REVISTA DO SEMINÁRIO INTERNACIONAL DE ESTATÍSTICA COM R | **ISSN: 2526-7299**
VOL 5, Nº 1, ABRIL DE 2024

Modelagem de Problema de Roteamento de Veículos com o pacote RouteR

Luciane Ferreira Alcoforado-Academia da Força Aérea – lucianea@id.uff.br

Resumo.

Este trabalho apresenta o pacote RouteR como auxiliar no processo de modelagem de Problemas de Roteamento de Veículos até a obtenção da solução ótima utilizando o algoritmo de Programação Inteira.
Palavras-chave: VPR, Vehicle Routing Problem, RouteR

Abstract

This work presents the RouteR package as an aid in the process of modeling Vehicle Routing Problems until obtaining the optimal solution using the Integer Programming algorithm.

Keywords: . VPR, Vehicle Routing Problem, RouteR

Introdução

O problema de roteamento de veículos (VRP - Vehicle Routing Problem) é um desafio de otimização logística que envolve determinar as rotas mais eficientes para um conjunto de veículos que devem atender a uma série de locais de destino, chamados de clientes, levando em consideração diversas restrições e objetivos. Tradicionalmente, o objetivo é minimizar os custos totais, como o tempo de viagem, a distância percorrida, o custo de combustível, ou maximizar a eficiência da operação.

As principais características do VRP incluem:

1. **Depósitos e clientes:** Existem um ou mais depósitos de onde os veículos partem e retornam após a conclusão das rotas, além de clientes que devem ser atendidos.
2. **Capacidade dos veículos:** Cada veículo possui uma capacidade limitada para transportar mercadorias, o que impõe uma restrição à quantidade de itens que podem ser entregues em cada rota.
3. **Distâncias e tempos de viagem:** As distâncias entre os locais e os tempos de viagem podem variar, impactando diretamente nos custos e na eficiência das rotas.
4. **Restrições operacionais:** Podem incluir restrições de tempo, como janelas de tempo em que os clientes podem ser atendidos, restrições de precedência (alguns clientes devem ser visitados antes de outros), entre outras.

De acordo com Toth & Vigo (2002), existem várias variantes (famílias) do VRP, cada uma com suas próprias características e complexidades, como o CVRP (*Capacitated Vehicle Routing Problem*) com restrições de capacidade de carga, o VRP com janelas de tempo (VRPTW – *VRP with time windows*), o VRP com capacidade de carga e tempo (CVRPTW), o VRP com frota heterogênea (HFVRP - *Heterogeneous Fleet VRP*), entre outros. O objetivo principal em todos os casos é encontrar um conjunto de rotas ótimas que atendam a todas as restrições e objetivos definidos.

> O sucesso da utilização de técnicas de otimização se deve não apenas à potência dos atuais sistemas informáticos e à plena integração dos sistemas de informação nas operações e processos comerciais, mas também pode ser atribuído ao desenvolvimento de modelos matemáticos rigorosos, capazes de levar em conta quase todas as características do VRP decorrentes de aplicações do mundo real. Além disso, os algoritmos correspondentes e suas implementações de computador (ferramentas de software) desempenham um papel essencial na busca de soluções viáveis de alta qualidade para instâncias do mundo real dentro tempos de computação aceitáveis. Em comparação com procedimentos não baseados em técnicas de otimização, podem ser alcançadas economias significativas de custos e uma melhor utilização da frota de veículos. (Toth & Vigo, 2002).

A experiência desta autora mostra que não basta ter conhecimento de um vasto conjunto de algoritmos que solucionam os problemas, mas sim é preciso dispor de meios práticos para que o usuário comum possa se apropriar de tais ferramentas e obter a solução desejada, promovendo a saúde financeira de instituições tanto públicas como privadas.

Ao longo dos anos, diversos softwares têm sido desenvolvidos para abordar os desafios relacionados ao roteamento eficiente de veículos em diversas aplicações, como logística, distribuição e transporte. Entre essas ferramentas, destacam-se soluções comerciais como o OR-Tools, da Google (Furnon & Perron, 2024), e o CPLEX Optimization Studio, da IBM (IBM,2024), que oferecem recursos avançados para modelagem e resolução de problemas de roteamento. Além disso, existem também soluções de código aberto, como o OpenVRP desenvolvida na linguagem LISP (Kuo, 2012) e o VROOM desenvolvido na linguagem C++20 (Coupey & Nicod & Varnier, 2024), que proporcionam opções acessíveis e flexíveis para lidar com diferentes tipos de problemas de roteamento de veículos. Nesse cenário, o pacote RouteR surge como uma alternativa promissora, fornecendo uma abordagem específica para modelagem e resolução de problemas de roteamento de veículos no ambiente R, ampliando

assim as opções disponíveis para profissionais e pesquisadores interessados nessa área. Particularmente no caso em questão quanto ao roteamento de veículos, a primeira dificuldade do usuário é informar ao software os dados necessários para o emprego do(s) algoritmo(s) adequado(s). Diante desta situação, surge a pergunta que este artigo pretende responder: **"Como auxiliar o usuário a desenvolver de maneira prática os dados necessário para a aplicação do algoritmo de Programação Inteira (IPA) no ambiente computacional R?"**

Objetivos

Objetivo Geral: Desenvolver e implementar estratégias práticas e eficazes para auxiliar os usuários na preparação e organização dos dados necessários para a aplicação do algoritmo de Programação Inteira (IPA) no ambiente computacional R.

Objetivos Específicos: Desenvolver e implementar um conjunto de funções e ferramentas no R que facilitem a entrada, organização e formatação dos dados necessários para o IPA; Testar e validar as soluções propostas em cenários reais, utilizando conjuntos de dados representativos e aplicando o algoritmo de Programação Inteira para resolver problemas de roteamento de veículos específicos; Fornecer documentação detalhada e orientações claras sobre o uso das ferramentas desenvolvidas, garantindo que os usuários possam aplicá-las em seus próprios projetos e desafios logísticos.

Material e Método

O processo metodológico é delineado em etapas distintas: 1- Realizamos uma revisão abrangente da literatura sobre modelagem de problemas de roteamento de veículos, algoritmos de Programação Inteira e técnicas de preparação de dados em R; 2- Conduzimos entrevistas com usuários e especialistas para identificar as principais dificuldades enfrentadas ao preparar os dados para o IPA no ambiente R; 3- Pesquisamos e analisamos as melhores práticas e técnicas existentes para simplificar e automatizar o processo de preparação de dados para o IPA no R; 4- Desenvolvemos um conjunto de funções e ferramentas no R para facilitar a entrada, organização e formatação dos dados necessários para o IPA; 5- Utilizamos conjuntos de dados representativos e aplicamos as funções do pacote, avaliando sua eficácia e eficiência em termos de tempo de execução, precisão e facilidade de uso.

Formulação

A notação básica adotada neste trabalho é descrita a seguir:

Seja G = (V, A) um grafo direcionado em que V representa o conjunto de vértices (clientes) e A representa o conjunto de arcos (trecho de ligação entre dois vértices), sendo V

= {1, 2, ..., n} e A = (V x V), veja um exemplo na Figura 1. Considere K o conjunto de veículos disponíveis, K = { 1, 2, ..., |K|}

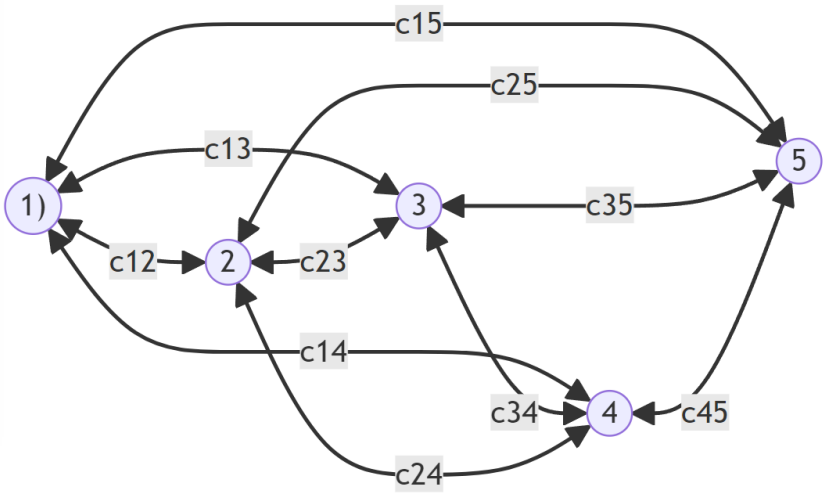

Figura 1 – Grafo completo com V = {1,2,3,4,5} e A = {(1,2), (1,3), …, (5,4)}, cij representa o custo do arco (i,j) em qualquer sentido.

Fonte: Autora, 2024.

A variável de decisão é definida por x_{ijk} = 1 se e somente se o veículo k se move pelo arco (i,j) ϵ A, sendo k o índice que representa o veículo, k = 1, 2, ..., |K|.

O modelo de otimização linear com o objetivo de minimizar o custo total da rota é dado por:

$$\text{Min } Z = \sum_{k=1}^{|K|} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ijk} x_{ijk}, \forall i \neq j \qquad \text{(Eq. 1)}$$

Sujeito a:

I. Família de restrições que garantem que cada veículo saia do ponto inicial 1.

$$\sum_{j=1}^{n} x_{1jk} = 1, \forall i \neq j, \text{k} = 1, 2, ..., |\text{K}| \qquad \text{(Eq. 2)}$$

II. Família de restrições que garantem que cada veículo retorne ao ponto inicial 1.

$$\sum_{i=1}^{n} x_{i1k} = 1, \forall i \neq j, \text{k} = 1, 2, ..., |\text{K}| \qquad \text{(Eq. 3)}$$

III. Família de restrições que garantem que cada cliente seja visitado uma única vez por algum dos veículos.

$$\sum_{k=1}^{|K|}\sum_{j=1}^{n} x_{ijk} = 1, \forall i \neq j, \text{i} = 1, 2, ..., \text{n} \qquad \text{(Eq. 4)}$$

IV. Família de restrições que garantem que nenhum cliente retenha o veículo que o visita.

$$\sum_{j=1}^{n} x_{ijk} - \sum_{j=1}^{n} x_{jik} = 0, \forall i \neq j, \text{i} = 1, 2, ..., \text{n} \text{ e k} = 1, 2, ..., |\text{K}| \qquad \text{(Eq. 5)}$$

V. Família de restrições que garantem a eliminação de sub-rotas de comprimento |S|-1, em que |S| representa o número de vértices na sub-rota, S contido em A.

$$\sum_{(ij)\in S} x_{ijk} = |S| - 1, \forall i \neq j, \text{k} = 1, 2, ..., |\text{K}| \qquad \text{(Eq. 6)}$$

VI. Define a variável de decisão como binária

$$x_{ijk} \in \{0,1\} \qquad \text{(Eq. 7)}$$

A modelagem apresentada nas equações Eq.1 a Eq.7 gera uma quantidade significativa de restrições, que cresce exponencialmente à medida que as restrições de impedimento de sub-rotas são adicionadas. Considerando um grafo completo em que todos os vértices se conectam entre si, o número de variáveis de decisão do problema será da ordem de $(n^2 – n)k$, assim, para um problema com n=5 e k=2 haverá precisamente (25 – 5)*2 = 40 variáveis binárias, com n=9 e k=5 serão 360 variáveis binárias. Cada família de restrição produzirá a seguinte quantidade de restrições: I. e II. serão |k| restrições, assim com n=9 vértices e k=5 veículos serão 5 restrições em cada categoria totalizando 10 restrições; III. serão n restrições, assim com n=9 vértices e k=5 veículos serão 9 restrições; IV. serão nk restrições, assim com n=9 e k=5 serão 45 restrições; V. serão $kC_{|S|}^{n}$ restrições, assim, com n=9 e k=5 e |S|=2, serão 5*(9!/(2!7!)) = 180 restrições, portanto um problema com n=9, k=5 e |S|=2 haverá um total de 360 variáveis e 244 restrições, tal modelo garante evitar sub-rotas de comprimento 1. A cada nova família de restrição de sub-rota o problema vai aumentando o número de restrições podendo alcançar o máximo de 2.359 restrições no total, evitando-se sub-rotas de comprimento 1, 2, 3, ..., 7.

A metodologia adotada neste trabalho considerou a observação do modelo apresentado, produzindo as funções para gerar cada família de restrições utilizando-se da linguagem computacional R e a lógica de programação.

Resultados e Discussão

O pacote RouteR

O pacote RouteR foi desenvolvido a partir das lacunas identificadas durante entrevistas realizadas com usuários e especialistas, juntamente com a análise das melhores práticas para a automação do processo de modelagem. Ao reconhecer os desafios inerentes à preparação de dados e modelagem do problema, tornou-se evidente a demanda por uma solução que simplificasse e otimizasse esse procedimento.

Cabe registrar que o nome "RouteR" é uma combinação entre a palavra "route" (rota, em inglês) e a letra "R". A escolha desse nome busca transmitir de forma direta e concisa o propósito do pacote: fornecer ferramentas e funcionalidades específicas para lidar com problemas de roteamento de veículos no ambiente R.

Funções do pacote RouteR

A seguir as funções desenvolvidas para o pacote, para um problema com **n** nós, sendo o nó 1 considerado o nó de origem e destino final, enquanto que os nós 2, …, n – 1 são os nós a serem visitados uma única vez. Todas as funções desenvolvidas no pacote RouteR (versão 0.1.0) baseiam-se nas equações apresentadas anteriormente Eq. 1 a Eq. 7 e são compostas basicamente de dois tipos: gerar_() e organizar_(), o grupo de funções gerar_restricoes_() possui o argumento *n_restricao* que possibilita a conveniência da numeração sequencial das restrições do modelo.

1- gerar_variaveis(i,j,k): gera os nomes das variáveis com base nos índices informados, por padrão k=1. O número de variáveis será igual a $(i^2 - i)k$, considerando a conexão entre todos os pares (i, j) com $i \neq j$.

```
RouteR::gerar_variaveis(5,5,3)
```

```
# A tibble: 60 × 4
       i     j     k x
   <int> <int> <int> <chr>
 1     1     2     1 x121
 2     1     2     2 x122
 3     1     2     3 x123
 4     1     3     1 x131
 5     1     3     2 x132
 6     1     3     3 x133
 7     1     4     1 x141
 8     1     4     2 x142
 9     1     4     3 x143
```

```
10     1     5     1 x151
# ℹ 50 more rows
```

2- gerar_custos(i, j, k, vetor_custo): gera um dataframe contendo os nomes das variáveis e seu respectivo custo. Caso o vetor_custo não seja informado, um vetor aleatório seja gerado. O vetor_custo deverá ter comprimento igual ao número de variáveis do modelo, ou seja, $(i^2 - i)k$.

```
custo=c(2,2,4,4,2,2,6,6,4,4,6,6)
RouteR::gerar_custo(3,3,2,custo)
```

```
     cod x121 x122 x131 x132 x211 x212 x231 x232 x311 x312 x321 x322 direcao b
custo 1    2    2    4    4    2    2    6    6    4    4    6    6        = 1
```

3- gerar_restricoes_saida_origem(i, j, k, `n_restricao`): gera a família de restrições da Eq. 2.

```
RouteR::gerar_restricoes_saida_origem(3,3,2, n_restricao=1)
```

```
    cod x121 x122 x131 x132 x211 x212 x231 x232 x311 x312 x321 x322 direcao b
R_1   1    1    0    1    0    0    0    0    0    0    0    0        = 1
R_2   2    0    1    0    1    0    0    0    0    0    0    0        = 1
```

4- gerar_restricoes_retorno_origem(i, j, k, `n_restricao`): gera a família de restrições da Eq. 3.

```
RouteR::gerar_restricoes_retorno_origem(3,3,2, n_restricao=3)
```

```
    cod x121 x122 x131 x132 x211 x212 x231 x232 x311 x312 x321 x322 direcao b
R_3   1    0    0    0    0    1    0    0    0    1    0    0        = 1
R_4   2    0    0    0    0    0    1    0    0    0    1    0        = 1
```

5- gerar_restricoes_unicidade(i, j, k, `n_restricao`): gera a família de restrições da Eq. 4.

```
RouteR::gerar_restricoes_unicidade(3,3,2, n_restricao=5)
```

```
    cod x121 x122 x131 x132 x211 x212 x231 x232 x311 x312 x321 x322 direcao b
R_5   1    0    0    0    0    1    1    1    1    0    0    0    0        = 1
R_6   2    0    0    0    0    0    0    0    0    1    1    1    1        = 1
```

6- gerar_restricoes_equilibrio(i, j, k, `n_restricao`): gera a família de restrições da Eq. 5.

```
RouteR::gerar_restricoes_equilibrio(3,3,2, n_restricao=7)
```

```
     cod x121 x122 x131 x132 x211 x212 x231 x232 x311 x312 x321 x322 direcao b
R_7    1   -1    0    0    0    1    0    1    0    0    0   -1    0        = 0
R_8    2    0    0   -1    0    0    0   -1    0    1    0    1    0        = 0
R_9    3    0   -1    0    0    0    1    0    1    0    0    0   -1        = 0
R_10   4    0    0    0   -1    0    0    0   -1    0    1    0    1        = 0
```

7- gerar_restricoes_subrota(i, j, k, `n_restricao, n`): gera a família de restrições da Eq. 6. Aqui deve-se atentar para o argumento n que significa o número de vértices na sub-rotas a ser considerada, este valor varia de 2 até i-2, desde que i>2, caso contrário essa família de restrições não terá sentido no modelo. Por exemplo no problema com 3 vértices uma solução possível seria a rota 1-2-1 no veículo 1 e 1-3-1 no veículo 2 e neste caso não teria sentido considerar sub-rotas. Além disso, essa família de restrições só deve ser considerada no caso em que a solução do modelo até a Eq. 5 apresente problemas de sub-rota. Nem sempre será necessário recorrer a esta família de restrições, sendo aplicável apenas em casos específicos onde a estrutura do problema o justifique.

```
RouteR::gerar_restricoes_subrota(4,4,1, n_restricao=1, n=2)
```

```
    cod x121 x131 x141 x211 x231 x241 x311 x321 x341 x411 x421 x431 direcao b
R_1   1    1    0    0    1    0    0    0    0    0    0    0      <= 1
R_2   2    0    1    0    0    0    0    1    0    0    0    0      <= 1
R_3   3    0    0    1    0    0    0    0    0    0    1    0      <= 1
R_4   4    0    0    0    0    1    0    0    1    0    0    0      <= 1
R_5   5    0    0    0    0    0    1    0    0    0    0    1      <= 1
R_6   6    0    0    0    0    0    0    0    0    1    0    0      <= 1
```

Após obter todos o conjunto de restrições, deve-se juntá-los em um único dataframe, utilizando-se do seguinte código:

```
n=5; k=1; vetor_custos=c(2.8,2,100,100,2.8,2.4,4.4,5.2,2,2.1,4.2,5,100,3.6,
4.1,0.65,0,0,0,0)
custo <- RouteR::gerar_custo(n,n,k,vetor_custos)
print(custo)
```

```
      cod x121 x131 x141 x151 x211 x231 x241 x251 x311 x321 x341 x351 x411 x421
custo   1  2.8    2  100  100  2.8  2.4  4.4  5.2    2  2.1  4.2    5  100  3.6
      x431 x451 x511 x521 x531 x541 direcao b
custo  4.1 0.65    0    0    0    0       = 1
```

```
n_restricao<-1
saida <- RouteR::gerar_restricoes_saida_origem(n,n,k,n_restricao)
saida
```

```
    cod x121 x131 x141 x151 x211 x231 x241 x251 x311 x321 x341 x351 x411 x421
R_1   1    1    1    1    1    0    0    0    0    0    0    0    0    0
    x431 x451 x511 x521 x531 x541 direcao b
R_1    0    0    0    0    0    0       = 1
```

```
n_restricao<- nrow(saida)+1
n_restricao
```

```
[1] 2
```

```
retorno <- RouteR::gerar_restricoes_retorno_origem(n,n,k,n_restricao)
retorno
```

```
    cod x121 x131 x141 x151 x211 x231 x241 x251 x311 x321 x341 x351 x411 x421
R_2   1    0    0    0    0    1    0    0    0    1    0    0    0    1    0
    x431 x451 x511 x521 x531 x541 direcao b
R_2    0    0    1    0    0    0       = 1
```

```
n_restricao<- n_restricao+nrow(retorno)
n_restricao
```

```
[1] 3
```

```
equilibrio <- RouteR::gerar_restricoes_equilibrio(n,n,k,n_restricao)
equilibrio
```

```
    cod x121 x131 x141 x151 x211 x231 x241 x251 x311 x321 x341 x351 x411 x421
R_3   1   -1    0    0    0    1    1    1    1    0   -1    0    0    0   -1
R_4   2    0   -1    0    0    0   -1    0    0    1    1    1    1    0    0
R_5   3    0    0   -1    0    0    0   -1    0    0    0   -1    0    1    1
R_6   4    0    0    0   -1    0    0    0   -1    0    0    0   -1    0    0
    x431 x451 x511 x521 x531 x541 direcao b
R_3    0    0    0   -1    0    0       = 0
R_4   -1    0    0    0   -1    0       = 0
R_5    1    1    0    0    0   -1       = 0
R_6    0   -1    1    1    1    1       = 0
```

```
n_restricao<- n_restricao+nrow(equilibrio)
n_restricao
```

```
[1] 7
```

```
unicidade <- RouteR::gerar_restricoes_unicidade(n,n,k,n_restricao)
unicidade
```

```
     cod x121 x131 x141 x151 x211 x231 x241 x251 x311 x321 x341 x351 x411 x421
R_7    1    0    0    0    0    1    1    1    1    0    0    0    0    0    0
R_8    2    0    0    0    0    0    0    0    0    1    1    1    1    0    0
R_9    3    0    0    0    0    0    0    0    0    0    0    0    0    1    1
R_10   4    0    0    0    0    0    0    0    0    0    0    0    0    0    0
     x431 x451 x511 x521 x531 x541 direcao b
R_7     0    0    0    0    0    0       = 1
R_8     0    0    0    0    0    0       = 1
R_9     1    1    0    0    0    0       = 1
R_10    0    0    1    1    1    1       = 1
```

```
Dados <- dplyr::bind_rows(custo,saida,retorno,equilibrio,unicidade)
```

Após organizar todos os elementos do modelo no dataframe *Dados*, podemos obter a solução final com a função gerar_rota() e posteriormente organizá-la para uma análise quanto a necessidade de acrescentar restrições de sub-rota.

```
RouteR::gerar_rota(Dados)
```

```
$rota
   var solucao
1 x131       1
2 x211       1
3 x321       1
4 x451       1
5 x541       1

$solucao
 [1] 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1

$objetivo
[1] 7.55
```

```
vetor_rota<-RouteR::gerar_rota(Dados)$rota$var
#Organizando a rota obtida
RouteR::organizar_rota(vetor_rota)
```

```
  Rota Veiculo
1   13       1
2   21       1
3   32       1
4   45       1
5   54       1
```

Ao analisarmos a solução gerada, observamos que há uma sub-rota 4-5-4 que deve ser eliminada. Assim, devemos acrescentar restrições da família V (Eq.6):

```
subrota1 <- RouteR::gerar_restricoes_subrota(n,n,k,n_restricao,2)

Dados <- dplyr::bind_rows(custo,saida,retorno,equilibrio,unicidade, subrota
1)
RouteR::gerar_rota(Dados)
```

```
$rota
   var solucao
1 x131       1
2 x241       1
3 x321       1
4 x451       1
5 x511       1

$solucao
 [1] 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 1 0 0 0

$objetivo
[1] 9.15
```

```
vetor_rota<-RouteR::gerar_rota(Dados)$rota$var
#Organizando a rota obtida
RouteR::organizar_rota(vetor_rota)
```

```
  Rota Veiculo
1   13       1
```

```
2   24       1
3   32       1
4   45       1
5   51       1

RouteR::ordenar_rota(vetor_rota)

# A tibble: 5 × 4
  Veiculo Rota  origem destino
    <dbl> <chr>  <dbl>   <dbl>
1       1 13         1       3
2       1 32         3       2
3       1 24         2       4
4       1 45         4       5
5       1 51         5       1
```

A tabela final mostra a rota obtida para o veículo 1 que deve partir do ponto 1 seguindo a rota 1-3-2-4-5-1, visitando todos os clientes, com custo mínimo de 9,15. O custo pode ser interpretado como o tempo em min, ou o custo de combustível em R$ ou ainda a distância em Km. Para este problema foram consideradas 20 variáveis e 20 restrições.

Testes realizados para validação do pacote.

Os testes de validação do pacote RouteR envolveram a aplicação de conjuntos de dados representativos em diversos cenários de roteamento de veículos. Eles foram conduzidos para avaliar a eficácia, precisão e usabilidade do pacote em diferentes configurações de problemas. Os resultados desses testes confirmaram a robustez e confiabilidade do RouteR na modelagem e resolução de problemas de roteamento de veículos, destacando sua utilidade para profissionais e pesquisadores nessa área.

Uma limitação identificada nos testes foi que o pacote RouteR não responde adequadamente para valores de i (número de nós) ou k (número de veículos) maiores que 9. Isso sugere uma restrição em relação à capacidade do pacote em lidar com problemas de roteamento de veículos mais complexos, que envolvem um grande número de nós ou veículos. Esta limitação pode afetar a aplicabilidade do pacote em cenários onde esses valores excedem o limite estabelecido.

Apesar da limitação identificada, o pacote RouteR pode ser qualificado como uma ferramenta útil e eficaz para a modelagem e resolução de problemas de roteamento de veículos de menor escala. Ele oferece funcionalidades específicas que simplificam o processo de modelagem, gerando variáveis, custos e restrições de forma automatizada. Além disso, o

pacote fornece uma interface amigável para os usuários do ambiente R, permitindo uma fácil integração com outras ferramentas e análises. Sua capacidade de gerar soluções para problemas de roteamento de veículos de pequeno a médio porte em um tempo razoável, juntamente com sua facilidade de uso, o torna uma opção valiosa para pesquisadores e profissionais que lidam com esses tipos de problemas.

Conclusão

A análise dos resultados obtidos através do pacote RouteR revela sua eficácia na resolução de problemas de roteamento de veículos (VRP). Inicialmente, ao abordar a criação do pacote, ficou claro que sua concepção foi baseada em lacunas identificadas durante entrevistas com usuários e especialistas, combinadas com uma análise das melhores práticas para a automação do processo de modelagem. A escolha do nome "RouteR" reflete diretamente seu propósito, enfatizando sua função de fornecer ferramentas específicas para lidar com problemas de roteamento de veículos no ambiente R. As funções desenvolvidas no pacote, como **gerar_variaveis**, **gerar_custos**, e **gerar_restricoes**, demonstram uma abordagem sistemática para a preparação de dados e modelagem do problema. Cada função desempenha um papel específico na construção do modelo, simplificando o processo e permitindo uma abordagem mais eficiente e precisa. Além disso, a conveniência proporcionada pelo pacote é evidente na flexibilidade oferecida para ajustar os parâmetros do problema, como o número de nós e veículos. A aplicação prática das funções do pacote RouteR em um exemplo de VRP mostrou sua utilidade na resolução de um problema real. Através da geração de restrições e custos, juntamente com a análise da solução obtida, foi possível identificar sub-rotas e otimizar a rota final, garantindo a visita a todos os clientes com o mínimo custo possível. No entanto, é importante reconhecer que o pacote apresenta limitações quanto ao tamanho do problema, especialmente quando se trata de modelos complexos com um grande número de variáveis e restrições. Portanto, futuras melhorias e aprimoramentos no pacote podem ser explorados para lidar com esses desafios e expandir ainda mais sua aplicabilidade em diferentes cenários de roteamento de veículos. Apesar disso, o pacote RouteR se destaca como uma ferramenta valiosa e eficiente para a modelagem e resolução de problemas de VRP de menor escala.

No entanto, é importante reconhecer que, apesar da eficácia do pacote, ainda existem desafios a serem enfrentados na resolução de problemas de VRP, especialmente quando se trata de modelos complexos com um grande número de variáveis e restrições. Portanto,

futuras melhorias e aprimoramentos no pacote podem ser explorados para lidar com esses desafios e expandir ainda mais sua aplicabilidade em diferentes cenários de roteamento de veículos.

Referências

Alcoforado, Luciane Ferreira. **RouteR: Um pacote para modelagem e resolução de problemas de roteamento de veículos em R** [Software], 2024. Disponível em: https://github.com/LUCIANE-ALCOFORADO/RouteR

Coupey, Julien & Nicod, Jean-Marc & Varnier, Christophe. **VROOM, Vehicle Routing Open-source Optimization Machine**, v1.14. Verso. França, 2024. Disponível em http://vroom-project.org/

Furnon, Vicent & Perron, Laurent. **OR-Tools Routing Library**. V9.9, Google. [Software], 2024. Disponível em: https://developers.google.com/optimization/routing

IBM, **Decision Optimization CPLEX** , Apache License v2.0. [Software], 2024. Disponível em: https://ibmdecisionoptimization.github.io/docplex-doc/

Kuo, Marc. **Open VRP**. V0.6.2. [Software], 2012. Disponível em: https://github.com/mck-/Open-VRP

R Core Team. **R: A language and environment for statistical computing**. R Foundation for Statistical Computing, Vienna, Austria, 2024. URL http://www.R-project.org/

Toth, Paolo & Vigo, Daniele. **Vehicle routing : problems, methods, and applications**. Bologna-Itália: University of Bologna, 2Ed, ISBN 978-1-611973-58-7, 2002.