UNIVERSIDADE FEDERAL DE VIÇOSA
CENTRO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Caixeiro Viajante com Passageiros e Lotação (PVC-PL)

Matheus Aguilar de Oliveira - 95653
Natan Seabra Garcias - 95683

Trabalho apresentado como parte do processo de avaliação da disciplina INF - 492, ministrada pelo professor André Gustavo.

Viçosa, Minas Gerais
Dezembro, 2019

Introdução

O Caixeiro Viajante com Passageiros e Lotação (PVC-PL) é um problema combinatório desafiador que trata-se da otimização do gasto realizado em uma viagem à diversas cidades realizados por um caixeiro, no problema abordado, o caixeiro é um motorista, sendo que em cada cidade existe um passageiro com o qual possa compartilhar os gastos da viagem. Além disso, o caixeiro pode aproveitar dos descontos obtidos em faixas de trânsito que isentam veículos lotados do pagamento de pedágio. Este problema foi proposto por Ranmsés E. M. Bastos, Marco C. Goldbarg e Elizabeth F. G. Goldbarg [1], e é baseado no PVC - Problema do Caixeiro Viajante.

Visto que o problema demanda grande esforço computacional e tempo quando resolvido de maneira exata, apresentamos aqui dois algoritmos baseados nas meta-heurísticas *Greedy Randomized Adaptive Search Procedure* (GRASP) e *Biased Random-Key Genetic Algorithm* (BRKGA), respectivamente. Os resultados e tempos dos algoritmos são comparados na tentativa de avaliarmos a eficiência das nossas heurísticas e avaliar o desempenho de uma meta-heurística populacional contra uma meta-heurística single-solution.

Dados do Problema

O problema em questão foi modelado de acordo com os seguintes modelos matemáticos propostos por Bastos, Goldbarg e Goldbarg [1].

Representação:

O problema é representado por um grafo completo valorado, no qual cada vértice é uma cidade e o peso de uma aresta corresponde ao custo de viajar entre as cidades que incidem sobre ela. Ademais, cada cidade contém um passageiro que deseja viajar para alguma outra do conjunto. É importante citar também que algumas das arestas equivalem às *high-occupancy vehicle lanes*, que correspondem a pedágios que reduzem o peso caso uma certa lotação seja atingida, diminuindo assim uma parte do custo da viagem caso essa condição ocorra.

Função Objetivo:

O objetivo do problema é minimizar a seguinte função, que é um somatório de valores relacionados a cada arco do grafo:

$$\sum_{i,j \in N} \frac{d_{ij} f_{ij} - c_{ij} w_{ij}}{\sum_{l \in L} v_{ij}^{l} + 1}$$

Figura 1: Função Objetivo do Problema.

Onde as variáveis seguem a seguinte correspondência:

d_{ij} = custo de viajar da cidade i para a cidade j.

f_{ij} = 1, se o arco ij é usado e 0, caso o contrário.

c_{ij} = valor da redução de custo caso a aresta seja uma *high-occupancy vehicle lane.*

w_{ij} = 1, se a lotação do carro é maior ou igual à lotação mínima para isenção de imposto no arco e 0, caso o contrário.

v^{l}_{ij} = 1, se o passageiro l passou pelo arco ij e 0, caso o contrário.

Características e Restrições do Problema:

- O motorista sempre deve sair da cidade 0.
- O motorista nunca deve passar duas vezes na mesma cidade, exceto pela 0.
- O motorista deve voltar para a cidade 0 ao final da viagem.
- Nenhum passageiro quer ir para a cidade 0 ou para sua própria cidade.
- O carro possui uma capacidade máxima de passageiros (contando o motorista) que nunca deve ser excedida.
- Um passageiro só estará disposto a viajar se o custo total de sua viagem não exceder o máximo que ele está disposto a pagar.
- O custo de um arco específico é dividido igualmente para o motorista e os passageiros presentes no arco.
- A lotação mínima exigida para a isenção em qualquer um dos arcos é sempre igual à lotação máxima do carro.

Dados das heurísticas

Foram utilizadas as meta-heurísticas *Greedy Randomized Adaptive Search Procedure* (GRASP) e *Biased Random-Key Genetic Algorithm* (BRKGA) no problema em questão, bem como os seguintes algoritmos para auxiliar o funcionamento dessas:

Heurística de Carregamento:

O PCV-PL exige um controle tanto da parte da ordem das cidades da rota como de quais passageiros serão alocados no veículo, o que se mostrou problemático de se fazer em conjunto. Por isso, seguindo o proposto por Bastos, Goldbarg e Goldbarg [1], tal tratamento foi feito de forma separada, com as meta-heurísticas cuidando da rota e uma heurística de carregamento cuidando da alocação.

A heurística em questão funciona de forma gulosa e determinística, alocando todo passageiro que pode ser alocado de início, sem considerar o limite de pagamento de cada. Caso a solução ao final atenda aos limites, essa será a utilizada, caso o contrário, a heurística tenta gerar outra solução, mas removendo o passageiro que mais estourou o seu limite do conjunto de passageiros válidos para serem alocados.

Pseudo-código:

1	$\theta \leftarrow FALSE$
2	**Desmarque** todos os passageiros; **Para** $i \leftarrow 1$ até n **faça** $L[i] \leftarrow 0$
3	**Enquanto** $\theta = FALSE$
4	**Para** $i \leftarrow 1$ até n
5	**Se** o embarque do passageiro p no vértice $s[i]$ é viável **faça**
6	**Se** p é desmarcado e existem assentos disponíveis **faça** $L[s[i]] \leftarrow 1$
7	$s.custo \leftarrow calcula\_custo(s)$
8	$maxtar \leftarrow 0; j \leftarrow 0$
9	**Para** $i \leftarrow 1$ até n
10	**Se** $L[i] = 1$ **faça** $tarifa[i] \leftarrow calcula\_tarifa(i)$
11	**Se** $maxtar < tarifa[i] - Tarmax[i]$ **faça**
12	$maxtar \leftarrow tarifa[i] - Tarmax[i]; j \leftarrow i$
13	**Se** $j \neq 0$ **faça**
14	**Marque** o passageiro j
15	**Para** $i \leftarrow 1$ até n **faça** $L[i] \leftarrow 0$
16	**Caso contrário,** $\theta \leftarrow TRUE$

Figura 2: Pseudocódigo da heurística de carregamento.

GRASP:

O algoritmo proposto segue o modelo-padrão da GRASP, com as seguintes características:

Parâmetros:

- T : Tamanho da Lista Candidata.
- N : Número de Iterações.

Representação de uma solução: uma permutação que contém a ordem das cidades a serem visitadas.

Critério de parada da GRASP: quando o número de iterações for atingido.

Tipo da Busca Local: *best-improvement.*

Vizinhança da Busca Local: troca vértices adjacentes.

Critério de Parada da Busca Local: quando não encontra uma solução melhor.

Pseudocódigo:

Algorithm 1: GRASP

```
Input  : Número de Iterações, Tamanho da Lista Candidata
N ← Número de Iterações
S ← Solução Vazia
J ← 0
T ← Tamanho da Lista Candidata
while J < N do
    S' ← Solução Inicial começando do vértice 0
    C ← Cidades que ainda não estão na rota
    while C ≠ ∅ do
        V ← Vetor com a distância para cada cidade do conjunto C
        for i in C do
            V[i] ← Distância da última cidade da rota até a cidade i
        end
        As T cidades com a menor distância farão parte da lista
        S' ← Escolho uma cidade α aleatoriamente da lista
        Removo a cidade α do conjunto C
    end
    S' ← Busca Local(S)
    if f(S') > f(S) then
        S ← S'
    J ← J + 1
end
Output: S'
```

Figura 3: Pseudocódigo da GRASP aplicada ao problema.

Algorithm 2: Busca Local

```
Input  : Solução S
S' ← S
while Melhorando do
    while i in S-1 do
        swap S[i],swap S[i+1]
        if f(S) < f(S') then
            S' ← S
    end
end
S ← S'
Output: S
```

Figura 4: Pseudocódigo da Busca Local aplicada.

BRKGA:

O algoritmo proposto segue o modelo-padrão do BRKGA, com as seguintes características:

Parâmetros:

- $|P|$: tamanho da população.
- $|P_e|$: tamanho da população elite.
- $|P_m|$: tamanho da população mutante/imigrante.
- N: tamanho dos vetores de chaves (igual ao número de cidades da instância).
- α: porcentagem de escolher um gene elite (no algoritmo utilizado este valor sempre é maior do que 50%).
- K: número de iterações seguidas sem melhora para o algoritmo parar.

Representação de um indivíduo da população: um vetor com $N - 1$ chaves aleatórias entre 0 e 1. Desse modo, a posição i do vetor corresponde à cidade $i + 1$.

Decodificação da solução: as chaves do vetor são ordenadas e a ordem que essas chaves assumiram passa a ser a ordem das cidades na rota, visto que cada chave corresponde a uma cidade (exceto a zero).

População Inicial: gerada de forma aleatória.

Geração: em cada geração os $|P_e|$ melhores indivíduos são escolhidos para compor o conjunto elite, enquanto os demais ficam no conjunto não elite.

Crossover: o crossover é sempre feito entre um membro elite e um membro não-elite, sendo que os genes de pai elite tem uma chance α de sobrepor os genes do não-elite, de modo a manter a qualidade da solução.

Mutação: no BRKGA não há operador de mutação e sim um acréscimo de novos $|P_m|$ indivíduos aleatórios (mutantes ou imigrantes), de modo a manter a diversidade das soluções.

Nova geração: a nova geração é formada pelo conjunto elite, o conjunto de mutantes e os filhos formados pelo crossover.

Critério de Parada: o algoritmo para após K iterações seguidas sem que haja melhora da população.

Pseudocódigo:

Algorithm 3: BRKGA

Input : $|P|$, $|P_e|$, $|P_m|$, N, α, K
Inicialize os valores da melhor solução encontrada com infinito
$F \leftarrow \infty$
while $i < K$ **do**
  Gere uma população inicial P aleatoriamente com n genes
  Avalie o valor de cada indivíduo da população
  Particione P em dois conjuntos: P_e e P_n
  A próxima geração recebe as melhores da geração atual
  $P_+ \leftarrow P_e$
  Gere o conjunto P_m de imigrantes aleatoriamente
  Adicione P_m a próxima geração
  $P_+ \leftarrow \mathrm{P}_+ \cup P_m$
  for $i \leftarrow$ *to* $|P|$ - $|P_e|$ - $|P_m|$ **do**
    Selecione um indivíduo a aleatoriamente do conjunto $|P_e|$
    Selecione um indivíduo b aleatoriamente do conjunto $|P_n|$
    for j *in* N **do**
      $Q \leftarrow \beta \in [0, 100]$
      if $Q < \beta$ **then**
        Escolha um gene do pai elite
      else
        Escolha um gene do pai normal
      end
    end
    Adicione o filho C a próxima geração
    $P_+ \leftarrow \mathrm{P}_+ \cup \mathrm{C}$
  end
  Atualize a população
  $P \leftarrow P_+$
  Encontre a melhor solução $X_+ \in P$
  $X_+ \leftarrow$ argmin [f(X) | X in P]
  if $f(X_+) < F$ **then**
    $i \leftarrow 0$
    $X_* \leftarrow X_+$
    $F \leftarrow \mathrm{f}(X_+)$
end
Output: X_*

Figura 5: Pseudocódigo do BRKGA aplicado.

Metodologia

Neste trabalho, os algoritmos foram implementados utilizando a linguagem C++11 e compilados pelo compilador utilizando a versão 9.2.120191008 do GNU Compiler Collection (GCC) em conjunto com a diretiva de compilação para otimização de código -O3. A execução das meta-heurísticas se deu em máquinas com um processador Intel© Core™ i5-5200U CPU @ 2.20GHz × 2 e com sistema operacional Linux Mint 19.1 Cinnamon.

De modo a avaliar o desempenho das heurísticas, foi utilizado o mesmo conjunto de instâncias de Bastos, Goldbarg e Goldbarg [1], contendo 300 casos de teste divididos em 12 classes com 25 instâncias. Cada uma dessas foi executada utilizando uma combinação de parâmetros diferente, cujos quais foram calibrados utilizando a ferramenta Minitab.

As instâncias estão disponíveis em http://www.dimap.ufrn.br/lae/projetos/CaRS.php

Calibragem

Com auxílio da ferramenta Minitab foram feitos os testes de Anova e de Tukey, de modo a detectar se algum conjunto de parâmetros era significativamente diferente dos demais, bem como avaliar quais eram os agrupamentos. Para isso, foram utilizados os resultados normalizados de cada configuração aplicada nas 300 instâncias disponíveis.

GRASP:

No caso da meta-heurística GRASP, haviam apenas dois parâmetros a serem considerados: o tamanho da lista candidata e o número de iterações. Com bases em testes empíricos, foram selecionados os seguintes conjuntos de valores para serem avaliados no Minitab:

Tamanho da Lista Candidata:

- 3 (fixo)
- 5 (fixo)
- $\log_2(n)$, sendo n o número de cidades da instância

Os tamanhos escolhidos para a lista foram relativamente pequenos, visto que a maioria das instâncias não possuía muitas cidades.

O tamanho da lista candidata referente a $\log_2(n)$ foi escolhido para verificar a qualidade dos resultados referente a uma lista candidata que se adapta ao tamanho das instâncias.

Iterações:

- 10 * n, sendo n o número de cidades da instância
- 100 * n, sendo n o número de cidades da instância

Resultados do Minitab para a GRASP:

Note que para os dados apresentados abaixo nV representa a quantidade de cidades de cada instância.

ANOVA

Análise de Variância

Fonte	GL	SQ (Aj.)	QM (Aj.)	Valor F	Valor-P
Fator	5	4,668	0,933591	267,76	0,000
Erro	1794	6,255	0,003487		
Total	1799	10,923			

Figuras 6: Análise do Valor-P.

Boxplot

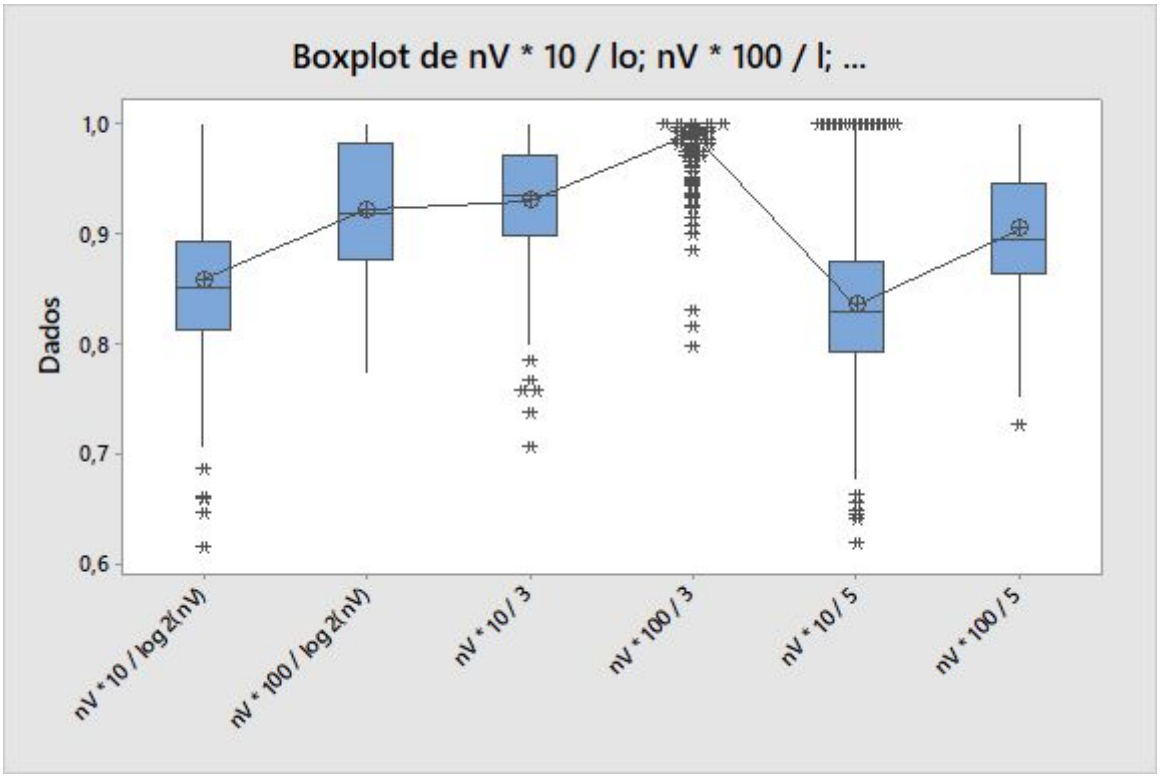

Figura 7: Análise do gráfico Bloxpot gerado.

Agrupamento de Tukey

Informações de Agrupamento Usando Método de Tukey e Confiança de 95%

Fator	N	Média	Agrupamento
nV * 100 / 3	300	0,99203	A
nV * 10 / 3	300	0,92944	B
nV * 100 / log2(nV)	300	0,92179	B
nV * 100 / 5	300	0,90440	C
nV * 10 / log2(nV)	300	0,85755	D
nV * 10 / 5	300	0,83519	E

Médias que não compartilham uma letra são significativamente diferentes.

Figura 8: Análise das médias e agrupamentos.

Como é possível observar nas imagens, o p-valor foi menor do que 0.05, logo há uma diferença significativa entre as médias. Além disso, pelo agrupamento, é possível notar que a combinação de 100 * n iterações com lista de tamanho 3 foi superior a todas as demais nesse conjunto de casos. Portanto, essa foi a configuração definitiva utilizada.

BRKGA:

No caso da meta-heurística BRKGA, haviam os seguintes parâmetros a serem considerados: o tamanho da população, o tamanho do conjunto elite, o tamanho do conjunto de mutantes/imigrantes, o número de iterações e a chance de herdar um gene elite. Como o número de parâmetros é consideravelmente grande, foram realizados testes empíricos para avaliar quais apresentavam maior significância na qualidade da solução. Por isso, os parâmetros foram divididos da seguinte forma:

Parâmetros que ficaram fixos:

Tamanho do conjunto elite: 20 % da população.
Chance de herdar um gene elite: 75 %.

Parâmetros que foram variados e calibrados pela análise do Minitab:

Tamanho da População:

- 1000.
- 5000.

Os tamanhos escolhidos para a lista foram relativamente grande, visto que a maioria das instâncias não possuía muitas cidades, o que permitia uma população maior sem afetar o desempenho.

Iterações seguidas sem melhora:

- 50.
- 100.

Tamanho do conjunto de mutantes/imigrantes:

- 10 % da nova população.
- 25 % da nova população.

ANOVA

Análise de Variância

Fonte	GL	SQ (Aj.)	QM (Aj.)	Valor F	Valor-P
Fator	7	1,915	0,273549	77,04	0,000
Erro	2392	8,494	0,003551		
Total	2399	10,408			

Figuras 9: Análise do Valor-P.

Boxplot

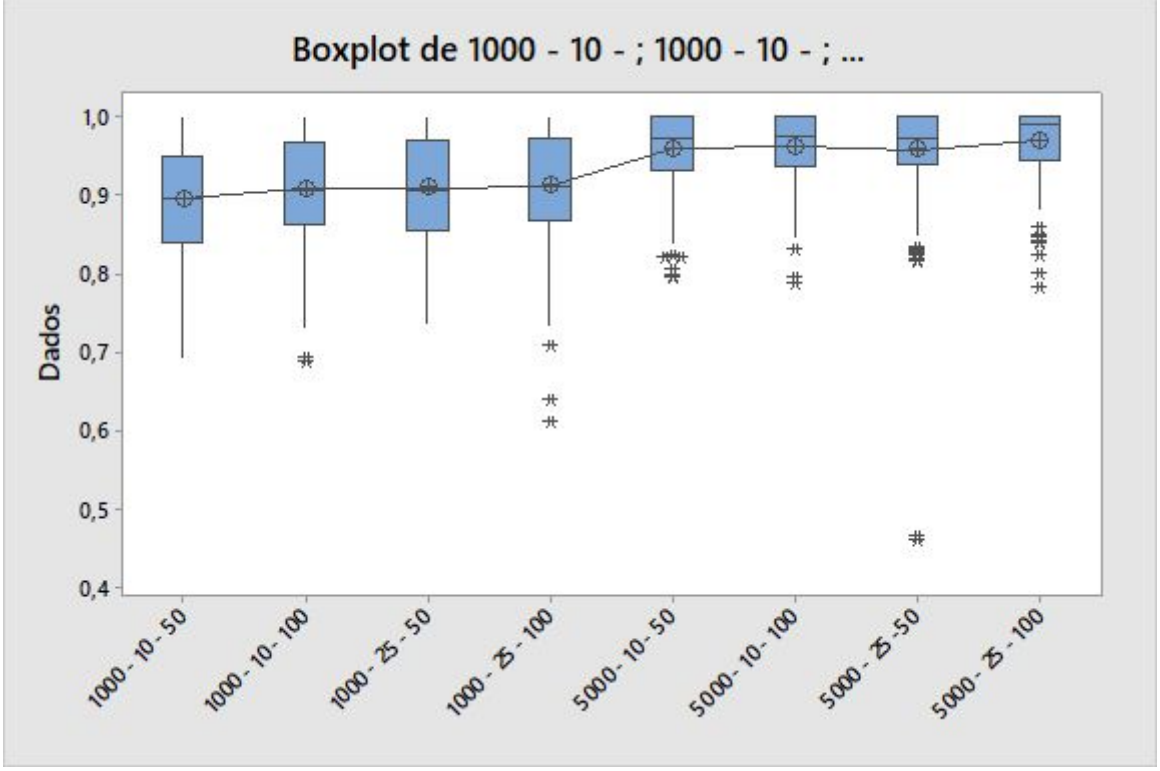

Figura 10: Análise do gráfico Bloxpot gerado.

Agrupamento de Tukey

Informações de Agrupamento Usando Método de Tukey e Confiança de 95%

Fator	N	Média	Agrupamento		
5000 - 25 - 100	300	0,96916	A		
5000 - 10 - 100	300	0,96129	A		
5000 - 10 - 50	300	0,95874	A		
5000 - 25 -50	300	0,95716	A		
1000 - 25 - 100	300	0,91146		B	
1000 - 25 - 50	300	0,90917		B	C
1000 - 10 - 100	300	0,90825		B	C
1000 - 10 - 50	300	0,89571			C

Médias que não compartilham uma letra são significativamente diferentes.

Figura 11: Análise das médias e agrupamentos.

Novamente, o teste ANOVA apresentou um p-valor menor do 0.05, logo há diferença significativa entre as médias para essas instâncias. É possível notar também que o fator mais significativo foi o tamanho da população, visto que as configurações com 5000 sobressaíram as com 1000. Ademais é possível notar que um número de mutantes/imigrantes igual a 25% é um pouco melhor do que 10% e que as opções com população de tamanho 5000 são todas equivalentes, por isso optou-se por manter aquela que possui a maior média.

Resultados

Abaixo, foi feito um processo de comparação das duas meta-heurísticas propostas.

Análise Estatística:

Novamente utilizando Minitab, aplicou-se os testes Anova e Tukey nos resultados normalizados da GRASP e do BRKGA em todas as 300 instâncias, para compará-los, obtendo os seguintes dados:

ANOVA

Análise de Variância

Fonte	GL	SQ (Aj.)	QM (Aj.)	Valor F	Valor-P
Fator	1	0,1274	0,127429	75,36	0,000
Erro	598	1,0112	0,001691		
Total	599	1,1386			

Figuras 12: Análise do Valor-P.

Boxplot

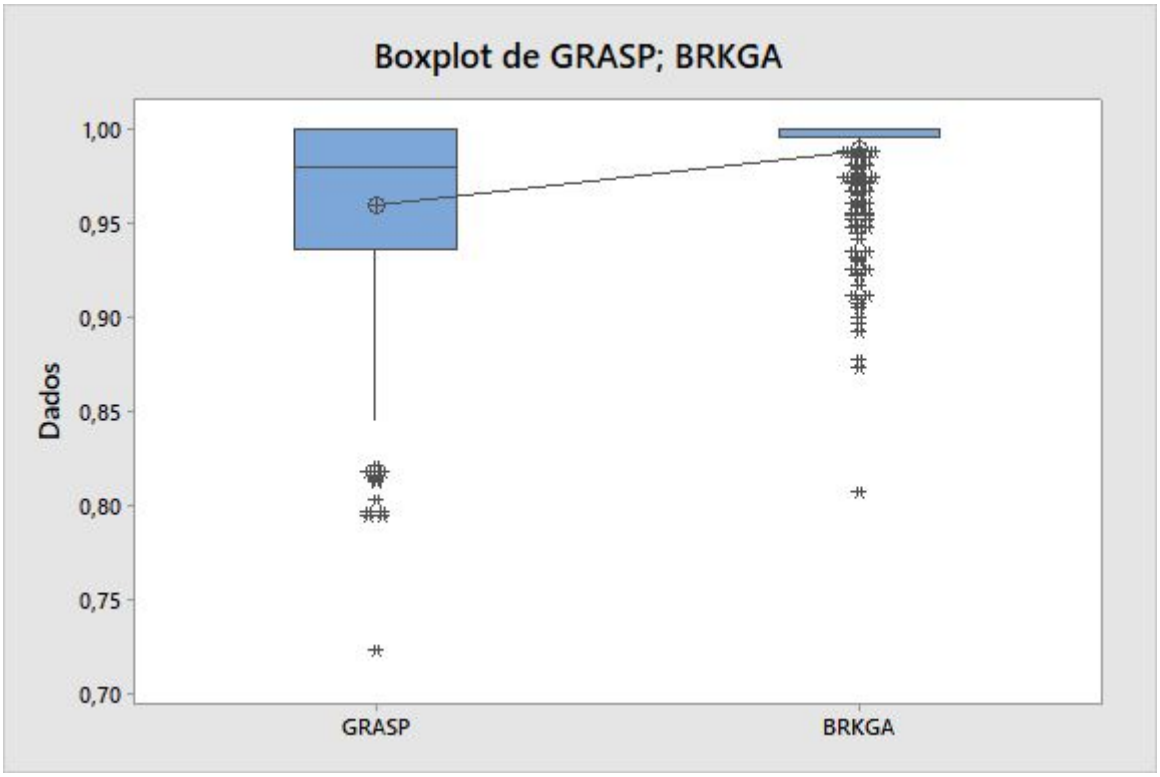

Figura 13: Análise do gráfico Bloxpot gerado.

Tukey

Informações de Agrupamento Usando Método de Tukey e Confiança de 95%

Fator	N	Média	Agrupamento	
BRKGA	300	0,98852	A	
GRASP	300	0,95937		B

Médias que não compartilham uma letra são significativamente diferentes.

Figura 14: Análise das médias e agrupamentos.

Resultados complementares:

Ambas as meta-heurísticas foram executadas para os 300 casos de teste fornecidos. Todavia, muitos desses possuíam instâncias bastante pequenas. Dessa maneira, optou-se por também realizar algumas análises nas instâncias maiores. A tabela a seguir apresenta os dados do resultados obtidos nas 60 maiores instâncias, sendo que todas elas contém 50 cidades:

GRASP	BRKGA	GRASP Normalizado	BRKGA Normalizado	Tempo GRASP(segundos)	Tempo BRKGA (segundos)
1541.67	1667.33	1.000	0.925	3.703	15.427
1640.00	1869.00	1.000	0.877	3.322	17.045
1631.33	1594.33	0.977	1.000	3.671	25.759
1608.17	1616.33	1.000	0.995	3.542	13.741
1603.00	1463.33	0.913	1.000	3.291	18.962
1675.67	1811.67	1.000	0.925	3.515	16.824
1830.17	1752.67	0.958	1.000	3.529	15.075
1564.50	1480.67	0.946	1.000	3.256	17.995
1574.67	1621.17	1.000	0.971	3.133	18.273
1633.33	1549.83	0.949	1.000	3.253	17.873
1193.67	1168.67	0.979	1.000	3.857	17.620
1270.33	1219.58	0.960	1.000	4.068	19.935
1175.00	1345.92	1.000	0.873	3.667	16.864
1159.75	1214.83	1.000	0.955	3.663	26.467
1334.17	1242.00	0.931	1.000	4.178	22.115
1355.50	1469.25	1.000	0.923	4.085	19.923
1363.00	1291.50	0.948	1.000	4.379	19.784
1280.50	1265.00	0.988	1.000	3.734	18.953
1315.83	1260.25	0.958	1.000	4.469	20.120
1182.17	1241.92	1.000	0.952	3.877	26.660
1491.67	1656.50	1.000	0.900	3.063	17.400
1637.67	1683.00	1.000	0.973	2.871	16.374
1635.00	1707.17	1.000	0.958	2.607	15.540
1648.00	1644.67	0.998	1.000	3.058	13.923
1699.17	1722.00	1.000	0.987	2.618	12.063
1655.67	1635.17	0.988	1.000	2.353	18.626
1524.83	1451.67	0.952	1.000	2.496	15.968
1579.33	1521.67	0.963	1.000	2.492	20.232
1602.83	1627.33	1.000	0.985	2.816	15.783
1515.50	1588.67	1.000	0.954	2.533	12.908
1303.67	1338.75	1.000	0.974	2.910	15.651
1217.17	1253.83	1.000	0.971	2.850	12.567
1310.75	1258.25	0.960	1.000	3.111	22.927
1239.92	1189.25	0.959	1.000	3.240	12.862

1238.75	1208.17	0.975	1.000	3.031	21.554
1174.58	1164.67	0.992	1.000	2.798	28.921
1188.17	1238.67	1.000	0.959	2.569	17.316
1220.67	1270.67	1.000	0.961	2.740	16.167
1138.83	1234.58	1.000	0.922	2.672	15.408
1235.67	1273.33	1.000	0.970	2.760	16.852
1515.83	1610.33	1.000	0.941	1.843	10.176
1455.67	1473.67	1.000	0.988	1.688	11.465
1575.83	1600.00	1.000	0.985	1.437	14.639
1590.33	1512.00	0.951	1.000	1.784	13.589
1528.00	1524.00	0.997	1.000	1.799	15.570
1649.17	1697.00	1.000	0.972	1.593	11.994
1513.17	1594.17	1.000	0.949	1.667	12.586
1434.17	1513.50	1.000	0.948	1.619	10.524
1548.50	1582.67	1.000	0.978	1.620	13.594
1580.00	1557.50	0.986	1.000	1.703	12.659
1230.33	1225.42	0.996	1.000	1.648	12.690
1248.83	1172.58	0.939	1.000	1.681	18.812
1211.83	1304.33	1.000	0.929	1.768	11.366
1282.08	1285.33	1.000	0.997	1.791	18.677
1181.42	1132.17	0.958	1.000	1.757	13.346
1212.92	1216.00	1.000	0.997	1.739	14.325
1280.17	1245.67	0.973	1.000	1.845	15.486
1260.67	1238.25	0.982	1.000	1.994	17.168
1339.25	1219.33	0.910	1.000	1.937	11.185
1241.75	1176.33	0.947	1.000	1.707	13.222

Figura 15: Tabela de comparação entre os resultados obtidos.

Foram também obtidos os seguintes dados:

Porcentagem de casos com vitória da GRASP:

- Nos 60 maiores: 51 %
- No conjunto inteiro: 26 %

Média dos valores normalizados da GRASP:

- Nos 60 maiores: 0.982
- No conjunto inteiro: 0.959

Porcentagem de casos com vitória do BRKGA:

- Nos 60 maiores: 48 %
- No conjunto inteiro: 61 %

Média dos valores normalizados do BRKGA:

- Nos 60 maiores: 0.989
- No conjunto inteiro: 0.977

Análise do Resultados

Novamente, o p-valor foi menor do que 0.05, o que indica diferença significativa entre médias a favor do BRKGA sobre a GRASP, o que condiz com as médias apresentadas ao se analisar todo o conjunto de dados. Porém, não é possível afirmar que esse comportamento vale para todo tipo de instância, visto que a GRASP possui melhor média e maior número de vitórias quando se trata apenas das 60 maiores instâncias. Isso indica uma vantagem do BRKGA apenas nos casos menores.

Ademais, outro ponto a se considerar é o fato do BRKGA ser cerca de 10 vezes mais lento do que a GRASP, o que pode ser problemático para casos grandes. Com isso, é possível inferir que o BRKGA é mais adequado do que a GRASP em situações nas quais a eficiência de tempo é menos relevante.

Conclusão

Este trabalho apresentou a aplicação das meta-heurísticas GRASP e BRKGA para o problema PVC-PL, o qual, embora tenha certa semelhança com o PCV, contém características que atribuem ao modelo uma maior complexidade.

Apesar disso, levando em consideração os aspectos apresentados acima, as meta-heurísticas se mostraram eficientes para a resolução de tal problema, sendo válido destacar também que a aplicação do BRKGA ao problema obteve resultados significativos, demonstrando assim que meta-heurísticas populacionais, quando bem calibradas, são tão boas quanto as meta-heurísticas single-solution.

Para trabalhos futuros, é considerado a aplicação de uma nova meta-heurística populacional como a Ant Colony Optimization, visto que sua criação se deu com o intuito de resolver o PCV e a aplicação de uma meta-heurística para escolher os passageiros que viajarão junto ao motorista, visto que isso pode convergir em melhores resultados.

Referências

[1] BASTOS, Ranmsés Emanuel Martins. O problema do caixeiro viajante com passageiros e lotação. 2017. 139f. Dissertação (Mestrado em Sistemas e Computação) - Centro de Ciências Exatas e da Terra, Universidade Federal do Rio Grande do Norte, Natal, 2017.

[2] RUIZ, Efrain et al. A biased random-key genetic algorithm for the capacitated minimum spanning tree problem. **Computers & Operations Research**, v. 57, p. 95-108, 2015.

[3] GONÇALVES, José Fernando; RESENDE, Mauricio GC. Biased random-key genetic algorithms for combinatorial optimization. **Journal of Heuristics**, v. 17, n. 5, p. 487-525, 2011.