. Com a lista dos vértices ainda descobertos, passa a fazer parte de LC e de LRC, apenas o vértices opcionais ($V \backslash T$), que cubram pelo menos um dos vértices de W ainda descobertos. E, finalmente, quando não houver mais vértices W descobertos, todos os demais vértices passam a ser candidatos. Os métodos de inserção de GENI continuam sendo aplicados até que a última restrição de viabilidade seja alcançada, a coleta integral do $PRIZE$, que é dada pelo somatório dos prêmios p_i dos vértices da rota.

Algoritmo 10 GENIUS

$i, j, k \leftarrow SelecaoAleatoria(LC)$;
$S \leftarrow \{i - j - k\}$;
$LC \leftarrow LC - \{i, j, k\}$
$LC \leftarrow DefinirListaCandidatos(vertices\_t)$;
enquanto $((\exists v_i \in T) \notin S)$ **faça**
    $LRC \leftarrow DefinirLRC(LC)$;
    $v \leftarrow SelecaoAleatoria(LRC)$;
    $S \leftarrow GENI(S, v)$;
    $LC \leftarrow LC \backslash v$;
fim enquanto
$LC \leftarrow DefinirListaCandidatos(vertices\_cobertura)$;
enquanto $(\exists v_i \in WDescobertos)$ **faça**
    $LRC \leftarrow DefinirLRC(LC)$;
    $v \leftarrow SelecaoAleatoria(LRC)$;
    $S \leftarrow GENI(S, v)$;
    $LC \leftarrow LC \backslash v$;
fim enquanto
$LC \leftarrow DefinirListaCandidatos(V \backslash S)$;
enquanto $(\sum_{v_i \in S} p_i < PRIZE)$ **faça**
    $LRC \leftarrow DefinirLRC(LC)$;
    $v \leftarrow SelecaoAleatoria(LRC)$;
    $S \leftarrow GENI(S, v)$;
    $LC \leftarrow LC \backslash v$;
fim enquanto
$S^* \leftarrow US(S)$; *//melhoria da rota*
retorne S^*;

Com a rota construída (fase GENI), a heurística GENIUS invoca seu procedimento de melhoria (fase US), com o objetivo de verificar se existe dentro da rota uma posição melhor para cada um dos seus vértices. A fase US faz dois tipos de remoção seguidos de reinserção pelos métodos de GENI, caso alguma dessas tentativas resulte em alguma

solução com custo melhor. Assim, essa nova solução passar a ser a rota atual do GENIUS. Como relatado na Seção 3.1.1.2, o objetivo da fase US é apenas diminuir a distância total percorrida pela rota por meio de movimentações internas, ou seja, as variáveis de viabilidade do PRRCP não são afetadas.

O Algoritmo 10 apresenta o pseudocódigo de GENIUS adaptada à construção de soluções iniciais para o PRRCP. Inicialmente, cria-se uma subrota S com três vértices do conjunto dos vértices T, selecionados aleatoriamente (linhas 1 e 2). A seguir, na linha 4, a lista de candidatos passa a ser composta exclusivamente por vértices do conjunto T e, da linha 5 à linha 10, enquanto houver vértices do conjunto T que não perteçam a rota, realiza-se, iterativamente, o processo pseudoaleatório de construção da rota. Com todos os vértices obrigatórios já inseridos, passa-se a restringir a lista de candidatos aos vértices opcionais (linha 11), e estes são iterativamente inseridos na rota até que não haja mais vértices do conjunto W descobertos (linha 12 à linha 17).

Com duas restrições do PRRCP já atendidas, a lista de candidatos passa a ser composta de todos os vértices que ainda não fazem parte da rota (linha 18), e, mais um vez repete-se o processo de construção GRASP até que se atinja a última restrição do PRRCP, a coleta de prêmios dada pelo $PRIZE$ estabelecido (linha 19 à linha 24). Finalmente, com uma solução S já construída, na linha 25 invoca-se a fase de melhoria da heurística GENIUS, o processo US, e a rota S^* obtida por esse procedimento é assumida como resultado do método de construção de soluções iniciais GENIUS. Ao final, na linha 26, a solução S^* será repassada para a fase de busca local do algoritmo GRASP que é guiada pelo método VND, conforme descrito na seção a seguir.

4.3.1.3 Método de Descida em Vizinhança Variável (VND) para PRRCP

Ao concluir a execução do módulo de construção de solução inicial utilizando a adaptação do GENIUS descrita na última seção, uma solução viável para o PRRCP é obtida. Porém, essa rota pode estar ainda bem distante dos ótimos locais.

O método de descida em vizinhança variável (VND) [18] foi usado neste trabalho. O VND é um método de refinamento que consiste em explorar o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança previamente ordenadas, aceitando somente soluções que melhorem a solução corrente, e retornando à primeira estrutura sempre que houver movimento que resulte em melhoria na solução atual.

A efetividade da busca local está ligada a vários aspectos, como a efetividade das

estruturas de vizinhança utilizadas, ou seja, a qualidade dos movimentos que serão realizados para navegar no espaço de soluções e a própria solução inicial recebida [30]. Nesta abordagem utilizam-se duas categorias de estruturas de vizinhança: intra-rota e extra-rota (descritas na Seção 4.2). VND permite uma série de ajustes que podem afetar diretamente o desempenho da busca local.

Na implementação das estruturas de vizinhança deste trabalho optou-se em restringí-las a movimentos que não interferissem nas medidas de viabilidade da rota obtida pelo algoritmo de solução de inicial, ou seja, todo e qualquer movimento em todas as estruturas de vizinhança sempre levam de uma rota viável a uma outra rota vizinha também viável. Outro ajuste realizado nesta proposta foi a utilização da técnica conhecida como *first improvement method* - ou método de primeira melhora - no qual ao invés de explorar toda uma vizinhança em busca do melhor vizinho - *best improvement method* - o VND interrompe a varredura na atual estrutura de vizinhança e retorna à primeira assim que encontrar o primeiro vizinho que apresente melhoria de custo em relação à rota atual. Em [19] pode-se verificar que a técnica *first improvement* pode ter o desempenho melhor que a exploração da vizinhança completa (*best improvement*). A fim de evitar que a abordagem *first improvement* torne o VND determinístico, ou seja, encontre sempre a mesma sequência de vizinhos, a ordem de aplicação das estruturas de vizinhança é aleatoriamente organizada a cada invocação do algoritmo de busca local, procedimento conhecido como RVND.

Algoritmo 11 VND

Entrada: S, $N^{(t)}$
1: $k \leftarrow 1$;
2: $k\_max \leftarrow t$;
3: $N \leftarrow InicializacaoAleatoria(k\_max)$;
4: **enquanto** $k \leq k\_max$ **faça**
5: $S' \leftarrow PrimeiroMelhorVizinho(N^{(k)}(S))$
6: **se** $(f(S') < f(S))$ **então**
7: $S \leftarrow S'$;
8: $k \leftarrow 1$;
9: **senão**
10: $k \leftarrow k + 1$;
11: **fim se**
12: **fim enquanto**
13: **retorne** S;

O pseudocódigo do módulo de busca local descrito acima é ilustrado no Algoritmo 11. Inicialmente, a ordem de aplicação das estruturas de vizinhanças é aleatoriamente definida (linha 3). Em seguida, enquanto houver vizinhança a ser explorada, submete-se a solução

S à vizinhança corrente, $N^{(k)}(S)$ (linha 5). Se houver vizinho S' que apresente melhoria de custo, atualiza-se a solução corrente (linha 7) e retorna-se à primeira vizinhança (linha 8). Caso contrário, se S não for melhorada, na linha 10 avança-se à proxima estrutura de vizinhança. A busca local para quando todas as estruturas de vizinhança forem exploradas e nenhum vizinho melhore a solução atual, retornando a última como resultado desse precedimento de intensificação de soluções (linha 13).

No Capítulo 5 são comparados os resultados obtidos pelas execuções do novo algoritmo proposto e as duas melhores abordagens ILS de Silva [35], a fim de verificar a eficiência dessa nova heurística híbrida.

Capítulo 5

Resultados Computacionais

Neste capítulo são apresentados os resultados dos experimentos computacionais realizados nesta dissertação. Inicialmente são apresentados os problemas-teste utilizados. Em seguida, os resultados gerais em qualidade de solução e em tempo computacional são apresentados e comparados aos resultados de duas abordagens encontradas na literatura. Finalmente, é realizada uma análise do comportamento do algoritmo proposto por meio de um teste de probabilidade empírica.

Dentre as cinco versões de algoritmos ILS propostos em [35], foram utilizadas, para realizar as comparações deste capítulo, as versões que apresentaram melhor desempenho em cada um dos grupos de instâncias utilizadas: para o Grupo 1 a versão ILS-MRD_AD e para o Grupo 2 a versão ILS-MRD_IB, que utilizam as heurística ADD e Inserção Mais Barata, respectivamente, como método de geração de solução inicial. O algoritmo evolutivo, embora tenha apresentado maior taxa de sucesso, não foi utilizado devido ao altos tempos computacionais de execução que ele requer, principalmente para as instâncias do Grupo 2, conforme relatado em [35], fato que diminuiu sua competitividade operacional em relação às versões ILS.

5.1 Ambiente Computacional

Os experimentos computacionais foram executados em ambiente com computador dotado de processador $Intel®Core^{TM}$ i5 CPU 650 @ 3.2GHz, com 4GB de RAM e sistema operacional Linux Fedora 15, por meio do Wine[1], que é uma camada de tradução que permite a execução de aplicações windows em ambiente linux ou em outros sistemas operacionais.

[1] http://www.winehq.org/

Embora o processador utilizado tenha quatro núcleos, esse recurso (multiprocessamento ou processamento paralelo) não foi utilizado.

O algoritmo GRASP-GENIUS-VND proposto neste trabalho foi desenvolvido utilizando-se a linguagem de programação C e compilado com o *g++*, versão 4.7.0. para o sistema operacional Windows. Já os algoritmos ILS, propostos em [35], disponibilizados pelo autor para uma comparação justa das abordagens em um mesmo ambiente computacional, foram implementados na linguagem de programação *C++* e compilados no ambiente integrado de desenvolvimento Microsoft Visual Studio 2008 para sistema operacional Windows. As duas abordagens foram compiladas para execução em ambiente com arquitetura de 64 bits.

5.2 Instâncias do PRRCP

Para validação da abordagem proposta neste trabalho, os experimentos computacionais foram realizados utilizando-se os conjuntos de problemas-teste desenvolvidos em [35], os quais foram adaptados a partir de problemas-teste para o problema do caixeiro viajante disponível no repositório TSPLIB [28]. Foram geradas 144 instâncias PRRCP, divididas em dois grupos: Grupo 1, formado por 111 instâncias contendo, entre 48 e 200 vértices, cada uma; e Grupo 2, composto por 33 instâncias cada uma com pelo menos 201 vértices e, no máximo, 400.

As instâncias foram nomeadas a partir do nome original usado na TSPLIB complementado pelas informações relativas às quantidades de cada tipo de vértices e o percentual de prêmio a ser coletado em relação ao total disponível. A instância *att48_ VT9_ T10_ W29_ 25*, por exemplo, é um problema teste gerado usando a instância TSPLIB de nome *att*48, que teve seus 48 vértices segmentados da seguinte forma:

- 9 vértices opcionais $V \backslash T$;
- 10 vértices obrigatórios T;
- 29 vértices de cobertura W; e
- 25 representa o percentual estabelecido como *PRIZE*, isto é, nesse caso deve-se coletar 25% do prêmio total da instância.

Foram aplicadas todas as regras de redução, encontradas na literatura do PRRCP e citadas na Seção 2.2.2, sobre as 144 instâncias utilizadas.

5.3 Definição do valor de alfa

Conforme descrito no Capítulo 3, um dos aspectos da metaheurística GRASP é seu comportamento probabilístico. Para isso, é definido, na fase de construção de solução inicial, um subconjunto com os melhores candidatos, nomeado lista restrita de candidatos (LRC), onde o critério de valor associado pode ser utilizado para definir quais vértices candidatos farão parte da LRC. Nesta abordagem proposta, LRC será composta por todos os elementos $e \in LC$ cujo custo $c(e) \in [c^{min}, c^{min} + \alpha(c^{max} - c^{min})]$. Dessa forma, α com valores próximos a zero, tornam a construções mais gulosas, e, de forma inversa, valores de α próximos a um geram-se soluções mais aleatórias e, consequentemente, mais diversas.

A escolha apropriada do valor do parâmetro α da LRC é bastante crítica e relevante para se alcançar um bom equilíbrio entre tempo computacional gasto e qualidade de soluções encontradas em algoritmos GRASP [30]. Diante disto, para definição do melhor valor de α para o algoritmo heurístico GRASP proposto neste trabalho, foram realizadas quatro execuções prévias para cada uma das instâncias, tanto do Grupo 1, quanto do Grupo 2, utilizando sementes diferentes de números aleatórios em cada um delas. Sendo que, para cada semente, foram realizadas dez execuções, variando-se em cada um delas o valor de α utilizado, partindo-se de 0.1 até 1.0, com intervalo de incremento de 0.1.

As Tabelas 5.1 e 5.2 resumem os resultados dos testes realizados para identificar o melhor valor de α, para as instâncias dos Grupos 1 e 2, respectivamente. Nessas tabelas, a primeira coluna indica o nome de cada um dos problemas-teste, as dez colunas seguintes constam as médias dos custos para os dez valores de α nas quatro execuções para cada instância. A última linha, em cada tabela, indica os valores médios gerais para cada α.

Analisando-se a Tabela 5.1, pode-se verificar que, para o Grupo 1, obtiveram melhor valor médio de custo as soluções obtidas com os algoritmos configurados com α igual a 0.6, 0.7 e 0.9, com custo médio igual a 20084.35, 20084.28 e 20084.28, respectivamente, e o pior desempenho foi do algoritmo definido com valor α igual a 0.1 (essa versão obteve valor médio de custo de solução de 20112.38). Já para os problemas-teste do Grupo 2, de acordo com Tabela 5.2, tiveram melhor desempenho os algoritmos utilizando os valores de α em 0.6 e 0.8, com valores iguais a 33717.82 e 33717.61, respectivamente. A exemplo do Grupo 1, o algoritmo com pior desempenho foi, também, a versão com α igual 0.1 (que teve valor médio de custo de soluções de 33828.95). Assim, de acordo com os dados das Tabelas 5.1 e 5.2, definiu-se, então, o valor de alfa igual a 0.6 para ser utilizado na definição da LRC do algoritmo heurístico GRASP proposto neste trabalho.

Tabela 5.1: Definição do valor de alfa - Grupo 1.

Instâncias Grupo 1										
Nome/Valores de Alfa	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
att48_VT9_T10_W29_25	2380.00	2292.00	2292.00	2292.00	2291.00	2291.00	2291.00	2291.00	2291.00	2291.00
att48_VT9_T29_W10_50	2835.25	2839.00	2824.00	2824.00	2824.00	2824.00	2824.00	2824.00	2824.00	2824.00
att48_VT16_T16_W16_75	2650.00	2650.00	2650.00	2650.00	2650.00	2650.00	2650.00	2650.00	2650.00	2650.00
berlin52_VT9_T11_W32_25	4868.00	4861.00	4861.00	4861.00	4861.00	4861.00	4861.00	4861.00	4861.00	4861.00
berlin52_VT16_T18_W18_50	5215.00	5200.50	5186.00	5186.00	5186.00	5186.00	5186.00	5186.00	5186.00	5186.00
berlin52_VT9_T32_W11_75	6916.00	6916.00	6916.00	6916.00	6916.00	6916.00	6916.00	6916.00	6916.00	6916.00
bier127_VT24_T77_W26_25	98725.00	98640.50	98683.75	98789.75	98555.25	98667.00	98687.75	98676.50	98591.25	98657.50
bier127_VT24_T26_W77_50	74745.00	74745.00	74745.00	74745.00	74745.00	74745.00	74745.00	74745.00	74745.00	74745.00
bier127_VT43_T42_W42_75	98835.50	98737.50	98661.25	98625.25	98637.00	98334.25	98329.00	98329.00	98311.00	98352.25
brazil58_VT18_T20_W20_25	19417.00	19417.00	19417.00	19417.00	19417.00	19417.00	19417.00	19417.00	19417.00	19417.00
brazil58_VT11_T35_W12_50	23312.00	23312.00	23312.00	23312.00	23312.00	23312.00	23312.00	23312.00	23312.00	23312.00
brazil58_VT11_T12_W35_75	20466.00	20466.00	20466.00	20466.00	20466.00	20466.00	20466.00	20466.00	20466.00	20466.00
brg180_VT36_T36_W108_25	550.00	550.00	550.00	550.00	550.00	550.00	550.00	550.00	550.00	550.00
brg180_VT60_T60_W60_50	2360.00	2360.00	2360.00	2360.00	2360.00	2360.00	2360.00	2360.00	2360.00	2360.00
brg180_VT36_T108_W36_75	2210.00	2210.00	2212.50	2210.00	2210.00	2210.00	2212.50	2210.00	2212.50	2210.00
ch130_VT44_T43_W43_25	4392.00	4387.50	4385.25	4383.00	4383.00	4383.00	4383.00	4383.00	4385.25	4383.00
ch130_VT26_T26_W78_50	4139.00	4139.00	4139.00	4139.00	4139.00	4139.00	4139.00	4139.00	4139.00	4139.00
ch130_VT26_T78_W26_75	5301.00	5303.00	5300.50	5299.50	5299.50	5302.50	5299.75	5301.00	5302.25	5304.00
ch150_VT50_T50_W50_25	4799.25	4799.25	4790.75	4795.50	4803.00	4796.25	4803.00	4798.75	4800.00	4790.00
ch150_VT30_T90_W30_50	5239.25	5239.25	5239.00	5242.50	5242.25	5242.75	5248.00	5247.50	5247.00	5248.50
ch150_VT30_T30_W90_75	4292.00	4292.00	4292.00	4292.00	4292.00	4292.00	4292.00	4292.00	4292.00	4292.00
d198_VT39_T40_W119_25	10916.00	10916.00	10916.00	10916.00	10916.00	10916.00	10916.00	10916.00	10916.00	10916.00
d198_VT39_T119_W40_50	13601.75	13605.50	13604.00	13598.50	13599.25	13602.00	13601.50	13598.00	13600.50	13601.00
d198_VT66_T66_W66_75	12913.50	12908.50	12911.50	12911.50	12920.00	12922.50	12928.75	12930.75	12924.00	12920.00

Tabela 5.1: Definição do valor de alfa - Grupo 1.

Instâncias Grupo 1										
Nome/Valores de Alfa	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
eil51_VT17_T17_W17_25	273.00	273.00	273.00	273.00	273.00	273.00	273.00	273.00	273.00	273.00
eil51_VT9_T11_W31_50	248.00	248.00	248.00	248.00	248.00	248.00	248.00	248.00	248.00	248.00
eil51_VT9_T31_W11_75	348.50	348.25	348.50	348.25	348.50	348.50	348.25	348.50	348.25	348.00
eil76_VT14_T46_W16_25	387.00	387.00	387.00	387.00	387.00	387.00	387.00	387.00	387.00	387.00
eil76_VT24_T26_W26_50	338.00	338.00	338.00	338.00	338.00	338.00	338.00	338.00	338.00	338.00
eil76_VT14_T16_W46_75	332.00	332.00	332.00	332.00	332.00	332.00	332.00	332.00	332.00	332.00
eil101_VT19_T21_W61_25	356.00	356.50	356.25	356.00	356.00	356.00	356.00	356.00	356.00	356.00
eil101_VT19_T61_W21_50	509.00	507.75	508.25	506.00	506.00	506.00	506.00	506.00	506.00	506.00
eil101_VT33_T34_W34_75	463.00	463.50	463.00	463.00	462.50	462.75	463.50	463.00	462.75	462.50
gr48_VT16_T16_W16_25	3571.00	3571.00	3571.00	3571.00	3571.00	3571.00	3571.00	3571.00	3571.00	3571.00
gr48_VT9_T10_W29_50	3526.00	3526.00	3524.00	3524.00	3524.00	3524.00	3524.00	3524.00	3524.00	3524.00
gr48_VT9_T29_W10_75	4498.25	4415.00	4415.00	4415.00	4415.00	4415.00	4415.00	4415.00	4415.00	4415.00
gr96_VT18_T58_W20_25	42775.75	42764.00	42746.00	42746.00	42746.00	42746.00	42746.00	42746.00	42746.00	42746.00
gr96_VT32_T32_W32_50	35461.00	35461.00	35461.00	35461.00	35461.00	35461.00	35461.00	35461.00	35461.00	35461.00
gr96_VT18_T20_W58_75	30015.00	30015.00	30015.00	30015.00	30015.00	30015.00	30015.00	30015.00	30015.00	30015.00
gr120_VT40_T40_W40_25	4301.00	4301.00	4301.00	4301.00	4301.00	4301.00	4301.00	4301.00	4301.00	4301.00
gr120_VT24_T72_W24_50	5165.50	5164.25	5167.50	5162.00	5161.75	5162.00	5160.25	5164.25	5160.25	5161.75
gr120_VT24_T24_W72_75	4275.00	4275.00	4275.00	4275.00	4275.00	4275.00	4275.00	4275.00	4275.00	4275.00
gr137_VT26_T28_W83_25	39305.00	39305.00	39305.00	39305.00	39305.00	39305.00	39305.00	39305.00	39305.00	39305.00
gr137_VT45_T46_W46_50	46122.00	46122.00	46122.00	46122.00	46122.00	46122.00	46122.00	46122.00	46122.00	46122.00
gr137_VT26_T83_W28_75	58731.75	58731.75	58720.00	58720.00	58743.50	58806.25	58731.75	58815.00	58731.75	58731.75
hk48_VT9_T29_W10_25	9709.50	9709.25	9708.00	9708.00	9708.00	9708.00	9708.00	9708.00	9708.00	9708.00
hk48_VT16_T16_W16_50	7593.00	7593.00	7593.00	7593.00	7593.00	7593.00	7593.00	7593.00	7593.00	7593.00
hk48_VT9_T10_W29_75	7803.00	7803.00	7803.00	7803.00	7803.00	7803.00	7803.00	7803.00	7803.00	7803.00

Tabela 5.1: Definição do valor de alfa - Grupo 1.

Instâncias Grupo 1										
Nome/Valores de Alfa	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
kroA100_VT20_T20_W60_25	10848.00	10848.00	10848.00	10848.00	10848.00	10848.00	10848.00	10848.00	10848.00	10848.00
kroA100_VT20_T60_W20_50	16544.00	16544.00	16544.00	16544.00	16544.00	16544.00	16544.00	16544.00	16544.00	16544.00
kroA100_VT34_T33_W33_75	14606.00	14606.00	14606.00	14606.00	14606.00	14606.00	14606.00	14606.00	14606.00	14606.00
kroA150_VT50_T50_W50_25	14403.00	14403.00	14403.00	14404.50	14389.00	14408.00	14453.75	14570.25	14551.50	14687.75
kroA150_VT30_T30_W90_50	13112.00	13029.50	13054.00	12999.50	12927.50	12980.00	12937.25	12947.00	12918.00	12918.00
kroA150_VT30_T90_W30_75	21981.75	22015.75	21977.00	21990.25	21980.00	21988.00	21980.00	21990.25	21987.25	21992.25
kroA200_VT40_T120_W40_25	23509.25	23501.75	23497.00	23496.00	23495.00	23496.00	23495.00	23495.00	23495.00	23495.00
kroA200_VT68_T66_W66_50	17347.50	17328.25	17318.75	17319.50	17319.50	17326.00	17327.00	17318.25	17325.50	17319.00
kroA200_VT40_T40_W120_75	15274.00	15204.75	15197.00	15197.00	15197.00	15197.00	15197.00	15197.00	15197.00	15197.00
kroB100_VT20_T60_W20_25	18406.00	18406.00	18406.00	18406.00	18406.00	18406.00	18406.00	18406.00	18406.00	18406.00
kroB100_VT34_T33_W33_50	14163.50	14160.00	14160.00	14163.25	14163.50	14163.25	14160.00	14160.00	14160.00	14160.00
kroB100_VT34_T33_W33_75	14768.00	14768.00	14768.00	14768.00	14768.00	14768.00	14768.00	14768.00	14768.00	14768.00
kroB150_VT30_T30_W90_25	11467.00	11467.00	11470.00	11473.00	11479.00	11476.00	11479.00	11479.00	11479.00	11500.00
kroB150_VT50_T50_W50_50	15626.00	15392.00	15394.00	15394.00	15418.00	15421.75	15404.75	15405.25	15415.25	15416.00
kroB150_VT30_T90_W30_75	21650.75	21606.25	21625.25	21576.00	21568.50	21520.25	21569.50	21560.25	21600.00	21558.75
kroB200_VT40_T120_W40_25	24042.25	24088.50	24070.00	24050.75	24022.00	24049.75	24045.50	24046.50	24027.50	24036.25
kroB200_VT40_T40_W120_50	14146.00	14146.00	14128.00	14128.25	14128.25	14120.25	14109.25	14123.75	14126.25	14129.25
kroB200_VT68_T66_W66_75	19589.75	19513.50	19546.25	19499.25	19472.50	19488.25	19512.00	19467.25	19488.00	19465.00
kroC100_VT20_T60_W20_25	16785.00	16785.00	16785.00	16785.00	16785.00	16785.00	16785.00	16785.00	16785.00	16785.00
kroC100_VT34_T33_W33_50	13297.00	13275.50	13235.00	13235.00	13235.00	13235.00	13235.00	13235.00	13235.00	13235.00
kroC100_VT20_T20_W60_75	12582.00	12582.00	12582.00	12582.00	12582.00	12582.00	12582.00	12582.00	12582.00	12582.00
kroE100_VT20_T60_W20_25	17199.00	17199.00	17199.00	17199.00	17205.25	17199.00	17199.00	17205.25	17199.00	17205.25
kroE100_VT20_T20_W60_50	10586.00	10586.00	10586.00	10586.00	10586.00	10586.00	10586.00	10586.00	10586.00	10586.00
kroE100_VT34_T33_W33_75	14622.00	14622.00	14622.00	14622.00	14622.00	14622.00	14622.00	14622.00	14622.00	14622.00

Tabela 5.1: Definição do valor de alfa - Grupo 1.

Instâncias Grupo 1										
Nome/Valores de Alfa	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
lin105_VT21_T21_W63_25	8893.00	8893.00	8893.00	8893.00	8893.00	8893.00	8893.00	8893.00	8893.00	8893.00
lin105_VT21_T63_W21_50	11583.00	11583.00	11583.00	11583.00	11583.00	11583.00	11583.00	11583.00	11583.00	11583.00
lin105_VT35_T35_W35_75	10566.00	10566.00	10566.00	10566.00	10566.00	10566.00	10566.00	10566.00	10566.00	10566.00
pr76_VT14_T16_W46_25	66007.00	66007.00	66007.00	66007.00	66007.00	66007.00	66007.00	66007.00	66007.00	66007.00
pr76_VT24_T26_W26_50	74539.00	74539.00	74540.25	74539.00	74539.00	74540.25	74539.00	74539.00	74540.25	74540.25
pr76_VT14_T46_W16_75	90862.00	90862.00	90862.00	90862.00	90862.00	90862.00	90862.00	90862.00	90862.00	90862.00
pr107_VT20_T65_W22_25	40994.00	40994.00	40994.00	40994.00	40994.00	40994.00	40994.00	40994.00	40994.00	40994.00
pr107_VT20_T22_W65_50	35869.00	35869.00	35869.00	35869.00	35869.00	35869.00	35869.00	35869.00	35869.00	35869.00
pr107_VT35_T36_W36_75	37324.00	37325.00	37314.00	37292.00	37292.00	37303.00	37325.00	37303.00	37314.00	37335.00
pr124_VT42_T41_W41_25	35465.00	35465.00	35465.00	35465.00	35465.00	35465.00	35465.00	35465.00	35465.00	35465.00
pr124_VT24_T75_W25_50	49160.00	49160.00	49160.00	49160.00	49160.00	49160.00	49160.00	49160.00	49160.00	49160.00
pr124_VT24_T25_W75_75	34359.00	34359.00	34359.00	34359.00	34359.00	34359.00	34359.00	34359.00	34359.00	34359.00
pr136_VT26_T28_W82_25	47620.00	47620.00	47620.00	47620.00	47620.00	47620.00	47620.00	47620.00	47620.00	47620.00
pr136_VT46_T45_W45_50	60044.00	60044.00	59862.25	59688.00	59615.50	59525.25	59447.25	59447.25	59449.50	59445.00
pr136_VT26_T82_W28_75	78730.75	78720.25	78648.50	78624.00	78624.00	78624.00	78624.00	78624.00	78624.00	78624.00
pr144_VT28_T87_W29_25	46940.00	46958.00	46952.00	46940.00	46946.00	46940.00	46940.00	46940.00	46940.00	46940.00
pr144_VT28_T29_W87_50	31183.00	30294.00	30294.00	30303.50	30513.00	30294.00	30294.00	30294.00	30294.00	30294.00
pr144_VT48_T48_W48_75	43713.00	43713.00	43713.00	43713.00	43713.00	43713.00	43746.75	43713.00	43713.00	43746.75
pr152_VT50_T51_W51_25	50667.00	50667.00	50667.25	50667.25	50667.00	50667.25	50667.00	50667.25	50667.00	50667.00
pr152_VT29_T92_W31_50	61068.00	61068.00	61068.00	61068.00	61068.00	61068.00	61068.00	61068.00	61068.00	61068.00
pr152_VT29_T31_W92_75	49110.00	49110.00	49110.00	49110.00	49110.00	49110.00	49110.00	49110.00	49110.00	49110.00
rat99_VT19_T20_W60_25	792.00	792.00	792.00	792.00	792.00	792.00	792.00	792.00	792.00	792.00
rat99_VT33_T33_W33_50	847.00	847.00	847.00	847.00	847.00	847.00	847.00	847.00	847.00	847.00
rat99_VT19_T60_W20_75	1017.00	1017.00	1017.00	1017.00	1017.00	1017.00	1017.00	1017.00	1017.00	1017.00

Tabela 5.1: Definição do valor de alfa - Grupo 1.

Instâncias Grupo 1										
Nome/Valores de Alfa	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
rat195_VT65_T65_W65_25	1606.50	1608.50	1610.75	1606.00	1606.75	1608.75	1606.75	1607.00	1608.50	1607.75
rat195_VT39_T39_W117_50	1199.00	1199.00	1199.00	1199.00	1199.00	1199.00	1199.00	1199.00	1200.00	1200.00
rat195_VT39_T117_W39_75	1948.00	1947.00	1938.00	1937.50	1937.50	1937.50	1931.25	1937.50	1935.75	1936.25
rd100_VT34_T33_W33_25	5552.00	5552.00	5552.00	5552.00	5552.00	5552.00	5552.00	5552.00	5552.00	5552.00
rd100_VT20_T60_W20_50	6657.00	6676.00	6658.75	6677.75	6639.75	6641.50	6676.00	6676.00	6666.75	6660.50
rd100_VT20_T20_W60_75	5427.00	5433.00	5421.00	5421.00	5421.00	5421.00	5421.00	5421.00	5421.00	5421.00
si175_VT35_T35_W105_25	4845.00	4845.00	4845.00	4845.00	4845.00	4845.00	4851.50	4845.00	4851.50	4845.00
si175_VT35_T105_W35_50	7034.50	7025.50	7030.00	7026.25	7036.75	7021.00	7026.00	7038.25	7034.75	7038.75
si175_VT59_T58_W58_75	8443.00	8431.00	8438.25	8444.00	8447.75	8453.75	8439.50	8436.75	8438.00	8428.50
st70_VT14_T14_W42_25	419.00	419.00	419.00	419.00	419.00	419.00	419.00	419.00	419.00	419.00
st70_VT14_T42_W14_50	554.00	554.00	554.00	554.00	554.00	554.00	554.00	554.00	554.00	554.00
st70_VT22_T24_W24_75	505.00	505.00	505.00	505.00	505.00	505.00	505.00	505.00	505.00	505.00
u159_VT31_T96_W32_25	35442.00	35442.00	35442.00	35442.00	35442.00	35442.00	35442.00	35442.00	35442.00	35442.00
u159_VT31_T32_W96_50	28613.00	28613.00	28613.00	28613.00	28613.00	28613.00	28613.00	28613.00	28613.00	28613.00
u159_VT53_T53_W53_75	33134.50	33100.50	33043.50	33043.50	33072.00	33072.00	33072.00	33072.00	33106.25	33106.00
Média Geral de Custo	20112.38	20096.37	20091.79	20089.10	20087.64	**20084.35**	**20084.28**	20085.34	**20084.28**	20086.20
Valor de alfa	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**

Tabela 5.2: Definição do valor de alfa - Grupo 2.

Instâncias Grupo 2										
Nome/Valores de Alfa	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
a280_VT94_T93_W93_25	1724.75	1724.00	1722.50	1723.75	1721.75	1720.00	1728.00	1725.50	1734.50	1730.00
a280_VT56_T168_W56_50	1992.50	1997.25	2003.00	2002.25	2000.50	2002.00	2001.50	2001.00	2001.50	2002.00
a280_VT56_T56_W168_75	1656.75	1652.50	1651.25	1649.50	1652.00	1652.75	1651.25	1650.50	1650.75	1651.50
gil262_VT51_T53_W158_25	1472.50	1471.00	1466.75	1468.25	1468.50	1471.50	1470.75	1467.00	1468.50	1470.50
gil262_VT88_T87_W87_50	1695.00	1692.00	1694.25	1681.50	1682.25	1680.75	1679.75	1680.00	1681.25	1680.00
gil262_VT51_T158_W53_75	2006.75	1999.00	2001.50	2000.00	2002.25	1994.50	2002.25	1999.00	2003.25	1999.25
gr202_VT39_T122_W41_25	32235.00	32190.00	32190.50	32172.00	32162.50	32163.00	32182.50	32165.50	32162.50	32162.00
gr202_VT68_T67_W67_50	28789.75	28753.00	28759.25	28755.25	28730.50	28738.75	28735.25	28735.75	28730.50	28735.00
gr202_VT39_T41_W122_75	25905.00	25905.00	25905.00	25905.00	25905.00	25905.00	25905.00	25905.00	25905.00	25905.00
gr229_VT45_T46_W138_25	61284.25	61284.25	61318.50	61304.50	61290.50	61296.75	61284.25	61278.00	61290.50	61324.75
gr229_VT45_T138_W46_50	96973.00	96896.25	96872.25	96931.00	96999.50	97016.75	97158.25	97490.25	97582.75	97540.25
gr229_VT77_T76_W76_75	90935.75	90844.75	90844.75	90726.75	90729.50	90785.75	90918.25	90892.75	90939.50	90997.75
lin318_VT63_T64_W191_25	16299.50	16297.00	16297.00	16297.00	16297.00	16297.00	16297.00	16297.00	16297.00	16297.00
lin318_VT108_T105_W105_50	23353.50	23325.50	23244.50	23314.75	23283.00	23304.00	23262.75	23266.75	23264.25	23265.75
lin318_VT63_T191_W64_75	33109.75	33079.25	32945.75	32900.50	32931.25	32899.50	32883.75	32873.50	32904.25	32898.25
pr226_VT76_T75_W75_25	57538.50	57538.50	57551.75	57528.00	57557.00	55967.50	55960.00	55842.00	55835.50	55842.00
pr226_VT44_T136_W46_50	67322.75	67410.75	67598.75	67679.25	67684.75	67710.50	67845.50	67717.25	67710.50	67645.75
pr226_VT44_T46_W136_75	51153.75	51079.25	51112.00	50844.50	50834.25	50798.00	50818.25	50821.00	50768.00	50768.00
pr264_VT52_T53_W159_25	27693.00	27693.00	27693.00	27693.00	27693.00	27693.00	27693.00	27693.00	27693.00	27693.00
pr264_VT88_T88_W88_50	34059.25	34086.75	34153.00	34073.00	34106.00	34101.00	34102.75	34090.00	34053.75	34066.75
pr264_VT52_T159_W53_75	45393.00	45367.50	45351.50	45364.75	45256.25	45291.75	45314.25	45215.75	45179.75	45323.25
pr299_VT59_T180_W60_25	35365.00	35062.50	34833.50	34802.00	34721.50	34728.75	34700.50	34696.25	34702.00	34687.50
pr299_VT59_T60_W180_50	20250.75	20194.75	20187.75	20191.75	20189.50	20190.00	20191.00	20198.00	20196.00	20227.00
pr299_VT101_T99_W99_75	30002.00	30057.50	30017.25	29984.75	29937.25	29789.00	29806.00	29812.00	29819.00	29754.50

Tabela 5.2: Definição do valor de alfa - Grupo 2.

Instâncias Grupo 2										
Nome/Valores de Alfa	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
rd400_VT80_T240_W80_25	10682.67	10640.67	10627.33	10621.00	10603.00	10615.67	10607.33	10586.00	10572.00	10566.00
rd400_VT80_T80_W240_50	5747.00	5747.00	5747.00	5747.00	5747.00	5747.00	5747.00	5747.00	5747.00	5747.00
rd400_VT136_T132_W132_75	8962.75	8911.75	8948.75	8913.75	8928.75	8928.75	8914.50	8936.25	8914.75	8920.50
ts225_VT45_T45_W135_25	88509.25	87889.00	87993.75	87889.00	87993.75	87993.75	88070.00	87993.75	88189.25	88146.75
ts225_VT45_T135_W45_50	109058.50	109068.00	109091.25	108890.50	108921.00	109022.00	108936.50	108835.50	108805.00	109022.00
ts225_VT75_T75_W75_75	97320.50	97268.25	97238.00	97211.75	97224.00	97322.00	97219.00	97205.50	97222.25	97179.50
tsp225_VT75_T75_W75_25	2676.00	2676.25	2676.25	2676.75	2676.75	2676.50	2677.00	2676.75	2676.50	2677.00
tsp225_VT75_T75_W75_50	2738.00	2739.50	2739.75	2739.25	2739.00	2737.75	2741.25	2740.25	2740.50	2737.50
tsp225_VT45_T45_W135_75	2449.00	2447.00	2448.25	2447.50	2448.25	2447.00	2447.00	2447.50	2447.50	2447.50
Média Geral de Custo	33828.95	33787.54	33785,62	33761.50	33761.12	**33717.82**	33725.80	**33717.61**	33723.89	33730.62
Valor de alfa	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**

5.4 Resultados e Comparações

Para cada abordagem, a deste trabalho, e as ILS-MRD_AD e ILS-MRD_IB de [35], em cada problema-teste, foram realizadas dez execuções, partindo-se de uma semente distinta. O único parâmetro de configuração necessário ao algoritmo GRASP proposto neste trabalho é o critério de parada, que foi definido da seguinte maneira: o método interrompe sua execução ou quando alcança o valor ótimo conhecido, ou quando gasta o mesmo tempo das versões ILS. Já em relação aos algoritmos de [35], os parâmetros foram utilizados exatamente com os mesmos valores relatados em [35]: o número máximo de iterações executadas pelo ILS ($iterMax\_ILS$) com valor 100; os graus mínimo, $kpMin$, e máximo, $kpMax$, utilizados no mecanismo de perturbação do ILS, com valores 5 e 10, respectivamente; o grau de incremento do $kpMin$ (δ) com valor 2; e, finalmente, o parâmetro que indica o número máximo de iterações do mecanismo de busca local MRD em cada uma das iterações ILS ($iterMax\_MRD$) com valor 350.

A Tabela 5.3 apresenta os resultados do experimentos realizados com as instâncias do Grupo 1, tanto para o algoritmo proposto neste trabalho, quanto para o ILS-MRD_AD [35]. Já os resultados relativos às instâncias do Grupo 2 constam na Tabela 5.4 para o algoritmo GRASP e para o ILS-MRD_IB [35]. Nessas tabelas, a primeira coluna indica o nome do problema-teste, a segunda coluna MelhorLit[2] mostra o melhor custo da literatura para a instância, a terceira coluna Algoritmo traz a indicação de quais algoritmos definiram o melhor valor relatado na segunda coluna, onde IB, VP, AD, DP e GE, representam respectivamente as versões ILS [35] com métodos de solução inicial: Inserção Mais Barata, Inserção Mais Próxima, ADD, DROP e GENIUS, e AE o representa o Algoritmo Evolutivo [35], o valor CPLEX indica que o valor foi definido pela formulação matemática proposta em [35], as colunas seguintes, Melhor Custo, Média, Tempo, são, respectivamente, para cada abordagem, o melhor custo encontrado pelo algoritmo, a média de custo das dez execuções e o tempo médio de execução, dado em segundos. Em seguida, a coluna *gap* mostra o desvio percentual da média de custo em relação a melhor valor conhecido. Por fim, a coluna DiffTemp, indica o percentual de redução de tempo de execução do algoritmo GRASP-GENIUS-VND em relação ao algoritmo correspondente proposto em [35] para cada instância, de acordo com o grupo. Valores destacados, em negrito, representam o melhor custo. O valor do *gap* é calculado pela equação (5.1).

$$gap = \frac{Media - MelhorLit}{MelhorLit} * 100 \tag{5.1}$$

[2]Valor definido pelo método exato ou algoritmos propostos em [35].

Tabela 5.3: Comparação dos resultados entre o GRASP-GENIUS-VND e o ILS-MRD_AD [35] - Grupo 1.

Instâncias Grupo 1										
Instância			ILS-MRD_AD				GRASP-GENIUS-VND				
Nome	MelhorLit	Algoritmo	Melhor	Média	Tempo	*gap*	Melhor	Média	Tempo	*gap*	DiffTemp
att48_VT9_T10_W29_25	2291	CPLEX	**2291**	2291.00	6.1209	**0**	**2291**	2291.10	0.9940	0.0044	**-83.8**
att48_VT9_T29_W10_50	2824	CPLEX	**2824**	2824.00	16.4692	0	**2824**	2824.00	0.5990	0	**-96.4**
att48_VT16_T16_W16_75	2650	CPLEX	**2650**	2650.00	27.9641	0	**2650**	2650.00	0.2180	0	**-99.2**
berlin52_VT9_T11_W32_25	4861	CPLEX	**4861**	4861.00	3.8738	0	**4861**	4861.00	0.2810	0	**-92.7**
berlin52_VT16_T18_W18_50	5186	CPLEX	**5186**	5186.00	45.7953	**0**	**5186**	5191.80	1.6760	0.1118	**-96.3**
berlin52_VT9_T32_W11_75	6916	CPLEX	**6916**	6916.00	115.0965	0	**6916**	6916.00	1.9690	0	**-98.3**
bier127_VT24_T77_W26_25	98453	CPLEX	**98453**	98453.00	2247.7793	**0**	**98453**	98512.10	1253.3850	0.0600	**-44.2**
bier127_VT24_T26_W77_50	74745	CPLEX	**74745**	74745.00	42.4118	0	**74745**	74745.00	1.1350	0	**-97.3**
bier127_VT43_T42_W42_75	98311	IB-VP-AD-DR-GE-AE	**98311**	98521.20	3470.2129	0.2138	**98311**	98320.30	569.8560	**0.0095**	**-83.6**
brazil58_VT18_T20_W20_25	19417	CPLEX	**19417**	19417.00	55.6378	0	**19417**	19417.00	0.5070	0	**-99.1**
brazil58_VT11_T35_W12_50	23312	CPLEX	**23312**	23312.00	8.6401	0	**23312**	23312.00	0.6320	0	**-92.7**
brazil58_VT11_T12_W35_75	20466	CPLEX	**20466**	20466.00	3.3714	0	**20466**	20466.00	0.1580	0	**-95.3**
brg180_VT36_T36_W108_25	550	IB-VP-AD-DR-GE-AE	**550**	551.00	552.0687	0.1818	**550**	550.00	12.3920	**0**	**-97.8**
brg180_VT60_T60_W60_50	2350	IB-VP-AD-DR-GE-AE	**2350**	2350.00	1078.3297	**0**	2360	2360.00	1082.1860	0.4255	0.4
brg180_VT36_T108_W36_75	2190	IB-VP-AD-DR-GE-AE	**2190**	2195.00	4681.7347	**0.2283**	2210	2210.00	4685.5320	0.9132	0.1
ch130_VT44_T43_W43_25	4383	IB-VP-AD-DR-GE-AE	**4383**	4385.50	1680.1484	0.0570	**4383**	4383.00	35.3610	**0**	**-97.9**
ch130_VT26_T26_W78_50	4139	CPLEX	**4139**	4139.00	92.9182	0	**4139**	4139.00	4.3220	0	**-95.3**
ch130_VT26_T78_W26_75	5287	IB-VP-AD-DR-GE-AE	**5287**	5295.40	9468.3194	0.1589	**5287**	5288.70	2444.4020	**0.0322**	**-74.2**
ch150_VT50_T50_W50_25	4778	IB-VP-AD-DR-GE-AE	**4778**	4786.30	4823.5730	0.1737	**4778**	4784.80	3037.0070	**0.1423**	**-37.0**
ch150_VT30_T90_W30_50	5239	IB-VP-AD-DR-GE-AE	**5239**	5245.00	10373.2316	0.1145	**5239**	5239.00	801.7200	**0**	**-92.3**
ch150_VT30_T30_W90_75	4292	CPLEX	**4292**	4292.00	107.7497	0	**4292**	4292.00	3.8860	0	**-96.4**
d198_VT39_T40_W119_25	10916	CPLEX	**10916**	10918.30	1187.3488	0.0211	**10916**	10916.00	18.5080	**0**	**-98.4**
d198_VT39_T119_W40_50	13598	IB-VP-AD-DR-GE-AE	**13598**	13613.20	18292.1363	0.1118	**13598**	13598.00	2470.6470	**0**	**-86.5**

Tabela 5.3: Comparação dos resultados entre o GRASP-GENIUS-VND e o ILS-MRD_AD [35] - Grupo 1.

Instâncias Grupo 1											
Instância			ILS-MRD_AD				GRASP-GENIUS-VND				
Nome	MelhorLit	Algoritmo	Melhor	Média	Tempo	*gap*	Melhor	Média	Tempo	*gap*	DiffTemp
d198_VT66_T66_W66_75	12905	IB-VP-AD-GE-AE	12935	12997.50	14042.0830	0.7168	12907	12910.30	14063.3680	**0.0411**	0.2
eil51_VT17_T17_W17_25	273	CPLEX	**273**	273.00	67.9350	0	**273**	273.00	0.8730	0	**-98.7**
eil51_VT9_T11_W31_50	248	CPLEX	**248**	248.00	3.6800	0	**248**	248.00	0.0430	0	**-98.8**
eil51_VT9_T31_W11_75	348	CPLEX	**348**	348.00	130.2269	0	**348**	348.00	18.4920	0	**-85.8**
eil76_VT14_T46_W16_25	387	IB-VP-AD-DR-GE-AE	**387**	387.00	56.1113	0	**387**	387.00	1.8230	0	**-96.8**
eil76_VT24_T26_W26_50	338	CPLEX	**338**	338.00	122.0825	0	**338**	338.00	1.4690	0	**-98.8**
eil76_VT14_T16_W46_75	332	CPLEX	**332**	332.00	4.3326	0	**332**	332.00	0.2520	0	**-94.2**
eil101_VT19_T21_W61_25	356	CPLEX	**356**	356.00	241.0444	0	**356**	356.00	4.5330	0	**-98.1**
eil101_VT19_T61_W21_50	506	IB-VP-AD-DR-GE-AE	**506**	506.20	1058.2507	0.0395	**506**	506.00	42.3250	**0**	**-96.0**
eil101_VT33_T34_W34_75	462	IB-VP-AD-DR-GE-AE	**462**	463.00	2947.0927	0.2165	**462**	462.10	1098.4960	**0.0216**	**-62.7**
gr48_VT16_T16_W16_25	3571	CPLEX	**3571**	3571.00	8.8902	0	**3571**	3571.00	0.0600	0	**-99.3**
gr48_VT9_T10_W29_50	3524	CPLEX	**3524**	3524.00	3.3075	0	**3524**	3524.00	0.0810	0	**-97.6**
gr48_VT9_T29_W10_75	4415	CPLEX	**4415**	4415.00	9.4485	**0**	**4415**	4426.10	1.1420	0.2514	**-87.9**
gr96_VT18_T58_W20_25	42746	CPLEX	**42746**	42746.00	69.6721	**0**	**42746**	42753.20	36.5860	0.0168	**-47.5**
gr96_VT32_T32_W32_50	35461	CPLEX	**35461**	35461.00	254.7702	0	**35461**	35461.00	7.9680	0	**-96.9**
gr96_VT18_T20_W58_75	30015	CPLEX	**30015**	30015.00	11.9085	0	**30015**	30015.00	0.7210	0	**-93.9**
gr120_VT40_T40_W40_25	4301	CPLEX	**4301**	4301.00	82.8328	0	**4301**	4301.00	1.4310	0	**-98.3**
gr120_VT24_T72_W24_50	5159	IB-VP-AD-DR-GE-AE	**5159**	5159.00	2020.7967	**0**	**5159**	5160.70	611.1400	0.0330	**-69.8**
gr120_VT24_T24_W72_75	4275	CPLEX	**4275**	4275.00	57.8840	0	**4275**	4275.00	0.7360	0	**-98.7**
gr137_VT26_T28_W83_25	39305	CPLEX	**39305**	39305.00	101.8566	0	**39305**	39305.00	1.5610	0	**-98.5**
gr137_VT45_T46_W46_50	46122	IB-VP-AD-DR-GE-AE	**46122**	46262.30	3410.3330	0.3042	**46122**	46122.00	110.0390	**0**	**-96.8**
gr137_VT26_T83_W28_75	58720	IB-VP-AD-DR-GE-AE	**58720**	58724.70	2383.1936	**0.0080**	**58720**	58729.40	524.0870	0.0160	**-78.0**
hk48_VT9_T29_W10_25	9708	CPLEX	**9708**	9708.00	41.5059	**0**	**9708**	9709.20	3.4710	0.0124	**-91.6**

Tabela 5.3: Comparação dos resultados entre o GRASP-GENIUS-VND e o ILS-MRD_AD [35] - Grupo 1.

Instâncias Grupo 1											
Instância			ILS-MRD_AD				GRASP-GENIUS-VND				
Nome	MelhorLit	Algoritmo	Melhor	Média	Tempo	*gap*	Melhor	Média	Tempo	*gap*	DiffTemp
hk48_VT16_T16_W16_50	7593	CPLEX	**7593**	7593.00	14.0107	0	**7593**	7593.00	0.4250	0	**-97.0**
hk48_VT9_T10_W29_75	7803	CPLEX	**7803**	7803.00	1.5332	0	**7803**	7803.00	0.1140	0	**-92.6**
kroA100_VT20_T20_W60_25	10848	CPLEX	**10848**	10848.00	52.1411	0	**10848**	10848.00	1.3030	0	**-97.5**
kroA100_VT20_T60_W20_50	16544	CPLEX	**16544**	16544.00	67.6739	**0**	**16544**	16559.00	11.8250	0.0907	**-82.5**
kroA100_VT34_T33_W33_75	14606	CPLEX	**14606**	14613.00	892.9801	0.0479	**14606**	14609.50	22.6090	**0.0240**	**-97.5**
kroA150_VT50_T50_W50_25	14382	VP	**14382**	14594.50	4297.4808	1.4775	**14382**	14388.20	1708.5520	**0.0431**	**-60.2**
kroA150_VT30_T30_W90_50	12846	IB-VP-AD-DR-GE-AE	**12846**	12857.40	1068.4973	**0.0887**	**12846**	12929.30	1006.7870	0.6485	**-5.8**
kroA150_VT30_T90_W30_75	21974	IB-VP-AD-DR-GE-AE	**21974**	22000.20	6454.7032	0.1192	**21974**	21975.20	1461.7130	**0.0055**	**-77.4**
kroA200_VT40_T120_W40_25	23495	IB-VP-AD-DR-GE-AE	**23495**	23519.30	10632.3575	0.1034	**23495**	23495.00	859.9360	**0**	**-91.9**
kroA200_VT68_T66_W66_50	17297	IB-VP-AD-DR-GE-AE	**17297**	17358.80	8353.0250	0.3573	**17297**	17323.10	7389.9500	**0.1509**	**-11.5**
kroA200_VT40_T40_W120_75	15197	CPLEX	**15197**	15201.00	2460.1220	0.0263	**15197**	15197.00	61.0380	**0**	**-97.5**
kroB100_VT20_T60_W20_25	18406	CPLEX	**18406**	18406.00	134.3905	0	**18406**	18406.00	9.1840	0	**-93.2**
kroB100_VT34_T33_W33_50	14160	IB-VP-AD-DR-GE-AE	**14160**	14171.70	2516.9982	0.0826	**14160**	14160.00	147.0100	**0**	**-94.2**
kroB100_VT34_T33_W33_75	14768	CPLEX	**14768**	14768.00	89.4750	0	**14768**	14768.00	13.8420	0	**-84.5**
kroB150_VT30_T30_W90_25	11467	IB-VP-AD-DR-GE-AE	**11467**	11475.40	1673.4331	0.0733	**11467**	11469.40	730.7470	**0.0209**	**-56.3**
kroB150_VT50_T50_W50_50	15388	IB-VP-AD-DR-GE	15417	15484.90	5330.5427	0.6297	**15388**	15388.80	1624.7850	**0.0052**	**-69.5**
kroB150_VT30_T90_W30_75	21489	IB-DR-GE-AE	**21489**	21549.00	7287.7058	0.2792	**21489**	21495.90	1294.6350	**0.0321**	**-82.2**
kroB200_VT40_T120_W40_25	23946	IB-VP-AD-DR-GE-AE	**23946**	23976.70	12877.3430	**0.1282**	23947	23999.00	12886.1500	0.2213	0.1
kroB200_VT40_T40_W120_50	14125	IB-VP-AD-DR-GE-AE	**14125**	14152.10	3022.6695	0.1919	**14103**	14114.50	858.0330	**-0.0743**	**-71.6**
kroB200_VT68_T66_W66_75	19458	IB-VP-AD	19471	19495.80	11836.5838	0.1943	**19458**	19463.60	11197.8388	**0.0288**	**-5.4**
kroC100_VT20_T60_W20_25	16785	CPLEX	**16785**	16785.00	55.8082	0	**16785**	16785.00	2.5560	0	**-95.4**
kroC100_VT34_T33_W33_50	13235	CPLEX	**13235**	13238.40	821.5178	0.0257	**13235**	13235.00	8.7790	**0**	**-98.9**
kroC100_VT20_T20_W60_75	12582	CPLEX	**12582**	12582.00	14.7710	0	**12582**	12582.00	0.3130	0	**-97.9**

Tabela 5.3: Comparação dos resultados entre o GRASP-GENIUS-VND e o ILS-MRD_AD [35] - Grupo 1.

Instâncias Grupo 1											
Instância			ILS-MRD_AD				GRASP-GENIUS-VND				
Nome	MelhorLit	Algoritmo	Melhor	Média	Tempo	*gap*	Melhor	Média	Tempo	*gap*	DiffTemp
kroE100_VT20_T60_W20_25	17199	CPLEX	**17199**	17199.00	345.2853	**0**	**17199**	17208.50	36.2819	0.0552	**-89.5**
kroE100_VT20_T20_W60_50	10586	CPLEX	**10586**	10586.00	16.1647	0	**10586**	10586.00	0.5160	0	**-96.8**
kroE100_VT34_T33_W33_75	14622	CPLEX	**14622**	14622.00	182.4946	0	**14622**	14622.00	18.7680	0	**-89.7**
lin105_VT21_T21_W63_25	8893	CPLEX	**8893**	8893.00	22.6181	0	**8893**	8893.00	1.6520	0	**-92.7**
lin105_VT21_T63_W21_50	11583	CPLEX	**11583**	11583.00	160.5502	0	**11583**	11583.00	11.9320	0	**-92.6**
lin105_VT35_T35_W35_75	10566	CPLEX	**10566**	10566.00	566.8494	0	**10566**	10566.00	72.8640	0	**-87.1**
pr76_VT14_T16_W46_25	66007	CPLEX	**66007**	66007.00	5.8705	0	**66007**	66007.00	0.1780	0	**-97.0**
pr76_VT24_T26_W26_50	74539	CPLEX	**74539**	74539.00	87.0389	**0**	**74539**	74553.00	15.0710	0.0188	**-82.7**
pr76_VT14_T46_W16_75	90862	CPLEX	**90862**	90862.00	90.3793	0	**90862**	90862.00	1.2730	0	**-98.6**
pr107_VT20_T65_W22_25	40994	IB-VP-AD-DR-GE-AE	**40994**	41014.00	1627.0405	0.0488	**40994**	40994.00	5.0820	**0**	**-99.7**
pr107_VT20_T22_W65_50	35869	CPLEX	**35869**	35869.00	22.7854	0	**35869**	35869.00	0.1970	0	**-99.1**
pr107_VT35_T36_W36_75	37292	CPLEX	**37292**	37325.60	1339.9952	0.0901	**37292**	37292.00	129.9614	**0**	**-90.3**
pr124_VT42_T41_W41_25	35465	IB-VP-AD-DR-GE-AE	**35465**	36107.90	1497.2719	1.8128	**35465**	35465.00	7.3310	**0**	**-99.5**
pr124_VT24_T75_W25_50	49160	IB-VP-AD-DR-GE-AE	**49160**	49164.40	1283.3561	0.009	**49160**	49160.00	74.5970	**0**	**-94.2**
pr124_VT24_T25_W75_75	34359	CPLEX	**34359**	34359.00	201.9400	0	**34359**	34359.00	1.7670	0	**-99.1**
pr136_VT26_T28_W82_25	47620	CPLEX	**47620**	47620.00	15.4324	0	**47620**	47620.00	0.7030	0	**-95.4**
pr136_VT46_T45_W45_50	59445	IB-VP-AD-DR-GE-AE	**59445**	59445.00	927.9911	**0**	**59445**	59447.70	422.0645	0.0045	**-54.5**
pr136_VT26_T82_W28_75	78624	IB-VP-AD-DR-GE-AE	**78624**	78624.00	940.5976	**0**	**78624**	78715.60	149.5312	0.1165	**-84.1**
pr144_VT28_T87_W29_25	46940	IB-VP-AD-DR-GE-AE	**46940**	46954.40	2720.7621	0.0307	**46940**	46940.00	107.7410	**0**	**-96.0**
pr144_VT28_T29_W87_50	30294	CPLEX	**30294**	30294.00	148.7562	**0**	**30294**	30297.80	9.5619	0.0125	**-93.6**
pr144_VT48_T48_W48_75	43713	IB-VP-AD-DR-GE-AE	**43713**	43868.60	3465.5045	0.3560	**43713**	43713.00	194.7820	**0**	**-94.4**
pr152_VT50_T51_W51_25	50667	IB-VP-AD-DR-GE-AE	**50667**	50667.20	735.8009	0.0004	**50667**	50667.20	155.1855	0.0004	**-78.9**
pr152_VT29_T92_W31_50	61068	IB-VP-AD-DR-GE-AE	**61068**	61077.70	2463.8883	0.0159	**61068**	61068.00	17.7250	**0**	**-99.3**

Tabela 5.3: Comparação dos resultados entre o GRASP-GENIUS-VND e o ILS-MRD_AD [35] - Grupo 1.

Instâncias Grupo 1											
Instância			**ILS-MRD_AD**				**GRASP-GENIUS-VND**				
Nome	**MelhorLit**	**Algoritmo**	**Melhor**	**Média**	**Tempo**	***gap***	**Melhor**	**Média**	**Tempo**	***gap***	**DiffTemp**
pr152_VT29_T31_W92_75	49110	CPLEX	**49110**	49110.00	47.6692	**0**	**49110**	49149.90	25.7086	0.0812	**-46.1**
rat99_VT19_T20_W60_25	792	CPLEX	**792**	792.00	74.6514	0	**792**	792.00	0.7580	0	**-99.0**
rat99_VT33_T33_W33_50	847	CPLEX	**847**	847.00	217.5816	0	**847**	847.00	2.4340	0	**-98.9**
rat99_VT19_T60_W20_75	1017	CPLEX	**1017**	1017.00	240.1827	0	**1017**	1017.00	22.8540	0	**-90.5**
rat195_VT65_T65_W65_25	1605	IB-VP-AD-DR-GE-AE	**1605**	1608.50	10686.4877	0.2181	**1605**	1606.30	5485.3899	**0.0810**	**-48.7**
rat195_VT39_T39_W117_50	1199	IB-VP-AD-DR-GE-AE	**1199**	1202.80	3308.1028	0.3169	**1199**	1199.00	49.4010	**0**	**-98.5**
rat195_VT39_T117_W39_75	1928	IB-VP-AD-DR-GE-AE	**1928**	1933.70	14847.8798	0.2956	**1928**	1928.80	10593.9149	**0.0415**	**-28.7**
rd100_VT34_T33_W33_25	5552	IB-VP-AD-DR-GE-AE	**5552**	5552.00	169.7170	0	**5552**	5552.00	1.5330	0	**-99.1**
rd100_VT20_T60_W20_50	6619	IB-VP-AD-DR-GE-AE	**6619**	6619.00	454.9698	**0**	**6619**	6626.60	196.9430	0.1148	**-56.7**
rd100_VT20_T20_W60_75	5421	CPLEX	**5421**	5421.00	35.5695	0	**5421**	5421.00	0.4290	0	**-98.8**
si175_VT35_T35_W105_25	4845	CPLEX	**4845**	4863.60	4252.2682	0.3839	**4845**	4849.40	97.6970	**0.0908**	**-97.7**
si175_VT35_T105_W35_50	6871	VP-AD-GE-AE	**6871**	6914.50	24299.8595	**0.6331**	7006	7007.00	24310.0180	1.9793	0.0
si175_VT59_T58_W58_75	8402	AE	8414	8448.30	21086.4769	0.5511	**8400**	8411.30	18804.9190	**0.1107**	**-10.8**
st70_VT14_T14_W42_25	419	CPLEX	**419**	419.00	10.8191	0	**419**	419.00	0.2360	0	**-97.8**
st70_VT14_T42_W14_50	554	IB-VP-AD-DR-GE-AE	**554**	554.00	92.5482	0	**554**	554.00	2.0110	0	**-97.8**
st70_VT22_T24_W24_75	505	CPLEX	**505**	505.00	53.6511	0	**505**	505.00	1.3280	0	**-97.5**
u159_VT31_T96_W32_25	35442	IB-VP-AD-DR-GE-AE	**35442**	35442.00	1721.6027	0	**35442**	35442.00	36.7250	0	**-97.9**
u159_VT31_T32_W96_50	28613	CPLEX	**28613**	28613.00	93.0355	0	**28613**	28613.00	4.3430	0	**-95.3**
u159_VT53_T53_W53_75	33015	IB-VP-AD-DR-GE-AE	**33015**	33056.60	2106.4349	**0.1260**	**33015**	33062.80	563.1840	0.1448	**-73.3**
Média Geral				**20092.91**	**2415.6200**	**0.1014**		**20079.81**	**1224.2413**	**0.0553**	**-81.4**

Tabela 5.4: Comparação dos resultados entre o GRASP-GENIUS-VND e o ILS-MRD_IB [35] - Grupo 2.

Instâncias Grupo 2											
Instância			**ILS-MRD_IB**				**GRASP-GENIUS-VND**				
Nome	**MelhorLit**	**Algoritmo**	**Melhor**	**Média**	**Tempo**	***gap***	**Melhor**	**Média**	**Tempo**	***gap***	**DiffTemp**
a280_VT94_T93_W93_25	1717	IB-VP-AD-DR-GE-AE	**1717**	1727.00	18566.2433	0.5824	**1717**	1718.30	15907.6640	**0.0757**	**-14.3**
a280_VT56_T168_W56_50	1985	IB-VP-AD-DR-GE-AE	**1985**	2001.60	23096.3252	0.8363	1996	1999.90	23135.1520	**0.7506**	0.2
a280_VT56_T56_W168_75	1647	IB-VP-AD-DR-GE-AE	**1647**	1649.00	8477.2677	0.1214	**1647**	1648.10	3337.2970	**0.0668**	**-60.6**
gil262_VT51_T53_W158_25	1465	IB-VP-AD-DR-GE-AE	**1465**	1466.80	3833.7540	**0.1229**	**1465**	1469.50	1755.7570	0.3072	**-54.2**
gil262_VT88_T87_W87_50	1678	IB-VP-AD-DR-AE	1680	1687.00	24499.1909	0.5364	**1678**	1678.30	15308.0250	**0.0179**	**-37.5**
gil262_VT51_T158_W53_75	1981	IB-VP-AD-DR-GE-AE	1987	1996.20	27911.7090	0.7673	1987	1989.90	27943.2700	**0.4493**	0.1
gr202_VT39_T122_W41_25	32162	IB-VP-AD-DR-GE-AE	**32162**	32174.60	10152.6355	0.0392	**32162**	32162.00	2252.9956	**0**	**-77.8**
gr202_VT68_T67_W67_50	28706	IB-VP-AD-GE-AE	**28706**	28752.40	7067.0414	0.1616	**28706**	28743.30	6518.8519	**0.1299**	**-7.8**
gr202_VT39_T41_W122_75	25905	IB-VP-AD-DR-GE-AE	**25905**	25934.40	1805.0639	0.1135	**25905**	25905.00	60.8320	**0**	**-96.6**
gr229_VT45_T46_W138_25	61278	IB-VP-AD-AE	61359	61379.70	3036.1000	0.1660	**61278**	61278.00	502.3508	**0**	**-83.5**
gr229_VT45_T138_W46_50	96707	IB-VP-AD-DR-GE-AE	**96707**	97087.20	22889.0579	0.3931	**96707**	96882.30	18552.1489	**0.1813**	**-18.9**
gr229_VT77_T76_W76_75	90701	AE	90712	91706.80	20065.3146	1.1089	**90701**	90704.30	10145.8130	**0.0036**	**-49.4**
lin318_VT63_T64_W191_25	16297	IB-VP-AD-DR-GE-AE	**16297**	16302.00	4552.5604	0.0307	**16297**	16297.00	269.9780	**0**	**-94.1**
lin318_VT108_T105_W105_50	23177	IB-GE	23292	23708.40	20372.6361	2.2928	23215	23303.00	20532.5140	**0.5436**	0.8
lin318_VT63_T191_W64_75	32790	IB-VP-AD-GE-AE	32857	33005.30	61751.4218	0.6566	**32790**	32855.20	59378.7612	**0.1988**	**-3.8**
pr226_VT76_T75_W75_25	55774	AE	55812	57827.50	8021.0541	3.6818	55812	55922.20	8042.2270	**0.2657**	0.3
pr226_VT44_T136_W46_50	67286	IB-VP-AD-DR-GE-AE	**67286**	67294.20	12470.6132	**0.0122**	**67286**	67557.70	10527.0080	0.4038	**-15.6**
pr226_VT44_T46_W136_75	50701	IB-VP-AD-DR-GE-AE	**50701**	50712.70	1762.4736	**0.0231**	**50701**	50823.90	1643.8280	0.2424	**-6.7**
pr264_VT52_T53_W159_25	27693	IB-VP-AD-DR-GE-AE	**27693**	27693.00	570.7435	0	**27693**	27693.00	27.2210	0	**-95.2**
pr264_VT88_T88_W88_50	34057	AE	34120	34173.60	17875.2342	0.3424	**34041**	34080.10	12346.6690	**0.0678**	**-30.9**
pr264_VT52_T159_W53_75	45020	GE	45092	45285.80	28103.3798	0.5904	45057	45200.20	28145.7320	**0.4003**	0.2
pr299_VT59_T180_W60_25	34671	IB-VP-AD-DR-GE-AE	34675	34856.70	29917.5395	0.5356	**34671**	34676.20	13382.9230	**0.0150**	**-55.3**
pr299_VT59_T60_W180_50	20181	VP-DR-GE-AE	20186	20269.90	5444.5786	0.4405	20184	20187.30	5458.7700	**0.0312**	0.3

Tabela 5.4: Comparação dos resultados entre o GRASP-GENIUS-VND e o ILS-MRD_IB [35] - Grupo 2.

Instâncias Grupo 2											
Instância			**ILS-MRD_IB**				**GRASP-GENIUS-VND**				
Nome	**MelhorLit**	**Algoritmo**	**Melhor**	**Média**	**Tempo**	***gap***	**Melhor**	**Média**	**Tempo**	***gap***	**DiffTemp**
pr299_VT101_T99_W99_75	29697	DR-AE	29875	30036.40	29928.1539	1.1429	29711	29807.60	30052.0260	**0.3724**	0.4
rd400_VT80_T240_W80_25	10523	VP-AD-DR-AE	**10523**	10562.20	59432.1341	**0.3725**	10543	10573.10	59694.6240	0.4761	0.4
rd400_VT80_T80_W240_50	5747	IB-VP-AD-DR-GE-AE	**5747**	5754.60	31233.5760	0.1322	**5747**	5747.00	1104.9200	**0**	**-96.5**
rd400_VT136_T132_W132_75	8849	AE	8912	9175.70	61552.4986	3.6919	8900	8924.40	61739.1298	**0.8521**	0.3
ts225_VT45_T45_W135_25	87889	IB-VP-AD-DR-GE-AE	**87889**	87889.00	1872.4797	0	**87889**	87889.00	163.6087	0	**-91.3**
ts225_VT45_T135_W45_50	108805	IB-VP-AD-DR-GE-AE	**108805**	109052.30	13747.0766	0.2273	**108805**	108823.40	5113.7750	**0.0169**	**-62.8**
ts225_VT75_T75_W75_75	96959	IB	97187	97644.10	19956.9162	0.7066	**96959**	97064.60	17822.8410	**0.1089**	**-10.7**
tsp225_VT75_T75_W75_25	2676	IB-VP-AD-DR-GE-AE	**2676**	2678.90	13993.6405	0.1084	**2676**	2676.00	2425.6570	**0**	**-82.7**
tsp225_VT75_T75_W75_50	2732	AE	2736	2746.10	18366.3018	0.5161	2733	2734.50	18378.1700	**0.0915**	0.1
tsp225_VT45_T45_W135_75	2447	IB-VP-AD-DR-GE-AE	**2447**	2449.60	3833.8732	0.1063	**2447**	2447.20	201.3340	**0.0082**	**-94.7**
Média Geral				**33838.81**	**18671.4724**	**0.6230**		**33680.65**	**14602.1780**	**0.1842**	**-37.5**

Os resultados apresentados nas Tabelas 5.3 e 5.4 são resumidos nas Tabelas 5.5 e 5.6 para os problemas-teste do Grupo 1 e do Grupo 2. Nessas tabelas, a coluna Algoritmo é nome da abordagem testada. A coluna Custo apresenta o custo médio em todas as instâncias em cada grupo. O tempo médio em segundos é apresentado na coluna Tempo. Já o desvio percentual médio e a quantidade de soluções em que o algoritmo encontra o melhor valor conhecido estão nas colunas *gap* e MelhorLit. Na última linha, tem-se as diferenças percentuais calculadas entre os resultados obtidos pelos algoritmos GRASP-GENIUS-VND e os algoritmos ILS-MRD_AD, para o Grupo 1, e ILS-MRD_IB, para o Grupo 2. Quando os valores aparecem negativos são favoráveis à abordagem deste trabalho.

Tabela 5.5: Resumo da comparação entre as abordagens testadas - Grupo 1.

Algoritmo	**Custo**	**Tempo (s)**	***gap* (%)**	**MelhorLit (qtd)**
ILS-MRD_AD	20092.91	2415.6200	0.1014	106
GRASP-GENIUS-VND	20079.81	1224.2413	0.0553	106
	-0.07%	**-49.32%**	**-45.47%**	

De acordo com os dados das Tabelas 5.3 e 5.5, observa-se que, em relação aos problemas-teste do Grupo 1, o algoritmo GRASP-GENIUS-VND teve desempenho superior ao da abordagem ILS-MRD_AD. Isto se deve ao fato de que, embora as soluções encontradas pelas duas abordagens tenham sido em geral de boa qualidade, o GRASP-GENIUS-VND obteve *gap* médio de 0.0553%, ou seja, reduziu ainda o desvio médio geral em 45.47% em relação ao alcançado pelo ILS-MRD_AD. O algoritmo GRASP proposto apresentou melhor *gap* em 33% das instâncias do Grupo 1, empatou em 47% das instâncias com o algoritmo ILS testado e perdeu somente em 20% das instâncias. Pode-se ainda verificar que, o algoritmo GRASP-GENIUS-VND apresentou uma expressiva diminuição nos tempos computacionais de execução, visto que a redução foi em média de 81.4%. O algoritmo GRASP deste trabalho foi o mais rápido em 95.5% dos problemas-teste do Grupo 1.

Em relação aos custos de solução, a média geral entre a duas abordagens foi muito semelhante, com diferença de apenas 0.07%. A taxa de sucesso, que significa o percentual de instâncias onde o algoritmo encontra o melhor valor conhecido, foi a mesma tanto para o algoritmo GRASP, quanto para o algoritmo ILS-MRD_AD. As duas abordagens alcançaram, no Grupo 1, o melhor valor conhecido em 106 das 111 instâncias, ou seja, taxa de sucesso igual a 95.5%, como pode ser observado na Tabela 5.5.

Tabela 5.6: Resumo da comparação entre as abordagens testadas - Grupo 2.

Algoritmo	Custo	Tempo (s)	*gap* (%)	MelhorLit (qtd)
ILS-MRD_IB	33838.81	18671.4724	0.6230	18
GRASP-GENIUS-VND	33680.65	14602.1780	0.1842	23
	-0.47%	**-21.79%**	**-70.44%**	

Em relação aos problemas-teste do Grupo 2, mostrados nas Tabelas 5.4 e 5.6, diferentemente do Grupo 1, verifica-se uma expressiva melhoria na qualidade de soluções encontradas pelo algoritmo GRASP-GENIUS-VND em relação ao ILS-MRD_IB, demonstrada pela redução de 70.44% no *gap* médio. O algoritmo ILS obteve *gap* de 0.6230%. Já o GRASP alcançou *gap* de 0.1842%. Das 33 instâncias do Grupo 2, o algoritmo GRASP-GENIUS-VND obteve valor de *gap* melhor que ILS-MRD_IB em 27 problemas-teste, ou seja, em 81.8% das instâncias. Empataram em apenas duas instâncias. Nas demais, o ILS-MRD_IB foi superior. Em relação ao custo de solução, no Grupo 2 o algoritmo GRASP proposto alcançou o melhor valor conhecido em 23 instâncias, enquanto que a abordagem ILS testada obteve 18. Ou seja, a taxa de sucesso alcançada pelo algoritmo GRASP foi de 69.7%, enquanto a versão ILS obteve 54.5%.

Em relação ao tempo computacional médio de execução, analogamente às instâncias do Grupo 1, o algoritmo GRASP-GENIUS-VND mostrou-se mais eficiente que o ILS-MRD_IB também com as instâncias do Grupo 2. Nesse caso, a redução foi, em média, de 37.5%. O algoritmo GRASP foi mais rápido em 23 das 33 instâncias desse grupo.

Deve-se ainda observar que, o algoritmo GRASP-GENIUS-VND, mesmo tendo como alvo uma solução específica, estabeleceu novo limite superior para três instâncias, a $kroB200\_VT40\_T40\_W120\_50$ e $si175\_VT59\_T58\_W58\_75$, do Grupo 1, com novo custo de 14103 e 8400, respectivamente, e a $pr264\_VT88\_T88\_W88\_50$, do Grupo 2, com novo custo de 35041.

Supõe-se que a superioridade do algoritmo proposto neste trabalho em relação às abordagens comparadas, no que diz respeito aos critérios de tempo computacional e de *gap*, pode estar relacionada às modificações realizadas tanto na heurística GENIUS, quanto nas estruturas de vizinhança para atender às especificidades do PRRCP, diferentemente dos algoritmos propostos em [35], que permitem o relaxamento de algumas das restrições do PRRCP durante a fase de busca local. Assim, com soluções iniciais de boa qualidade e com movimentos mais direcionados na fase de busca local, alcançam-se mais rapidamente a soluções de boa qualidade.

5.5 Significância Estatística

Nesta seção, será feita uma verificação se os ganhos referentes a média de solução, reportados nas Tabelas 5.3 e 5.4 têm significância estatística, realizando o teste estatístico não paramétrico de Friedman [33]. Esse teste é geralmente aplicado para avaliar dois algoritmos que possuem componentes aleatórias, e identificar se a diferença entre as médias de resultados obtidos pelos algoritmos foi, de fato, devido à superioridade de algum deles ou simplesmente devido à aleatoriedade dos métodos. Para realizar o teste, foram considerados p-valor igual a 0.05 e duas hipóteses:

- Hipótese nula (H0): Não há diferença entre as médias encontradas pelos algoritmos comparados; e
- Hipótese alternativa (H1): Há diferença entre as médias encontradas pelos algoritmos comparados.

Dessa forma, H0 poderá ser rejeitada com 95% de certeza se, para cada instância, o valor retornado pelo teste de Friedman para a comparação entre os algoritmos for menor ou igual ao p-valor definido. Caso H0 seja rejeitada, a hipótese alternativa H1 é considerada.

A Tabela 5.7 mostra uma comparação, entre o algoritmo GRASP-GENIUS-VND proposto e as versões de ILS, usando o pacote R [26], separados por grupos de instâncias. O número fora dos parênteses indica em quantas instâncias aquela estratégia foi melhor que a outra em relação à média de solução, enquanto o valor entre parênteses indica o número de vezes em que o p-valor foi menor que 0.05, indicando que a probabilidade de a diferença de desempenho dos algoritmos ser devido à aleatoriedade é menor que 5%.

Tabela 5.7: Análise de Significância Estatística

Par de Algoritmos	**Grupo 1**	**Grupo 2**
GRASP-GENIUS-VND	37(20)	—
ILS-MRD_AD	22(5)	—
GRASP-GENIUS-VND	—	27(16)
ILS-MRD_IB	—	4(2)

Na comparação entre os algoritmos GRASP-GENIUS-VND e ILS-MRD_AD para o Grupo 1, nota-se que a maioria das vezes em que o GRASP-GENIUS-VND foi superior ao ILS-MRD_AD teve significância estatística, isto é, das 37 vezes que o algoritmo proposto

superou o ILS-MRD_AD, 20 das vitórias tiveram significância estatística. O mesmo não ocorreu em relação ao ILS-MRD_AD, isto é, das 22 vezes em que o algoritmo de [35] foi superior ao GRASP-GENIUS-VND, somente 5 delas tiveram significância estatística.

Quando é realizada a comparação entre os algoritmos GRASP-GENIUS-VND e ILS-MRD_IB para o Grupo 2, observa-se que, analogamente ao Grupo 1, a maioria das vezes em que o GRASP-GENIUS-VND foi superior ao ILS-MRD_IB teve significância estatística, isto é, das 27 vezes que o algoritmo proposto obteve melhores médias, 16 das vitórias tiveram significância estatística. Já para o ILS-MRD_IB somente metade das vitórias em relação ao GRASP-GENIUS-VND teve significância estatística.

Outro teste estatístico foi usado para comprovar que a heurística GRASP-GENIUS-VND é melhor que as versões de ILS de [35]. O teste de *Wilcoxon* é comumente usado quando são analisadas duas amostras independentes, possivelmente de tamanhos diferentes, e se deseja utilizar um teste estatístico para rejeitar a hipótese nula de que não existem diferenças significativas entre essas duas amostras, com um nível de significância α [33]. Como esse teste também faz uso de hipóteses, as mesmas hipóteses feitas para o Teste de Friedman foram usadas nessa análise.

Considerando os resultados obtidos para o Grupo 1, usando o pacote R [26], é possível rejeitar H0 com $\alpha = 1.17 \times 10^{-8}$. Já para o Grupo 2, é possível rejeitar H0 com $\alpha = 2.2 \times 10^{-16}$. Portanto, com uma probabilidade maior que 99%, pode-se concluir que existem diferenças significativas entre as soluções encontradas pela heurística GRASP-GENIUS-VND proposta e as versões de ILS de [35].

5.6 Análise do Comportamento das Estratégias

A fim de comprovar a eficiência da abordagem proposta neste trabalho, foram realizados experimentos para verificar a distribuição de probabilidade empírica. Para isto, utilizou-se o método descrito em [1], que consiste em realizar n execuções com os algoritmos, cada uma com uma semente diferente, onde o algoritmo é interrompido ao encontrar um valor alvo definido. Em seguida, deve-se ordenar os tempos $T = \{t_1, t_2, ..., t_n\}$ que cada algoritmo encontrou o valor alvo. O próximo passo é calcular a probabilidade acumulada, dada por $p_i = (i - 1/2)/n$, e associá-la ao i-ésimo tempo. Finalmente, plota-se a curva usando os pontos $z_i = (t_i, p_i)$.

A Figura 5.1 apresenta o gráfico da distribuição da probabilidade para o problema-teste *gr96_VT18_T58_W20_25*, após 100 execuções dos algoritmos GRASP-GENIUS-

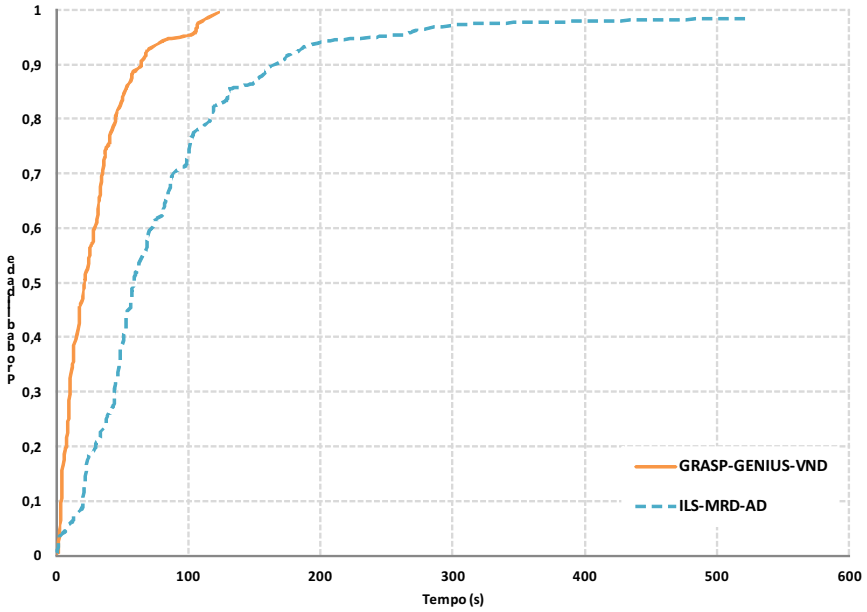

Figura 5.1: Distribuição da probabilidade acumulada para instância *gr96_VT18_T58_W20_25*, do Grupo 1, com alvo 42746.

VND e ILS-MRD_AD, utilizando-se como valor alvo 42746, que é o valor exato para essa instância. Nesse gráfico, o eixo das coordenadas mostra a probabilidade acumulada de um algoritmo encontrar a solução alvo, enquanto os tempos de execução estão no eixo das abscissas.

De acordo com a Figura 5.1, pode-se observar que o algoritmo GRASP-GENIUS-VND alcança o valor alvo com probabilidade em torno de 95% já aos 100 segundos de execução, enquanto o algoritmo ILS-MRD_AD, com esse mesmo tempo, acumula, apenas, um pouco mais de 70% de probabilidade de alcançar a solução alvo. Verifica-se ainda que, o algoritmo GRASP-GENIUS-VND requer em torno de 120 segundos para chegar a quase 100% de probabilidade de encontrar o valor alvo, diferentemente do ILS-MRD_AD que, em algumas execuções extrapolou os 500 segundos de tempo de execução.

Na Figura 5.2 tem-se a comparação das probabilidades acumuladas para a instância *pr264_VT52_T53_W159_25* obtidas pelos algoritmos GRASP-GENIUS-VND e ILS-MRD_IB, do Grupo 2, cujo alvo estabelecido, 27693, é o melhor valor conhecido para essa instância. Pode-se verificar na Figura 5.2 que, claramente, o algoritmo GRASP-GENIUS-VND é bem mais rápido que o ILS-MRD_IB. Isto porque o GRASP-GENIUS-VND requer, aproximadamente, 205 segundos para alcançar o valor alvo com probabilidade acumulada acima de 99%. Por outro lado, o ILS-MRD_IB, nesse mesmo tempo, alcança o alvo com pouco menos de 30% de probabilidade. Conforme já descrito, as instâncias

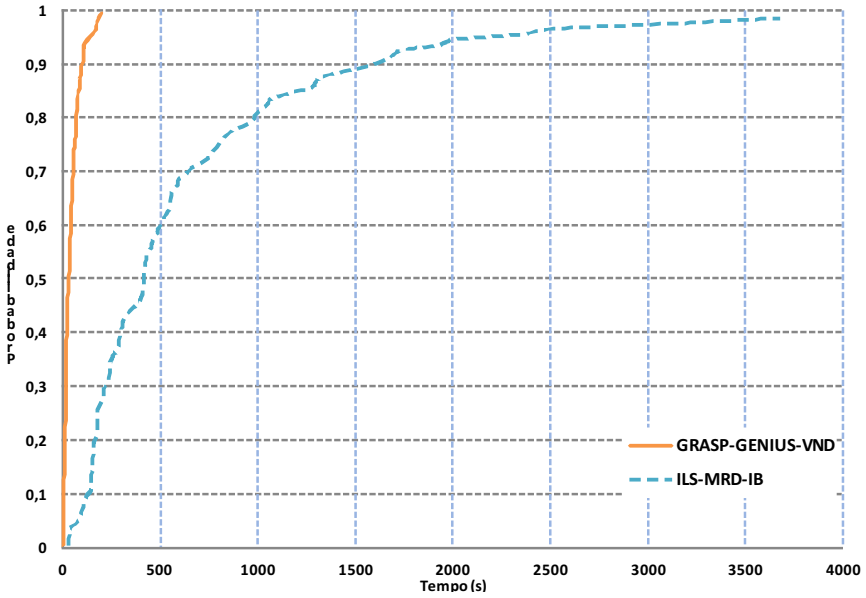

Figura 5.2: Distribuição da probabilidade acumulada para instância *pr264_VT52_T53_W159_25*, do Grupo 2, com alvo 27693.

do Grupo 2 tem quantidade de vértices acima de 200 vértices, fato este que pode ter influenciado os tempos requeridos pelo ILS-MRD_IB, que embora tenha alcançado o tempo médio aproximado de 624 segundos, chega a 98% de probabilidade somente após 3600 segundos. Supõe-se que o mecanismo de perturbação de solução, característico da metaheurística ILS, pode fazer como que a fase de busca local necessite de mais tempo para reorganizar a grande quantidade vértices possivelmente afetados pela perturbação e, então, chegar a uma boa solução ou ao ótimo local.

5.7 Considerações Finais

De fato as Figuras 5.1 e 5.2 mostram que o algoritmos GRASP-GENIS-VND e os algoritmos ILS-MRD_AD e ILS-MRD_IB são bem semelhantes no que diz respeito à efetividade em encontrar soluções alvo de boa qualidade com alta probabilidade, visto que nos dois casos testados, ambos passam dos 96%. Porém há uma drástica diferença na eficiência ao se considerar o tempo necessário para alcançar a solução alvo. Isto é bem evidenciado no gráfico da Figura 5.2 e também nas Tabelas 5.3 e 5.4. Nessas tabelas, pode-se observar que, para mais de 52.8% das instâncias o GRASP-GENIUS-VND apresenta redução no tempo computacional com índice acima de 90% em relação aos tempos registrados pelos algoritmos de [35]. A expressiva competitividade do algoritmo proposto neste trabalho em relação às abordagens da literatura testadas, também é demonstrada no desvio médio de

qualidade de soluções onde, no Grupo 1, o GRASP-GENIUS-VND obteve *gap* de 0.0553% e o ILS-MRD_AD obteve um *gap* de 0.1014%. Já no Grupo 2, o GRASP-GENIUS-VND obteve um *gap* de 0.1842%, enquanto que o ILS-MRD_IB obteve um *gap* de 0.6230%, ou seja, foi verificada uma redução acima de 70%, como pode ser observado nas Tabelas 5.5 e 5.6.

De acordo com resultados obtidos e relatados neste trabalho pode-se verificar que a nova heurística híbrida para o PRRCP proposta neste trabalho é bem competitiva em relação às versões de algoritmo de buscal local ILS com aos quais foi comparada. O novo algoritmo possui uma estrutura simplificada com apenas dois parâmetros. Além disso, o algoritmo GRASP-GENIUS-VND consegue gerar soluções de boa qualidade com expressiva redução do esforço computacional, conforme demonstrado nas tabelas e gráficos deste capítulo.

No próximo capítulo, as conclusões finais, de acordo com resultados obtidos, e as sugestões de trabalhos futuros são apresentadas.

Capítulo 6

Conclusões e Trabalhos Futuros

Neste trabalho foi realizado um estudo sobre o problema de recobrimento de rotas com coletas de prêmios (PRRCP). Esse problema pode ser definido em um grafo não direcionado e simétrico $G = (V \cup W, E)$, onde $V \cup W$ é o conjunto dos vértices e E é o conjunto das arestas. O conjunto V é composto de nós obrigatórios T e opcionais $V \backslash T$. A cada vértice de V tem-se um prêmio não negativo associado. O conjunto W representa os vértices que devem ser cobertos por algum vértice do subconjunto V. Assim, o PRRCP consiste em encontrar um ciclo hamiltoniano de custo mínimo que contenha todos os vértices obrigatórios de T e que todos os vértices de W estejam cobertos, realizando uma coleta mínima de prêmios.

Devido o PRRCP ser da classe NP-difícil, foi proposto um algoritmo heurístico híbrido denominado GRASP-GENIUS-VND para sua resolução aproximada. Neste algoritmo, faz-se a hibridização da metaheurística GRASP com a heurísica GENIUS, como método de solução inicial, e o método VND, como mecanismo de busca local. Adaptou-se a heurística GENIUS, assim como um conjunto de estruturas de vizinhanças, às especificidades do PRRCP.

A validação do algoritmo proposto deu-se por meio de experimentos computacionais realizados sobre dois grupos de instâncias da literatura, em um total de 144 problemas-testes e cujos os resultados foram confrontados com os algoritmos ILS-MRD_AD e ILS-MRD_IB, ambos propostos em [35]. Foram executadas, no mesmo ambiente computacional, a implementação descrita neste trabalho e as versões dos algoritmos desenvolvidos em [35], disponibilizado pelo autor para realização dos testes. Para o Grupo 1, que contém 111 instâncias, o GRASP-GENIUS-VND conseguiu encontrar soluções com qualidade igual ou superior aos encontrados na literatura em 95.5% das instâncias, com redução da média geral de tempo computacional de 49.32% e diminuição do *gap* em 45.47%. Já

no Grupo 2, composto de 33 instâncias, o GRASP-GENIUS-VND apresentou expressiva redução do *gap* geral, obtendo um valor de *gap* 70.44% inferior ao registrado pela abordagem ILS-MRD_IB. A redução do tempo médio geral foi de 21.79%. Vale ressaltar que o algoritmo proposto neste trabalho requer apenas dois parâmetros, o critério de parada e o α, diferentemente das abordagens baseadas em ILS.

De acordo com resultados apresentados, pode-se concluir que o algoritmo GRASP-GENIUS-VND, proposto neste trabalho, é competitivo em relação às abordagens comparadas diante dos resultados obtidos nos problemas-teste disponíveis na literatura, com expressiva eficiência tanto em qualidade de solução (*gap*), quanto em tempo computacional.

Como trabalho futuro sugere-se a incorporação de novas técnicas de hibridização, como a implantação de memória de longo prazo, por meio mineração de dados, por exemplo. A aplicação de reconexão por caminhos também é outra proposta viável. Outra possibilidade é investir no desenvolvimento de novas estruturas de vizinhança, ou no aprimoramento das já desenvolvidas, para o problema obter soluções com melhores custos. Assim como, a implantação de mecanisno dinâmico de escolha do valor do α, conhecido como Grasp Reativo [24]. Sugere-se ainda, a aplicação do algoritmo desenvolvido neste trabalho a outros problemas correlacionados ao problema do caixeiro viajante que sejam semelhantes ao PRRCP, tais como, *Prize Collecting Traveling Salesman Problem* (PCTSP) [2] e *Shortest Covering Path Problem* (SCPP) [5].

Referências

[1] Aiex, R. M., Resende, M. G. C., Ribeiro, C. C. TTTplots: A perl program to create time-to-target plots. *Optimization Letters 1* (2006), 10 – 1007.

[2] Balas, E. The prize collecting traveling salesman problem. *Networks 19* (1989), 621 – 636.

[3] Claus, A. A new formulation for the traveling salesman problem. *SIAM Journal of Algorithm and Discrete Mathematics 5* (1984), 21–25.

[4] Current, J. R. Multiobjective design of transportation networks. *Tese de Doutorado* (1981).

[5] Current, J. R., Revelle, C., Cohon, J. The shortest covering path problem: An application of locational constraints to network design. *Journal of Regional Science 24* (1984), 161– 183.

[6] Dorigo, M., Stutzle, T. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

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

[8] Feo, T. A., Resende, M. G. C. Greedy randomized adaptive search procedures. *Jornal of Global Optimization 6* (1995), 109–133.

[9] Gendrau, M. An introduction to tabu search. In *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Kluwer Academic Publishers, Norwell, MA, 2003, p. 37–54.

[10] Gendreau, M., Hertz, A., Laporte, G. New insertion and postoptimization procedures for the traveling salesman problem. *Operational Research 40* (1992), 1086–1094.

[11] Gendreau, M., Laporte, G., Semet, F. The covering tour problem. *Operations Reaserch 45* (1995), 569–576.

[12] Glover, F. Tabu search and adaptive memory programing - advances, applications and challenges. In *Interfaces in Computer Science and Operations Research* (1996), Kluwer, p. 1–75.

[13] Gutin, G., Punnen, A. P. *The Traveling Salesman Problem and Its Variations*, vol. 12 of *Combinatorial Optimization*. Kluwer, Dordrecht, 2002.

[14] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., Shmoys, D. B. *The Traveling Salesman Problem.* Wiley, New York, 1985.

[15] Lourenço, H. R., Martin, O. C., Stutzle, T. Iterated local search. In *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Kluwer Academic Publishers, Norwell, MA, 2003, p. 321–353.

[16] Lyra, A. R. O Problema de Recobrimento de Rotas com Coleta de Prêmios: Regras de Redução, Formulação Matemática e Heurísticas. *Dissertação de Mestrado, Universidade Federal Fluminense* (2004).

[17] Maniezzo, V., Baldacci, R., Boschetti, M., Zamboni, Z. Scatter search methods for the covering tour problem. *Scienze dell'Informazione, University of Bologna* (1999).

[18] Mladenovic, N., Hansen, P. Variable neighboorhood search. *Computers and Operations Research 24* (1997), 1097–1100.

[19] Mladenovic, N., Hansen, P. First improvement may be better than best improvement - an empirical study. *Les Cahiers du GERAD G-99-40* (1999).

[20] Mladenovic, N., Hansen, P. Variable neighboorhood search. In *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Kluwer Academic Publishers, Norwell, MA, 2003, p. 145–184.

[21] Motta, L. C. S. Novas Abordagens para o Problema de Recobrimento de Rotas. *Dissertação de Mestrado, Universidade Federal Fluminense* (2001).

[22] Ochi, L. S., Silva, M. B., Drummond, L. Metaheuristics based on GRASP and VNS for solving the traveling purchaser problem. *MIC'2001 - 4th Metaheuristic Internacional Conference* (2001), 489–492.

[23] Prais, M. Estratégias de variação de parâmetros em procedimentos GRASP e aplicações. *Tese de Doutorado, Pontifícia Univesidade Católica do Rio de Janeiro, Departamento de Informática* (2000).

[24] Prais, M., Ribeiro, C. C. Parameter variation in GRASP procedures. *Investigación Operativa 9* (2000), 1–20.

[25] Prais, M., Ribeiro, C. C. Reactive GRASP: an application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing 12* (2000), 164–176.

[26] R Project. The R project for statistical computing, 2014. `http://www.r-project.org/`, última visita em 31/03/2014.

[27] Reeves, C. Genetic algorithms. In *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Kluwer Academic Publishers, Norwell, MA, 2003, p. 55–81.

[28] Reinelt, G. TSPLIB - a traveling salesman problem library. *ORSA - Journal on Computing* (1991), 376–384.

[29] Reinelt, G. *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer Verlag, Heidelberg, Berlim, 1994.

[30] Resende, M. G. C., Ribeiro, C. C. Greedy randomized adaptive search procedures. In *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Kluwer Academic Publishers, Norwell, MA, 2003, p. 219–250.

[31] Resende, M. G. C., Silva, R. M. A. GRASP: Procedimentos de busca gulosos, aleatórios e adaptativos. *Metaheurística em Pesquisa Operacional* (2013), 1–20.

[32] Rosseti, I. Estratégias sequenciais e paralelas de GRASP com reconexão por caminhos para o problema de síntese a 2-caminhos. *Tese de Doutorado, Pontifícia Univesidade Católica do Rio de Janeiro, Departamento de Informática* (2003).

[33] Siegel, S., Castellan Jr, N. J. *Nonparametric Statistics for the Behavioral Sciences*, 2^a ed. McGraw-Hill, 1988.

[34] Silva, D. M. Uma Heurística para o Problema de Roteamento de Veículos com Múltiplas Viagens. *Dissertação de Mestrado, Universidade Federal Fluminense* (2012).

[35] Silva, M. S. A. Problema de Recobrimento de Rotas com Coleta de Prêmios. *Dissertação de Mestrado, Universidade Federal Fluminense* (2009).

[36] Silva, M. S. A., Mine, M. T., Ochi, L. S., Souza, M. J. F. Um Algoritmo Evolutivo Híbrido para O Problema de Recobrimento de Rotas com Coleta De Prêmios. In *Learning and Nonlinear Models - Revista da Sociedade Brasileira de Redes Neurais (SBRN)* (2010), p. 100–110.

[37] Wong, R. T. Integer programming formulations of the traveling salesman problem. In *IEEE Conference on Circuits and Computers* (1980), p. 149–152.