Heurística Híbrida GRASP-GENIUS-VND aplicada ao Problema de Recobrimento de Rotas com Coletas de Prêmios

Rogério da Silva
Instituto Federal de Educação Ciência e Tecnologia do Piauí (IFPI)
Praça da Liberdade, 1597 - Centro - CEP 64000-040 - Teresina/PI - Brasil
rogerio.silva@ifpi.edu.br

Isabel Rosseti
Instituto de Computação - Universidade Federal Fluminense (UFF)
Rua Passo da Pátria, 156 - Bloco E - CEP 24210-240 - Niterói/RJ - Brasil
rosseti@ic.uff.br

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 de 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. 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. Neste trabalho propõe-se um algoritmo para o PRRCP, baseado na metaheurística GRASP e hibridizado com as heurísticas GENIUS, na fase construtiva, e VND na fase de busca local. Os resultados dos experimentos sobre 144 instâncias da literatura comprovaram que a nova abordagem desenvolvida mostrou-se competitiva em relação a heurística considerada estado da arte.

PALAVRAS CHAVE. Metaheurística Híbrida, PRRCP, Otimização Combinatória, Área de classificação principal (Metaheurística).

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 up new hybrid method outperformed the state-of-the-art heuristic.

KEYWORDS. Hybrid Metaheuristic, PRRCP, Combinatorial Optimization, Main area (Metaheuristic).

1. Introdução

A prestação de serviços de assistência social, médica ou jurídica a comunidades geograficamente distribuídas e distantes dos grandes centros urbanos, por meio de equipes itinerantes, é uma das aplicações do problema de recobrimento de rotas com coletas de prêmios (PRRCP). O PRRCP é um problema $\mathcal{NP}$-difícil (Lyra, 2004) e é 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 de 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 V da solução. Entende-se por cobertura a existência na solução de pelo menos um vértice V que esteja a uma distância igual ou inferior a distância D. 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.

O estudo de técnicas para solucionar o PRRCP é de alta relevância, pois vários problemas reais são facilmente caracterizados como instâncias do PRRCP. No caso do exemplo da disponibilização de equipes itinerantes, os vértices de T seriam as cidades ou os populosos bairros periféricos que são definidos como pontos de visita obrigatória. Já as cidades ou bairros um pouco menores são potencialmente ordenados como candidatos a pontos de parada, sendo, portanto, os vértices opcionais $(V \backslash T)$. Existe ainda, um último conjunto de cidades ou de bairros, onde a população deve obrigatoriamente ser atendida em cidades (ou bairros) vizinhas a no máximo uma distância D, evitando grandes deslocamentos dos cidadãos. Esses últimos pontos seriam os vértices do conjunto W. A população de cada cidade dos conjuntos T e $V \backslash T$ e a quantidade total mínima de cidadãos a ser atendida, são, respectivamente, o prêmio de cada vértice e o prêmio total a ser coletado.

Embora exista uma variedade de possíveis aplicações em problemas reais na área de logística operacional, o PRRCP só foi investigado em dois trabalhos recentes. Em Lyra (Lyra, 2004) o PRRCP foi proposto. Foram apresentados uma formulação matemática e um algoritmo aproximado baseado na metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*) (Feo e Resende, 1989), combinada com uma variação da metaheurística VNS (*Variable Neighbourhood Search*) (Mladenovic e Hansen, 1997), para resolver o problema. Foi incorporada também a técnica de reconexão de caminhos (Glover, 1996) como elemento intensificador de soluções da heurística GRASP proposta. Lyra validou sua proposta gerando instâncias aleatórias em um plano cartesiano 100×100, que não foram disponibilizadas na literatura até o momento. Os experimentos mostraram que essa abordagem encontrou o ótimo global para a maioria dos casos onde ele era conhecido.

Em Silva (Silva, 2009), trabalho considerado o estado da arte para o PRRCP, foram propostos uma nova formulação matemática e seis algoritmos heurísticos, cinco baseados na metaheurística *Iterated Local Search* (ILS) (Lourenço *et al.*, 2003), combinada com cinco diferentes métodos de geração de solução inicial (ADD, DROP, inserção mais próxima, inserção mais barata, GENIUS (Gendreau *et al.*, 1992)), e um baseado em algoritmos evolutivos, combinado com a técnica de reconexão de caminhos como estratégia de intensificação de soluções. Silva gerou dois conjuntos de problemas-testes, a partir das instâncias do repositório TSPLIB (Reinelt, 1991). O primeiro, denominado Grupo 1, continha instâncias com até 200 vértices. Já o segundo, conhecido como Grupo 2, era composto de instâncias de 201 até, no máximo, 400 vértices. Os experimentos mostraram que os algoritmos baseados em ILS encontraram o valor ótimo ou o melhor limite superior em mais

de 95% das instâncias do Grupo 1 e em pelo menos 70% das instâncias do Grupo 2. Já o algoritmo evolutivo encontrou o valor ótimo ou melhor valor conhecido em 99.1% das instâncias do Grupo 1 e em 91% do Grupo 2. No entanto, o algoritmo evolutivo despendeu, em média, o dobro do tempo gasto pelas heurísticas baseadas em ILS e, de acordo com (Silva, 2009), tornou-se uma abordagem impraticável.

O objetivo deste trabalho é a proposta de um novo algoritmo aproximado eficiente para resolver o PRRCP baseado na metaheurística GRASP, utilizando a abordagem GENIUS, adaptada ao PRRCP, na fase construtiva, e o método de descida em vizinhança variável (VND) (Mladenovic e Hansen, 1997) na fase de busca local. Os resultados dos experimentos sobre as 144 instâncias descritas em (Silva, 2009) comprovaram que a nova abordagem mostrou-se competitiva em relação a heurística considerada estado da arte.

Este trabalho é organizado como segue. A Seção 2 descreve a abordagem proposta, isto é, a hibridização da heurística GRASP para o PRRCP com GENIUS e VND. Os resultados dos experimentos e comparações com outras abordagens são apresentados na Seção 3. Já na Seção 4 são relatadas as conclusões e sugeridos alguns trabalhos futuros.

2. Algoritmo Proposto: GRASP-GENIUS-VND

A metaheurística GRASP é um procedimento multipartida e iterativo para problemas de otimização combinatória proposto em (Feo e Resende, 1989), onde cada iteração é composta de duas fases: construção e busca local. Na fase de construção uma solução viável S para o problema é construída, geralmente elemento a elemento. Uma vez obtida uma solução viável, essa solução tem sua vizinhança $N(S)$ explorada até chegar ao ótimo local $S^{'}$, na fase de busca local. Por fim, a melhor solução encontrada S^{*} é retornada como resultado do algoritmo GRASP, quando o critério de parada predefindo é atingido.

Algoritmo 1 Algoritmo GRASP padrão

Parâmetros $criterio\_parada$, $seed$, α

1: $S^{*} \leftarrow \emptyset$;
2: **Enquanto** não atinjir $criterio\_parada$ **faça**
3: $S \leftarrow solucao\_inicial\_semigulosa(seed, \alpha)$;
4: $S^{'} \leftarrow busca\_local(S)$;
5: **Se** $(f(S^{'}) < f(S^{*}))$ **então**
6: $S^{*} \leftarrow S^{'}$;
7: **Fim-se**
8: **Fim-enquanto**
9: **Retorne** S^{*};

No Algoritmo 1 tem-se o pseudocódigo da estrutura básica de um algoritmo GRASP de minimização, onde $seed$ é o valor de inicialização do gerador de números pseudoaleatórios e α é a componente auxiliar usada para criar a lista restrita de candidatos (LRC). A cada iteração, o algoritmo de construção gera um solução viável S tendo como base uma lista de candidatos (LC). Em LC os itens estão ordenados seguindo o critério de valor associado, isto é, pelo incremento de custo na solução. LC é, então, construída seguindo uma função adaptativa gulosa. Adicionamente, no GRASP uma componente probabilística é adicionada, onde a partir de LC é construída a LRC. Em seguida, seleciona-se aleatoriamente da LRC o elemento a fazer parte da solução.

Algoritmo 2 Algoritmo de Construção Semi-Gulosa

Parâmetros $seed$, 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: $c^{min} \leftarrow \{c(e), e \in LC \mid c(e) \leq c(e^{'}) \; \forall e^{'} \in LC\}$
6: $c^{min} \leftarrow \{c(e), e \in LC \mid c(e) \geq c(e^{'}) \; \forall e^{'} \in LC\}$
7: $LRC \leftarrow \{e \in C \mid c(e) \leq c^{min} + \alpha(c^{max} - c^{min})\}$
8: $k \leftarrow selecao\_aleatoria(LRC, seed)$
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;

Para resolução do PRRCP foi proposto um algoritmo, chamado de 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 GENIUS, como método de geração de solução inicial, e VND como mecanismo de exploração da vizinhança das soluções geradas. Assim, buscou-se unir a simplicidade da implementação de GRASP ao elaborado método de construção de GENIUS e a flexibilidade de VND em explorar diversas estruturas de vizinhanças.

Nesta nova heurística o critério de parada usado é ou o encontro de uma solução prefixada, ou o alcance de um tempo máximo de execução previamente estabelecido. No Algoritmo 3, a estrutura geral do GRASP proposto é apresentada. 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 adaptado para atender às restrições do PRRCP e aos princípios de GRASP, e em seguida a rota inicial PRRCP obtida é informada, 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 verifica-se se a solução alvo predefinida (ou uma melhor) foi alcançada. Se o resultado desse teste for verdadeiro, o algoritmo é interrompido retornando essa solução (linha 9). Na linha 12 atualiza-se o tempo de execução.

2.1. Heurística GENIUS Aplicada ao PRRCP

Proposta em (Gendreau *et al.*, 1992) para solucionar o problema do caixeiro viajante (PCV) (Lawler *et al.*, 1985), a heurística GENIUS original é composta de dois métodos: uma abordagem de construção, a fase GENI *(Generalized Insertion)* e um método de melhoria, a fase US *(Unstringing and Stringing)*. A fase GENI, responsável pela criação de uma solução viável, possui dois métodos diferentes de inserção de vértices. Analogamente, a fase US possui dois procedimentos de remoção de vértices da rota e invoca os métodos de inserção de GENI para, eventualmente, reposicionar os vértices na solução.

O Algoritmo 4 apresenta o pseudocódigo da heurística GENIUS Semi-Gulosa pro-

Algoritmo 3 GRASP-GENIUS-VND

Parâmetros $tempo\_max$, $alvo$, $seed$, α

1: $S^* \leftarrow \emptyset$; $tempo\_execucao \leftarrow 0$;
2: **Enquanto** $tempo\_execucao < tempo\_max$ **faça**
3: $S \leftarrow solucao\_inicial\_semigulosa\_GENIUS(seed, \alpha)$;
4: $S' \leftarrow busca\_local\_VND(S, seed)$;
5: **Se** $(f(S') < f(S^*))$ **então**
6: $S^* \leftarrow S'$;
7: **Se** $(f(S^*) <= alvo)$ **então**
8: **Retorne** S^*;
9: **Fim-se**
10: **Fim-se**
11: $tempo\_execucao \leftarrow obter\_tempo\_execucao()$;
12: **Fim-enquanto**
13: **Retorne** S^*;

posta neste trabalho como método de geração de solução inicial para o algoritmo GRASP. GENIUS foi adaptado para ser usado na resolução do PRRCP. Primeiramente, o critério que define a integralidade de uma solução, ou seja, quando esse método deve ser interrompido é composto pelas seguintes restrições: (a) conter todos os vértices T; (b) cobrir todos os vértices W; e (c) coletar o prêmio exigido na instância.

Considerando-se as especificidades do PRRCP, a lista de candidatos LC e consequentemente a LRC são inicialmente compostas exclusivamente de vértices T. Posteriormente, a LC e a LRC passam a ser compostas apenas por vértices pertencentes a $V \backslash T$ que cubram vértices W ainda descobertos, mesmo com todos os vértices de T já inseridos. E por fim, os demais vértices (isto é, pertencentes ao conjunto $V \backslash T$) são definidos como candidatos, a fim de coletar o prêmio requerido por uma dada instância.

Para tornar algoritmo GENIUS um procedimento semiguloso, adicionou-se a componente probabilística por meio do uso do parâmetro α para composição da LRC. Assim, os vértices são ordenados em LC de acordo com a previsão de custo incremental na solução, observando-se as características dos métodos de inserção da fase GENI.

2.2. Método VND Aplicado ao PRRCP

De acordo com (Mladenovic e Hansen, 1997), o método de Descida em Vizinhança Variável (VND) é uma heurística de busca local que tem como fundamento básico a exploração do espaço de soluções por meio da troca sistemática de estruturas de vizinhança. A ideia principal do VND é que um ótimo local para uma certa estrutura de vizinhança não é necessariamente o ótimo local para uma outra estrutura de vizinhança.

A busca local do algoritmo GRASP proposto é guiada pelo método VND, que percorre o espaço de soluções em cada uma das 15 estruturas de vizinhança, todas elas adaptadas ao PRCCP, em busca do ótimo global. As vizinhanças estão divididas em intra-rota, cujos movimentos envolvem exclusivamente vértices que já pertencem à rota, alterando apenas as suas posições dentro da solução, e as extra-rota, em que seus movimentos utilizam o subconjunto de vértices opcionais que não pertecem à rota, isto é, $V \backslash T \notin S$. As estruturas de vizinhança são classificadas, ainda, em movimentos simples, que são os movimentos clássicos do PCV, e em movimentos compostos, que utilizam as técnicas dos

Algoritmo 4 Algoritmo Semi-Guloso GENIUS

Parâmetros $seed$, α

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

métodos de inserção de GENIUS e do método de inserção mais barata para gerar vizinhos de S. As dez vizinhanças intra-rota usadas são as seguintes:

1. *shift*: troca um vértice de posição dentro da rota;
2. *swap*: 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;
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, 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.

As cinco vizinhanças extra-rota usadas são as que seguem:

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.

A cada chamada do VND, as estruturas de vizinhança são ordenadas de maneira aleatória. Neste trabalho usou-se a técnica da primeira melhoria *(First Improvement)* (Mladenovic e Hansen, 1997), na qual a exploração da vizinhança é interrompida assim que uma solução que apresente qualquer melhoria de custo em relação à solução atual seja alcançada, retornando-se à primeira estrutura de vizinhança. O algoritmo termina quando as 15 vizinhanças forem percorridas e não foi possível melhorar a solução corrente.

3. Experimentos e Resultados Computacionais

Esta seção apresenta os resultados computacionais resumidos obtidos pelo algoritmo descrito na Seção 2 (Os resultados completos podem ser encontrados em (Silva, 2014)). O objetivo dos experimentos é comparar a eficiência, principalmente em relação à qualidade das soluções encontradas (*gap*), obtido pelo algoritmo proposto neste trabalho, diante dos resultados alcançados pelas melhores versões de algoritmos de busca local ILS desenvolvidos em (Silva, 2009), quando se disponibiliza o mesmo ambiente computacional. O valor do *gap* é calculado pela seguinte equação:

$$gap = \frac{Media - MelhorLit}{MelhorLit} * 100 \quad (1)$$

Dentre as cinco versões de algoritmos ILS propostos em (Silva, 2009), foram utilizadas, para realizar as comparações desta seção, 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 heurísticas ADD e inserção mais barata, respectivamente, como método de geração de solução inicial.

Os experimentos computacionais foram executados em ambiente com computador dotado de processador $IntelCore^{TM}$ i5 CPU 650 @ 3.2GHz, com 4GB de RAM e sistema operacional Linux Fedora 15. 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 implementado utilizando-se a linguagem de programação C e compilado com o g++, versão 4.7.0. Já os algoritmos baseados em ILS, desenvolvidos em (Silva, 2009) e disponibilizados pelo autor, foram implementados na linguagem de programação *C*++. As duas abordagens foram compiladas para execução em ambiente com arquitetura de 64 bits.

3.1. Definição do Valor de α

Conforme descrito na Seção 2, um dos aspectos da metaheurística GRASP é seu comportamento probabilístico, definindo, na fase de construção de solução inicial, uma lista com os melhores candidatos, a LRC, onde o critério de valor associado pode ser utilizado para definir quais vértices candidatos pertencerão a essa lista. Nesta abordagem proposta, a LRC é composta por todos os elementos $e \in LC$ cujo custo $c(e)$ pertence ao intervalo $[c^{min}, c^{min} + \alpha(c^{max} - c^{min})]$.

A fim de definir um valor de α satisfatório para o GRASP proposto neste trabalho, foram realizadas quatro execuções prévias para cada uma das instâncias, tanto do Grupo 1, quanto do Grupo 2, utilizando-se diferentes sementes em cada um delas. 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.

Tabela 1. Resultados do teste para escolha do valor de Alfa (α)

Grupo/Alfa	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Grupo 1 - média	20112.38	20096.37	20091.79	20089.10	20087.64	**20084.35**	**20084.28**	20085.34	**20084.08**	20086.20
Grupo 2 - média	33828.75	33787.67	33785,67	33761.33	33760.79	**33717.58**	33725.80	**33717.40**	33723.77	33730.70

Analisando-se os dados resumidos na Tabela 1, pode-se verificar que, para o Grupo 1, obtiveram melhor valor médio de custo as soluções obtidas com os valores de α iguais a 0.6, 0.7 e 0.9. Já para o Grupo 2, tiveram melhor desempenho os algoritmos usando os valores de α iguais a 0.6 e 0.8. Como o valor de α igual a 0.6 obteve um desempenho satisfatório para ambos os grupos, ele foi escolhido para ser usado na definição da LRC.

3.2. Resultados e Comparações

Para cada abordagem, a deste trabalho, e as ILS-MRD_AD e ILS-MRD_IB de (Silva, 2009), em cada instância, foram realizadas dez execuções, partindo-se de uma semente distinta em cada execução. 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 gastar o mesmo tempo das versões baseadas em ILS. Os parâmetros das abordagens ILS utilizados foram exatamente os mesmos valores relatados em (Silva, 2009).

Nas Tabelas 2 e 3 são apresentados os resultados resumidos obtidos para as instâncias do Grupo 1 e do Grupo 2. A coluna **Algoritmo** é nome da abordagem testada. A coluna **Média Custo** apresenta o custo médio em todas as instâncias em cada grupo. O tempo médio em segundos é apresentado na coluna **Tempo (s)**. 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 pelo GRASP-GENIUS-VND e o ILS-MRD_AD, para o Grupo 1, e o ILS-MRD_IB, para o Grupo 2. Os resultados são favoráveis à nova abordagem quando os valores da última linha dessas tabelas são negativos.

De acordo com os dados da Tabela 2, 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 valores de custos de solução encontrados, 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, uma taxa de sucesso igual a 95.5%.

Tabela 2. Resumo da comparação entre as abordagens testadas para o Grupo 1.

Algoritmo	Média Custo	Tempo (s)	*gap* (%)	# MelhorLit
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%**	

Em relação aos problemas-teste do Grupo 2, mostrados nas Tabela 3, 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 aos valores de custos de solução encontrados, 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 os problemas-teste do Grupo 2. Nesse caso, a redução foi, em média, de 37.5%. O algoritmo GRASP conseguiu ser mais rápido em 23 das 33 instâncias desse grupo.

Tabela 3. Resumo da comparação entre as abordagens testadas para o Grupo 2.

Algoritmo	Custo	Tempo (s)	*gap* (%)	# MelhorLit
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%**	

Deve-se ainda salientar 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 a $si175\_VT59\_T58\_W58\_75$, 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.

3.3. Significância Estatística

Nesta seção, é realizada uma verificação se os ganhos referentes a média de solução, reportados nas Tabelas 2 e 3, têm significância estatística, aplicando o teste estatístico não paramétrico de Friedman (Siegel e Castellan Jr, 1988). Esse teste é geralmente usado 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 duas hipóteses e o $p\text{-}valor$ igual a 0.05: (a) a hipótese nula (H0), isto é, quando não há diferença entre as médias encontradas pelos algoritmos comparados; e (b) a hipótese alternativa (H1), isto é, quando existem diferenças entre as médias encontradas. 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 $p\text{-}valor$ definido. Caso H0 seja rejeitada, a hipótese alternativa H1 é considerada.

A Tabela 4 mostra a comparação, entre GRASP-GENIUS-VND proposto e as versões baseadas em ILS, usando-se o pacote R (R Project, 2014), 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 $p\text{-}valor$ foi menor que 0.05, indicando que a probabilidade da diferença de desempenho dos algoritmos ser devido à aleatoriedade é menor que 5%.

Tabela 4. 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 (Silva, 2009) foi superior ao GRASP-GENIUS-VND, 5 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.

3.4. Análise do Comportamento das Estratégias

A fim de comprovar a eficiência da abordagem proposta neste trabalho, comparando-o ao desempenho dos algoritmos ILS-MRD_AD e ILS-MRD_IB, foram realizados experimentos para verificar a distribuição de probabilidade empírica. Utilizou-se o método descrito em (Aiex *et al.*, 2006), 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 $T = \{t_1, t_2, ..., t_n\}$

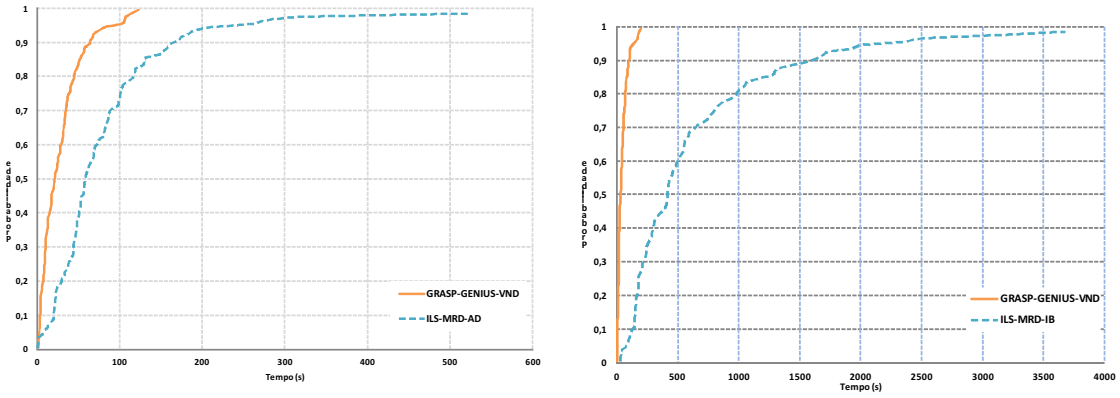

Figura 1. Distribuição da probabilidade acumulada: (a) instância *gr96_VT18_T58_W20_25*, do Grupo 1, com alvo 42746 e (b) *pr264_VT52_T53_W159_25*, do Grupo 2, com alvo 27693.

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 1 apresenta os gráficos da distribuição da probabilidade para as instâncias *gr96_VT18_T58_W20_25* (Grupo 1) e *pr264_VT52_T53_W159_25* (Grupo 2), após 100 execuções de GRASP-GENIUS-VND e ILS-MRD_AD, utilizando-se como valor alvo 42746, e de GRASP-GENIUS-VND e ILS-MRD_IB, usando-se como valor alvo 27693, respectivamente. 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.

Pode-se observar que, para o primeiro gráfico, 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. Já para o segundo gráfico, 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.

4. Conclusões e Trabalhos Futuros

Neste trabalho foi realizado um estudo sobre o problema de recobrimento de rotas com coletas de prêmios (PRRCP). Devido o PRRCP ser da classe $\mathcal{NP}$-difícil, foi proposto um algoritmo heurístico eficiente denominado GRASP-GENIUS-VND para sua resolução aproximada, baseado em GRASP e hibridizado com a heurísica GENIUS, como método de solução inicial, e com o método VND, como mecanismo de busca local.

A validação do algoritmo proposto deu-se por meio de experimentos computacionais realizados sobre dois grupos de instâncias da literatura, cujos os resultados foram confrontados com os algoritmos ILS-MRD_AD e ILS-MRD_IB, ambos propostos em

(Silva, 2009). Para o Grupo 1, contendo 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*, 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.

Como trabalhos futuros pretende-se implantar algum mecanismo de memória de longo prazo, por meio de mineração de dados e um mecanisno dinâminco de escolha do valor do α, conhecido como GRASP reativo (Prais e Ribeiro, 2000).

Referências

Aiex, R. M., Resende, M. G. C., e Ribeiro, C. C. (2006). TTTPlots: A Perl Program to Create Time-to-Target Plots. *Optimization Letters*, 1:10 – 1007.

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

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

Glover, F. (1996). Tabu search and adaptive memory programing - Advances, applications and challenges. Em *Interfaces in Computer Science and Operations Research*, páginas 1–75. Kluwer.

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

Lourenço, H. R., Martin, O. C., e Stutzle, T. (2003). Iterated Local Search. Em Glover, F. e Kochenberger, G., editores, *Handbook of Metaheuristics*, páginas 321–353. Kluwer Academic Publishers, Norwell, MA.

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

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

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

R Project (2014). The R Project for Statistical Computing. `http://www.r-project.org/`, última visita em 31/03/2014.

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

Resende, M. G. C. e Ribeiro, C. C. (2003). Greedy Randomized Adaptive Search Procedures. Em Glover, F. e Kochenberger, G., editores, *Handbook of Metaheuristics*, páginas 219–250. Kluwer Academic Publishers, Norwell, MA.

Siegel, S. e Castellan Jr, N. J. (1988). *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, 2^a edição.

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

Silva, R. (2014). Metaheurística aplicada ao Problema de Recobrimento de Rotas com Coleta de Prêmios. *Dissertação de Mestrado, Universidade Federal Fluminense*.