

Random-Key Optimizer para o Problema de Roteamento de Veículos Capacitados com Entregas e Coletas Reversas

Davi J. F. Alves, Antônio A. Chaves, Cleder M. Schenekemberg

Universidade Federal de São Paulo

Avenida Cesare Mansueto Giulio Lattes, n. 1201. Eugênio de Mello, São José dos Campos, São Paulo, Brasil. CEP: 12247-014

davi.juliano@unifesp.br, antonio.chaves@unifesp.br, cledercms@gmail.com

RESUMO

Este trabalho trata do Problema de Roteamento de Veículos Capacitados com restrições encontradas no contexto de uma empresa logística. O problema envolve um leilão de rotas para uma frota heterogênea que realiza entregas e coletas reversas, buscando otimizar custos e distâncias percorridas com base em uma matriz de distâncias. O processo inclui um pré-planejamento com clusterização em microrregiões e distribuição igualitária de preços de produtos com custos elevados. O objetivo é desenvolver um método de otimização para entrega e coleta reversa de produtos entre depósito e consumidores finais. Para isso, propõe-se o método *Random-Key Optimizer* (RKO), comparando-o com outras abordagens independentes. Além disso, aplica-se o algoritmo *Q-Learning* para ajuste de parâmetros. Os resultados indicam que as metaheurísticas são eficazes, oferecendo boas soluções em tempo computacional viável. Entre as abordagens testadas, VNS e ILS destacaram-se em desempenho geral. Porém, o método RKO com várias metaheurísticas em paralelo foi o que apresentou os melhores resultados para o problema.

PALAVRAS CHAVE. Logística, Roteamento, Metaheurísticas

ABSTRACT

This study addresses the Capacitated Vehicle Routing Problem with constraints, within the specific business context of a logistics company. The problem involves auctioning routes for a heterogeneous fleet that performs both deliveries and reverse pickups, aiming to optimize route costs and distances traveled, based on a predefined distance matrix. The approach includes a pre-planning stage with route clustering into micro-regions and equitable distribution of high-cost product prices. The objective is to develop an optimization method for product delivery and reverse logistics from the warehouse to the final customer. The Random-Key Optimizer (RKO) method is proposed and compared against other independent metaheuristics used in the process. Additionally, parameter tuning is performed using a reinforcement learning algorithm: *Q-Learning*. Preliminary results show that metaheuristics are effective in solving the problem, achieving good solutions within a reasonable computational time. Among the methods tested, VNS and ILS performed well, but RKO stood out as the most effective approach tailored to the problem.

KEYWORDS. Logistics, Routing, Metaheuristics.

1. Introdução

Com o crescimento da tecnologia e também com as mais diversas condições que o mercado oferece, é necessário que haja algoritmos de otimização que realizem processos de alto desempenho. Diante desse fato, metaheurísticas ganham notoriedade, visto que são algoritmos eficientes para resolver problemas extremamente complexos. Neste artigo, o foco será um problema de otimização combinatória pertencente à classe computacional NP-difícil: o problema de roteamento de veículos (VRP, do inglês *Vehicle Routing Problem*). De acordo com [Bressan e de Oliveira Campos, 2020], o VRP pode ser definido como o problema de determinar uma rota a fim de visitar uma série de locais, com o uso de veículos, ao menor custo possível, atendendo a todas as demais imposições do problema, como demandas, tempo e número de veículos disponíveis [Goldbarg e Luna, 2005].

Neste artigo propõe-se a resolução de uma variante do VRP aplicada a um caso prático, envolvendo a otimização das rotas de distribuição de uma empresa logística que opera com veículos convencionais. O problema abordado envolve uma frota heterogênea de veículos, cada um com custos fixos e variáveis distintos, incluindo custos específicos por quilômetro percorrido. O objetivo é minimizar o custo total da operação, considerando tanto as distâncias percorridas quanto os custos individuais de cada veículo. Além da complexidade inerente ao roteamento, o problema incorpora diversas restrições práticas, como a necessidade de agrupar entregas por microrregiões, balancear o valor dos produtos transportados entre as rotas, realizar coletas reversas, evitar zonas de periculosidade e atender a uma lógica de leilão de rotas, na qual a seleção dos veículos é feita com base na relação entre custo e distância. Um estudo estratégico dentro das restrições geradas por um problema de negócios foi conduzido por meio de parcerias com uma empresa do setor de transporte e tecnologia, gerando instâncias reais, com possibilidade de aplicações do problema em questão.

Dada a relevância e a complexidade dos problemas logísticos enfrentados por empresas de distribuição, este trabalho propõe métodos eficientes para a resolução de uma variante prática do VRP. Como ponto de partida, será desenvolvido um modelo matemático que formaliza o problema, incorporando características específicas do contexto real, como frota heterogênea, balanceamento do valor dos produtos transportados, coleta reversa e desvio de zonas de periculosidade.

Em seguida, o problema será tratado por meio do *Random-Key Optimizer* (RKO) [Chaves et al., 2024], um método que integra, de forma estruturada, diferentes técnicas de otimização. O objetivo principal é otimizar a entrega de produtos desde os pontos de armazenamento e distribuição até os consumidores finais, considerando as restrições operacionais envolvidas. A qualidade das soluções obtidas será analisada em função da complexidade do problema, e os resultados serão comparados com aqueles reportados pelo modelo matemático, com o intuito de avaliar a eficácia da abordagem proposta.

Este trabalho está organizado da seguinte forma. A seção 2 apresenta o modelo matemático do VRP abordado. A seção 3 descreve o método RKO e o *decoder* proposto. A seção 4 apresenta os resultados computacionais e na seção 5 faz-se as considerações finais.

2. Definição do Problema

Este trabalho aborda um problema de roteamento de veículos capacitado com dois objetivos, sendo minimizar simultaneamente os custos operacionais e o desbalanceamento das cargas entre veículos da frota, para assim determinar rotas ótimas de um determinado conjunto de clientes. Além das restrições clássicas de capacidade, como volume e peso, o problema incorpora fatores de periculosidade das rotas e penalizações relacionadas ao valor

de produtos transportados, além de tratar uma frota heterogênea com múltiplas unidades de cada tipo de veículo, cuja quantidade é determinada conforme a necessidade de empacotamento da demanda, permitindo a escolha do veículo mais adequado para cada rota. Há uma necessidade de introduzir um conceito de penalidade de periculosidade dentro de um arco (i, j) , pois não é um conceito comumente utilizado na literatura. A penalidade de periculosidade atua como um desincentivo à utilização de rotas que passam por zonas perigosas. Trata-se de uma modelagem de segurança que pode priorizar caminhos potencialmente mais longos ou mais caros, em detrimento de trajetos mais curtos porém arriscados.

A Tabela 1 apresenta os parâmetros de entrada do problema. O conjunto C representa todos os vértices/clientes da rede, incluindo o depósito. Os arcos possíveis (i, j) entre os vértices são definidos por A , e os tipos de veículos são compostos por elementos do conjunto V . Cada arco (i, j) possui uma determinada distância d_{ij} e uma penalidade de periculosidade $Risco_{ij}$. As características individuais de cada cliente, como valor de produto ($PrecoCliente_i$), seu peso ($PesoCliente_i$) e seu volume ($VolumeCliente_i$), são parâmetros relevantes para a composição do problema. Já os veículos possuem capacidades máximas de peso ($PesoCarro_k$) e de volume ($VolumeCarro_k$), além de custos fixos ($CFixoCarro_k$) e custos variáveis ($CVariavelCarro_k$) associados ao uso de cada veículo. Há também dois parâmetros: o λ , que controla o balanceamento da função multiobjetiva, e α , que define a ativação da penalização das zonas de periculosidade dentro da função objetivo.

Parâmetro	Descrição
$C(i)$	Conjunto de vértices, incluindo todos os clientes e o depósito (vértice 0).
$A(i, j)$	Conjunto de arcos possíveis entre os vértices i e j .
$V(k)$	Conjunto de tipos de veículos disponíveis.
d_{ij}	Distância associada ao arco (i, j) .
$Risco_{ij}$	Penalidade associada ao risco ou periculosidade do arco (i, j) .
$VolumeCarro_k$	Capacidade máxima de volume do veículo do tipo k .
$PesoCarro_k$	Capacidade máxima de peso do veículo do tipo k .
$CVariavelCarro_k$	Custo variável por quilômetro do veículo do tipo k .
$PesoCliente_i$	Peso da carga associada ao cliente i . Pesos negativos são relativos a coleta reversa.
$VolumeCliente_i$	Volume da carga associada ao cliente i . Volumes negativos são relativos a coleta reversa.
$CFixoCarro_k$	Custo fixo de utilização do veículo do tipo k .
$PrecoCliente_i$	Valor do produto do cliente i .
λ	Peso de penalização são parâmetros entre 0 e 1 que definem o quanto o modelo deve balancear os preços entre veículos.
α	Peso de penalização são parâmetros entre 0 e 1 que definem o quanto o risco de uma zona de periculosidade deve influenciar no modelo.

Tabela 1: Descrição dos Parâmetros e Dados de Entrada

O modelo matemático proposto foi desenvolvido a partir de uma adaptação do modelo clássico do VRP [Toth e Vigo, 2002], incorporando as particularidades do contexto prático analisado.

As variáveis de decisão utilizadas no modelo são dadas da seguinte forma: $x_{ijk} \in \{0, 1\}$ é uma variável binária que assume o valor 1, caso o arco (i, j) seja percorrido pelo veículo k , e assume 0 caso contrário. De maneira análoga, $y_k \in \{0, 1\}$ é uma variável binária que assume o valor 1, caso o veículo k esteja na solução, e 0 caso contrário. Para as variáveis de balanceamento dos veículos, $\theta_k \in \mathbb{R}_+$ representa o preço total dos produtos alocados a uma determinada rota do veículo k , enquanto $\theta_{max}, \theta_{min} \in \mathbb{R}_+$ representam, respectiva-

mente, o valor máximo e o valor mínimo do somatório das cargas de uma determinada rota alocada a um veículo. As variáveis relacionadas às capacidades são dadas por: $p_{ik} \in \mathbb{R}$ é o peso acumulado da rota do veículo k ao visitar o cliente i e $v_{ik} \in \mathbb{R}$ é o volume acumulado da rota do veículo k ao visitar o cliente i . Finalmente, $u_{ik} \in \mathbb{R}$ é uma variável auxiliar de ordenação responsável pela eliminação de sub-rotas [Miller et al., 1960].

$$\text{Minimizar } \lambda \cdot \sum_{k \in V} \sum_{(i,j) \in A} (CVariavelCarro_k \cdot d_{ij} + \alpha \cdot Risco_{ij}) \cdot x_{ijk} + (1 - \lambda) \cdot (\theta_{max} - \theta_{min}) \quad (1)$$

Sujeito à:

$$\sum_{k \in V} \sum_{j \in C, i \neq j} x_{ijk} = 1, \quad \forall i \in C \setminus \{0\} \quad (2)$$

$$\sum_{j \in C, j \neq 0} x_{0jk} = y_k, \quad \forall k \in V \quad (3)$$

$$\sum_{i \in C, i \neq 0} x_{i0k} = y_k, \quad \forall k \in V \quad (4)$$

$$\sum_{j \in C} x_{ijk} = \sum_{j \in C} x_{jik}, \quad \forall i \in C, k \in V \quad (5)$$

$$\theta_k = \sum_{i \in C} \sum_{j \in C} PrecoCliente_i \cdot x_{ijk}, \quad \forall k \in V \quad (6)$$

$$\theta_{max} \geq \theta_k, \quad \forall k \in V \quad (7)$$

$$\theta_{min} \leq \theta_k + M \cdot (1 - y_k), \quad k \in V \quad (8)$$

$$p_{jk} \geq p_{ik} - PesoCliente_j - M \cdot (1 - x_{ijk}), \quad \forall i, j \in C, i \neq j, k \in V \quad (9)$$

$$-PesoCarro_k \leq p_{ik} \leq PesoCarro_k, \quad \forall i \in C, k \in V \quad (10)$$

$$v_{jk} \geq v_{ik} - VolumeCliente_j - M \cdot (1 - x_{ijk}), \quad \forall i, j \in C, i \neq j, k \in V \quad (11)$$

$$-VolumeCarro_k \leq v_{ik} \leq VolumeCarro_k, \quad \forall i \in C, k \in V \quad (12)$$

$$u_{ik} - u_{jk} + (|C| - 1) \cdot (1 - x_{ijk}) \geq 1, \quad \forall i, j \in C, i \neq j, k \in V \quad (13)$$

$$p_{0k} = 0, \quad v_{0k} = 0, \quad \forall k \in V \quad (14)$$

$$x_{ijk}, y_k \in \{0, 1\}; \quad u_{ik}, \theta_k, \theta_{max}, \theta_{min} \geq 0; \quad p_{ik}, v_{ik} \in \mathbb{R}, \quad \forall i, j \in C, k \in V \quad (15)$$

A função objetivo 1 tem como finalidade ponderar uma escolha utilizando um determinado peso entre o custo total da rota (sendo esse o primeiro termo) e o balanceamento de cargas entre os veículos (segundo termo). O primeiro termo representa o custo logístico, considerando a distância percorrida, o custo operacional de cada veículo e um fator de periculosidade associado ao trajeto. O segundo termo visa minimizar a diferença entre as cargas máximas e mínimas atribuídas aos veículos, promovendo um balanceamento entre os preços dos produtos dentro das rotas. Além disso, há um parâmetro $\lambda \in [0, 1]$ que permite

ajustar a função objetivo em eficiência operacional e a distribuição de cargas. As restrições do modelo estão intimamente ligadas ao problema real de logística proposto a ser resolvido nesse trabalho. A restrição (2) assegura que cada cliente seja atendido exatamente uma vez. As restrições (3) e (4) garantem que todo veículo que for utilizado parta do depósito e também retorne a ele. A restrição (5) impõe a conservação de fluxo para cada cliente e veículo, exigindo que a quantidade de entradas seja igual à de saídas. A restrição (6) define a variável θ_k , que representa o valor total transportado por um veículo k com base nos custos dos produtos dos clientes atendidos pelo veículo. Para balancear economicamente as rotas, as restrições (7) e (8) impõem limites superiores e inferiores ao valor transportado por veículo, permitindo que veículos não utilizados sejam ignorados no cálculo do mínimo via uma constante grande M . As restrições (9) e (10) modelam o peso acumulado, respeitando os limites máximos do veículo, inclusive em casos de coletas reversas (peso negativo). Analogamente, as restrições (11) e (12) controlam o volume acumulado nas rotas. Para evitar a formação de sub-rotas desconectadas do depósito, a restrição (13) aplica a formulação MTZ [Miller et al., 1960]. Finalmente, a restrição (14) assegura que os veículos comecem suas rotas com peso e volume nulos ao sair do depósito, e as restrições (15) são as restrições de domínio das variáveis do modelo.

3. Random-Key Optimizer

O *Random-Key Optimizer* (RKO) é uma estrutura metaheurística eficiente fundamentada no conceito do algoritmo genético de chaves aleatórias (*Random-Key Genetic Algorithm*, RKGa), originalmente proposto por [Bean, 1994]. Essa abordagem codifica soluções como vetores de chaves aleatórias (números reais uniformemente distribuídos no intervalo $[0, 1)$), permitindo uma representação única de problemas de otimização combinatória em um espaço de busca contínuo [Chaves et al., 2024].

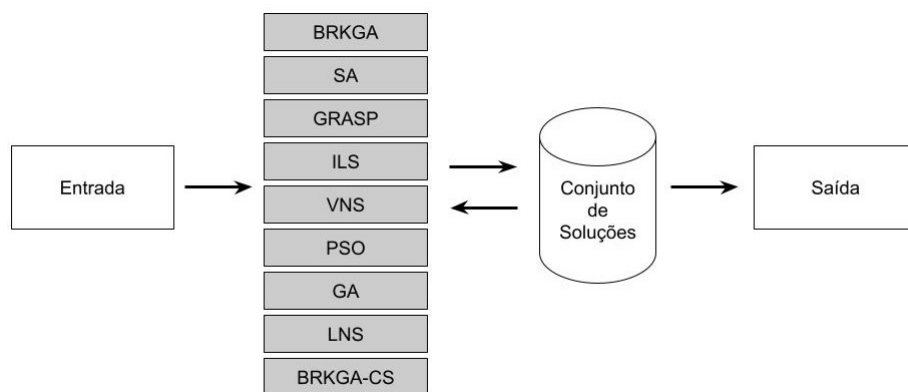


Figura 1: O *pipeline* de otimização do RKO (Fonte: [Chaves et al., 2024]).

O processo de otimização do RKO, como ilustrado pela Figura 1, envolve a interação entre diferentes metaheurísticas que compartilham e refinam soluções de maneira colaborativa, por meio de um conjunto dinâmico de elite que armazena as melhores soluções encontradas ao longo da execução [Chaves et al., 2024]. O funcionamento do RKO pode ser decomposto em seis principais componentes, sendo eles: soluções iniciais, conjunto de soluções elite, perturbações, combinações, buscas locais e metaheurísticas.

A inicialização do processo se dá com a geração de soluções representadas por vetores $\chi \in [0, 1)^n$, cada um avaliado pela função objetivo aplicada à sua decodificação. Um **conjunto inicial** com γ soluções é construído e mantido ao longo da execução, sendo

atualizado sempre que uma nova solução promissora supera a qualidade das existentes. Para preservar a diversidade, clones são perturbados por operadores específicos.

A **perturbação** de solução é controlada pelo parâmetro β , escolhido de maneira aleatória em um intervalo definido, que determina quantas alterações serão realizadas no vetor solução. Quatro movimentos são aplicados aleatoriamente sobre as chaves aleatórias: (Troca, Troca com Vizinho, Espelhamento e Aleatório).

Já o mecanismo de **combinações** gera novas soluções a partir de dois vetores pais. Um parâmetro de herança probabilística ρ define de qual vetor pai cada posição do novo vetor combinado será herdada. O parâmetro *factor* permite modular o vetor secundário, podendo aplicar complemento da chave, enquanto uma taxa de mutação μ injeta valores totalmente novos, aumentando a diversidade da população.

3.1. Buscas Locais

O procedimento de busca local adotado no RKO é o algoritmo *Randomized Variable Neighborhood Descent* proposto por [Subramanian et al., 2010] e aplicado ao contexto de chaves aleatórias por [Chaves et al., 2024]. O RVND explora diversas estruturas de vizinhanças $N_k(\chi)$, selecionadas de forma aleatória a partir de uma lista inicial. A cada iteração, busca-se uma solução melhor dentro da vizinhança escolhida, ou seja, caso haja uma melhoria, a solução atual é atualizada e a lista de vizinhanças é reiniciada, e caso não haja nenhuma melhoria, essa vizinhança é descartada. Esse processo continua até que todas as vizinhanças sejam exploradas.

Para esse problema, foram desenvolvidas quatro buscas locais diretamente ligadas ao problema abordado. Essas heurísticas constituem as estruturas de vizinhança distintas integradas no RVND, que são: *Swap* Intra-rota, *Swap* Inter-rota (ambas descritas na Subseção 3.1.1, o 2-Opt Intra-rota e o 2-Opt Inter-rota (ambas descritas na Subseção 3.1.2).

3.1.1. Movimentos de *Swap* Intra e Inter-rota

O movimento de *swap intra-rota* ocorre dentro de uma mesma rota. Dois nós são selecionados e têm suas posições trocadas, mantendo os demais pedidos da rota em suas respectivas posições. Essa troca é avaliada com base na nova distância total da rota resultante, ou seja, se a troca leva a uma redução na função objetivo, ela é aceita e, caso contrário, a solução original é mantida. Esse procedimento é aplicado de forma exaustiva até que nenhuma troca adicional traga melhoria. Por atuar apenas dentro das rotas, o *swap* intra-rota é útil para refinar rotas já viáveis, buscando sequências mais eficientes de atendimento dos clientes.

Já o movimento de *swap inter-rota* envolve dois nós pertencentes a rotas diferentes. Nesse caso, um nó de uma rota r_n é trocado com outro nó de uma rota r_m . A troca é realizada preservando a ordem dos demais elementos nas duas rotas. Após a troca, o algoritmo recalcula o custo total das duas rotas envolvidas e mantém a troca somente se houver uma redução na função objetivo combinada e se a troca for factível. Esse tipo de movimento é particularmente útil para redistribuir a carga entre rotas e corrigir alocações ineficientes, atuando não apenas na sequência dos atendimentos, mas também na composição das rotas. O *Swap* intra-rota e inter-rota podem ser observados na Figura 2.

3.1.2. Movimentos de 2-Opt Intra e Inter-rota

Os movimentos de *2-Opt* são operadores clássicos de busca local amplamente utilizados para melhorar sequências em problemas de roteamento. Seu princípio baseia-se na remoção de duas arestas e na reconexão dos segmentos resultantes de maneira a reduzir o custo total da solução.



Figura 2: Movimentos de troca entre pedidos em uma ou diferentes rotas.

O movimento de *2-Opt intra-rotas* atua dentro de uma mesma rota. Duas arestas não adjacentes são removidas e as subsequências intermediárias são revertidas antes da reconexão, criando uma nova sequência de visita. O algoritmo percorre todas as possíveis combinações de pares de arestas na rota e avalia se a nova estrutura resulta em uma função objetivo menor. Formalmente, ao selecionar dois nós i e j com $i < j$, o segmento entre eles é invertido, e as conexões anteriores são substituídas por novas arestas (i, j) e $(i + 1, j + 1)$. A troca é aceita somente se a nova rota tiver um custo na função objetivo inferior ao original. Esse processo é repetido até que nenhuma troca adicional leve a melhorias, resultando em rotas mais eficientes com menor sobreposição de trajetos.

Na variante de *2-Opt inter-rotas*, o conceito é estendido para pares de rotas distintos. Duas arestas são removidas — uma de cada rota — e os segmentos restantes são trocados entre as rotas. Isso modifica a composição e sequência de ambas as rotas envolvidas. Esse movimento é útil para redistribuir pedidos entre rotas e corrigir rotas com trajetos ineficientes. A troca é mantida apenas se a função objetivo total combinada das duas rotas for reduzida após a modificação. Como no caso intra-rotas, o processo pode ser repetido iterativamente. O 2-Opt intra-rotas e inter-rotas podem ser observados na Figura 3.



Figura 3: Movimentos 2-Opt aplicados dentro de uma mesma rota ou entre rotas distintas.

3.2. Metaheurísticas

Nesse contexto, cada meta-heurística oferece uma abordagem para explorar o espaço de busca, aproveitando as funcionalidades da representação de chaves aleatórias e do mecanismo de decodificação do RKO. Todas as meta-heurísticas iniciam o processo de busca a partir de vetores de chaves aleatórias gerados aleatoriamente. Após a geração das soluções iniciais, cada meta-heurística segue seu próprio paradigma de busca. O RKO também utiliza um limite de tempo de CPU predefinido como critério de parada, garantindo que todas as meta-heurísticas sejam executadas com o mesmo tempo computacional [Chaves et al., 2024]. Neste trabalho, será utilizado um ajuste de parâmetros online utilizando um algoritmo de aprendizado de máquinas por reforço: o método *Q-Learning*.

Há a utilização de metaheurísticas robustas dentro do método RKO, tais como *Variable Neighborhood Search* (VNS), *Iterated Local Search* (ILS), *Greedy Randomized Adaptive Search Procedure* (GRASP), *Biased Random-Keys Genetic Algorithm* (BRKGA), *Biased Random-Keys Genetic Algorithm with Clustering Search* (BRKGA-CS), *Genetic Algorithm* (GA), *Simulated Annealing* (SA), *Particle Swarm Optimization* (PSO) e *Large Neighborhood Search* (LNS).

3.3. Decoders para o CVRP abordado

O método RKO utilizado neste trabalho precisa de uma heurística construtiva para as soluções envolvendo as chaves aleatórias, e esse método dentro do processo é denominado *Decoder*. Para o problema específico, foram desenvolvidos dois tipos de *decoders* distintos. Os *decoders* do trabalho são heurísticas baseadas na ordenação das chaves aleatórias e na inserção nas rotas de acordo com suas regras distintas.

O *decoder* de Ordenação das Chaves Aleatórias é uma heurística construtiva que atribui pedidos às rotas com base na ordenação crescente de chaves aleatórias associadas a cada pedido. Como o número de rotas não é previamente conhecido, o *decoder* aloca os pedidos sequencialmente até que os limites de capacidade de peso e volume dos veículos sejam atingidos, gerando novas rotas conforme necessário. Formalmente falando, seja um vetor A de chaves aleatórias associadas aos pedidos do conjunto N de pedidos, e uma quantidade $m < n$ de rotas. Cada rota R_j , com $j \leq m$, possui capacidade de peso W_j e volume V_j , enquanto cada pedido $i \in N$ possui peso w_i e volume v_i . O vetor A é ordenado, e os pedidos $a_i \in A$ são alocados em sequência à última posição da rota R_j até que as restrições de capacidade (Equações 16 e 17) sejam violadas, momento em que uma nova rota é iniciada.

$$\sum_{a_i \in R_j} v_{a_i} \leq V_j, \forall j < m \quad (16)$$

$$\sum_{a_i \in R_j} w_{a_i} \leq W_j, \forall j < m \quad (17)$$

O *decoder* de Ordenação das Chaves Aleatórias com Melhor Inserção é uma heurística construtiva que combina a ideia do *decoder* de Ordenação das Chaves Aleatórias com uma estratégia de inserção otimizada. A ideia é alocar os pedidos seguindo a ordem das chaves, inserindo cada um na posição da rota que resulte no menor aumento de distância, respeitando as restrições de capacidade. Formalmente falando, seja um vetor A de chaves aleatórias associadas aos pedidos do conjunto N de pedidos, e uma quantidade $m < n$ de rotas. Cada rota R_j , com $j \leq m$, possui capacidade de peso W_j e volume V_j , enquanto cada pedido $i \in N$ possui peso w_i e volume v_i . O vetor A é ordenado, e o pedido $a_i \in A$ é alocado na posição da rota R_j que minimiza o custo incremental, até que as restrições de capacidades (Equações 16 e 17) sejam atingidas e a rota seja encerrada.

Os dois decoders fazem o empacotamento nas rotas via restrição de peso ou de volume de um veículo, entretanto, há uma decisão em escolher outro carro com um suporte para peso maior para aumentar a rota ou fechar a rota e abrir uma nova. A decisão é descrita por $\Delta_{custo} = R_1 \cdot c_i - R_2 \cdot c_j$, sendo R_1 a distância total da rota inicial, R_2 a distância total da rota inicial com os novos produtos até lotar o próximo carro, c_1 o custo por km do carro i e c_2 o custo por km do carro j . Se o $\Delta_{custo} \leq 0$, isso significa que é mais custoso utilizar do carro c_j , e caso o $\Delta_{custo} > 0$, é mais custoso utilizar do carro c_i .

A Figura 4a demonstra o processo de ordenação e a criação das rotas a partir da ordenação. Note que as rotas se dão por inserção, e o pressuposto é de inserção de 3 pedidos

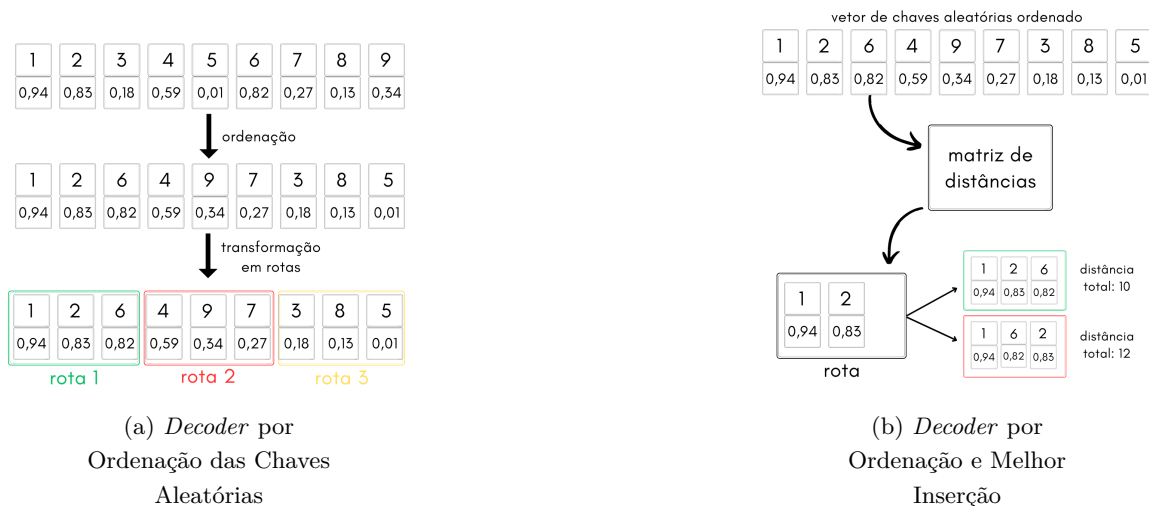


Figura 4: Exemplos dos *Decoders* utilizados para 8 vértices (Fonte: Autor).

em cada rota. Já a Figura 4b demonstra o processo de ordenação e criação das rotas. Note que as rotas se dão por inserção e busca da melhor posição, e o pressuposto é de inserção de 3 pedidos em cada rota. O vértice 6 é inserido na posição que minimiza a distância total, com base na matriz de distâncias. A melhor inserção (em verde) é aceita, as demais são rejeitadas, e a rota é finalizada com os três pedidos. Nesse exemplo, não foi abordada a dinâmica de decisão dos carros, apenas a do empacotamento.

4. Testes Computacionais

As instâncias utilizadas nesse trabalho são instâncias inspiradas no problema real em um dia de trabalho comum de uma empresa logística. Os dados foram anonimizados, e as instâncias foram divididas em três grandes conjuntos: o $CVRP_{Reduzido}$, o $CVRP_1$ e o $CVRP_2$. As instâncias do $CVRP_{Reduzido}$ são instâncias com os 25, 50, 75 e 100 vértices mais próximos do depósito de instâncias do conjunto $CVRP_1$. O conjunto de instâncias do $CVRP_1$ descreve instâncias de 117 a 544 vértices, enquanto o conjunto de instâncias do $CVRP_2$ descreve instâncias de 700 a 916 vértices. O $CVRP_{Reduzido}$ é composto de 9 instâncias-base, com 36 reduções, enquanto $CVRP_1$ possui 22 instâncias-base e $CVRP_2$ possui 6 instâncias-base. A matriz de distâncias, nesse caso, é um dado de entrada da instância e está diretamente ligada aos vértices das instâncias. Estas instâncias estão disponíveis em [inserir-apos-revisao](#).

O RKO foi implementado em C++ e seu *framework* está disponível no trabalho de [Chaves et al., 2024]. Os testes computacionais foram executados em um computador equipado com uma CPU Intel i9-12990K com 128GB de RAM. Já o modelo matemático foi executado no solver Gurobi 12.0.1, em um computador equipado com uma CPU Intel i5-13400F com 32GB de RAM. Os testes para as instâncias do conjunto $CVRP_{Reduzido}$ possuem um tempo limitado de 1200 segundos para um $\lambda \in \{0, 0, 5, 1\}$, enquanto as instâncias do conjunto $CVRP_1$ e do $CVRP_2$ foram executadas em um tempo limite de 3600 segundos com um $\lambda \in \{0, 0, 5, 1\}$ tanto para o modelo quanto para as metaheurísticas. Nas Tabelas 2 e 3 há uma comparação sumarizada de desempenho de cada metaheurística nas instâncias de teste, apresentando a média de melhor solução (MS), a média do RPD de melhor solução (RPD_{ms}), o RPD da média de melhor solução (RPD_{avg}), o desvio padrão médio de todas as

execuções do método (σ_{med}), a média dos tempos de execução até a melhor solução (Tempo (s)) e a média da diferença da melhor solução para a melhor solução obtida (#BKS).

A Tabela 2 apresenta o desempenho dos métodos no conjunto $CVRP_{Reduzido}$ para diferentes valores de $\lambda \in (1, 0, 0, 5, 0, 0)$. Para $\lambda = 1, 0$, o método RKO obteve o menor custo médio ($MS = 34814, 49$), sendo o mais eficiente nesse cenário, ainda que com menor qualidade que a solução ótima de referência encontrada pelo GUROBI ($MS = 23199, 92$). Já para $\lambda = 0, 5$, o RKO foi o destaque, com o menor RPD_{ms} (2,72) e RPD_{avg} (3,90), mantendo ótimo equilíbrio entre qualidade e tempo. Os métodos RKO-VNS e RKO-ILS também tiveram boa performance, com RPD_{ms} em torno de 46%. E para $\lambda = 0, 0$, apenas RKO, RKO-PSO e RKO-LNS conseguiram manter resultados razoáveis, e tudo isso se dá ao baixo valor da FO, quando falamos sobre o equilíbrio dos carros.

Tabela 2: Resultados computacionais para o conjunto $CVRP_{Reduzido}$.

λ	Método	MS	RPD_{ms}	RPD_{avg}	σ_{med}	Tempo (s)	#BKS
1,0	RKO	34814,49	49,97	51,46	494,54	609,15	10403,47
	RKO-BRKGGA	36388,33	56,38	60,90	1570,00	481,36	11977,31
	RKO-SA	50364,62	101,90	107,35	1980,39	676,10	25953,60
	RKO-GRASP	36287,26	57,38	59,75	525,47	564,66	11876,24
	RKO-ILS	36554,15	56,82	58,78	667,90	524,34	12143,13
	RKO-VNS	36459,93	56,35	58,62	725,12	534,02	12048,91
	RKO-PSO	35942,14	56,78	59,45	875,39	635,37	11531,12
	RKO-GA	36878,97	60,40	62,84	756,60	464,60	12467,96
	RKO-BRKGGA-CS	38738,85	67,61	71,31	1150,24	681,96	14327,83
	RKO-LNS	48416,51	97,23	106,99	3754,79	139,97	24005,49
	GUROBI	23199,92	0,58	-	-	1200	247,35
0,5	RKO	15805,49	2,72	3,90	280,06	473,03	756,61
	RKO-BRKGGA	19980,19	49,54	52,33	701,96	527,14	4931,31
	RKO-SA	26563,07	82,00	85,26	631,54	653,07	11514,20
	RKO-GRASP	19648,33	48,10	50,11	258,05	614,82	4599,46
	RKO-ILS	19762,13	45,99	47,54	301,68	646,87	4713,25
	RKO-VNS	19833,12	46,38	47,83	239,24	612,57	4784,24
	RKO-PSO	19304,68	46,62	48,49	420,37	602,89	4255,81
	RKO-GA	20005,81	50,20	52,21	339,44	553,96	4956,93
	RKO-BRKGGA-CS	20874,53	56,50	59,22	475,95	607,68	5825,66
	RKO-LNS	26209,15	80,99	87,96	1859,62	196,74	11160,28
	GUROBI	81178,63	202,52	-	-	1200	66.129,75
0,0	RKO	1,95	0,01	9,79	57,30	363,27	1,95
	RKO-BRKGGA	5,30	0,09	5,68	3,78	279,59	5,30
	RKO-SA	10,64	136,69	228,71	0,04	319,65	10,64
	RKO-GRASP	68,12	50253,91	281313,07	30,64	276,65	68,12
	RKO-ILS	105,71	67772,35	307705,54	14,26	350,96	105,71
	RKO-VNS	88,55	30167,34	1059826,45	7,51	328,79	88,55
	RKO-PSO	0,13	125,51	238,46	0,08	337,02	0,13
	RKO-GA	10,67	2634,92	2731,13	0,28	341,40	10,67
	RKO-BRKGGA-CS	10,99	1589,38	8134,46	1,23	344,81	10,99
	RKO-LNS	2,29	384,88	549,39	3,52	337,32	2,28
	GUROBI	188,90	1487485,65	-	-	1200	164,28

A Tabela 3 apresenta o desempenho dos métodos nos conjuntos $CVRP_1$ e $CVRP_2$ para diferentes valores de $\lambda \in (1, 0, 0, 5, 0, 0)$. Para $\lambda = 1,0$, o método RKO apresentou o melhor desempenho entre as heurísticas, com $RPD_{ms} = 6,67$ e $RPD_{avg} = 8,50$, superando os demais em qualidade de solução. E os métodos como RKO-VNS e RKO-ILS também se destacaram, com RPD abaixo de 13% e tempos de execução abaixo de 2000 segundos. Já para $\lambda = 0,5$, o RKO novamente obteve o melhor resultado ($RPD_{ms} = 0,91$), com tempos abaixo de 1800 segundos. RKO-ILS, RKO-GRASP e RKO-VNS também apresentaram soluções próximas ao ótimo, com baixo desvio-padrão, mostrando estabilidade e boa adaptabilidade. E para $\lambda = 0,0$, diversos algoritmos apresentaram valores absurdamente elevados de RPD , como RKO-ILS ($> 10^7$) e RKO-GRASP ($> 10^7$), evidenciando instabilidade completa no método, com apenas o RKO, RKO-PSO, e RKO-LNS mantendo níveis de RPD abaixo de 10^5 .

Tabela 3: Resultados computacionais para os conjuntos $CVRP_1$ e $CVRP_2$.

λ	Método	MS	RPD_{ms}	RPD_{avg}	σ_{med}	Tempo (s)	#BKS
1,0	RKO	$4,21 \times 10^5$	6,67	8,50	$1,05 \times 10^4$	1989,52	$7,59 \times 10^3$
	RKO-BRKGGA	$4,87 \times 10^5$	19,99	25,89	$2,80 \times 10^4$	2078,12	$7,38 \times 10^4$
	RKO-SA	$1,25 \times 10^6$	170,59	177,75	$4,83 \times 10^4$	3080,76	$8,42 \times 10^5$
	RKO-GRASP	$4,43 \times 10^5$	12,36	14,10	$9,88 \times 10^3$	1664,40	$2,99 \times 10^4$
	RKO-ILS	$4,30 \times 10^5$	11,32	13,28	$8,28 \times 10^3$	1791,57	$1,74 \times 10^4$
	RKO-VNS	$4,29 \times 10^5$	11,05	12,72	$8,22 \times 10^3$	1719,44	$1,54 \times 10^4$
	RKO-PSO	$5,32 \times 10^5$	25,30	29,22	$2,40 \times 10^4$	2453,86	$1,19 \times 10^5$
	RKO-GA	$4,47 \times 10^5$	16,35	19,06	$1,55 \times 10^4$	1746,26	$3,41 \times 10^4$
	RKO-BRKGGA-CS	$4,73 \times 10^5$	23,34	28,28	$2,07 \times 10^4$	1978,06	$5,98 \times 10^4$
	RKO-LNS	$5,08 \times 10^5$	44,23	51,62	$3,16 \times 10^4$	432,97	$9,48 \times 10^4$
	GUROBI	$1,02 \times 10^5$	2,01	-	-	3600	$1,44 \times 10^3$
0,5	RKO	$2,13 \times 10^5$	0,91	2,54	$5,66 \times 10^3$	1716,79	$1,19 \times 10^3$
	RKO-BRKGGA	$2,53 \times 10^5$	17,04	22,55	$1,74 \times 10^4$	1859,96	$4,13 \times 10^4$
	RKO-SA	$6,35 \times 10^5$	159,42	164,63	$1,90 \times 10^4$	2890,42	$4,24 \times 10^5$
	RKO-GRASP	$2,29 \times 10^5$	6,23	7,95	$4,98 \times 10^3$	1925,30	$1,75 \times 10^4$
	RKO-ILS	$2,17 \times 10^5$	3,47	5,60	$5,56 \times 10^3$	1831,01	$5,84 \times 10^3$
	RKO-VNS	$2,18 \times 10^5$	4,38	5,88	$5,17 \times 10^3$	1416,35	$7,00 \times 10^3$
	RKO-PSO	$2,72 \times 10^5$	20,76	23,84	$9,44 \times 10^3$	2392,60	$6,08 \times 10^4$
	RKO-GA	$2,34 \times 10^5$	11,35	13,25	$5,17 \times 10^3$	1780,94	$2,26 \times 10^4$
	RKO-BRKGGA-CS	$2,45 \times 10^5$	17,25	19,91	$6,10 \times 10^3$	1795,15	$3,38 \times 10^4$
	RKO-LNS	$2,58 \times 10^5$	31,43	38,93	$1,27 \times 10^4$	188,59	$4,68 \times 10^4$
	GUROBI	$1,67 \times 10^5$	111,49	-	-	3600	$1,11 \times 10^5$
0,0	RKO	36,01	10179,49	17559,03	16,41	2049,77	2,12
	RKO-BRKGGA	39,13	122502,40	135773,14	47,22	2092,86	5,24
	RKO-SA	363,43	61843,76	117554,36	26,94	2293,27	329,55
	RKO-GRASP	910,26	$5,51 \times 10^6$	$2,03 \times 10^7$	131,73	1624,04	876,37
	RKO-ILS	779,62	$1,06 \times 10^7$	$2,21 \times 10^7$	123,52	1781,80	745,73
	RKO-VNS	741,13	$5,60 \times 10^6$	$7,19 \times 10^6$	100,27	1691,24	707,24
	RKO-PSO	330,31	1603,29	32016,98	20,76	2436,54	296,43
	RKO-GA	414,34	$2,38 \times 10^5$	$4,71 \times 10^5$	65,52	2200,98	380,45
	RKO-BRKGGA-CS	457,52	$3,48 \times 10^5$	$4,61 \times 10^5$	44,38	2283,39	423,64
	RKO-LNS	420,44	$1,04 \times 10^5$	$1,46 \times 10^5$	81,88	2184,31	386,55
	GUROBI	$2,78 \times 10^4$	$1,11 \times 10^9$	-	-	3600	$2,78 \times 10^4$

Apesar de seu desempenho ótimo em instâncias menores, o GUROBI apresentou limitações nas instâncias do conjunto completo $CVRP_1$ e $CVRP_2$, sendo capaz de resolver apenas 7 das 28 instâncias dentro do tempo limite de 3600 segundos. Isso mostra que sua escalabilidade é comprometida em cenários com maior complexidade computacional.

5. Conclusão

Como o problema proposto é multi-objetivo, a definição do λ é de extrema importância para relacionar qual objetivo de otimização é pretendido ser resolvido. Neste estudo, os experimentos computacionais foram realizados para comparar o desempenho das metaheurísticas propostas, avaliar e encontrar boas soluções para roteamento de veículos restrito a condições reais de uma corporação. Os resultados preliminares mostram uma performance superior de algumas metaheurísticas em detrimento de outras, mostrando a superioridade do método proposto: o RKO. Em relação à modelagem matemática e ao solver exato GUROBI, sua eficácia foi evidente em instâncias pequenas, fornecendo soluções ótimas de referência para $\lambda \in (1, 0, 0, 5)$. No entanto, nas instâncias maiores dos conjuntos $CVRP_1$ e $CVRP_2$ para o mesmo λ , sua escalabilidade foi comprometida: o solver conseguiu resolver apenas 7 das 28 instâncias dentro do tempo limite, confirmando que, embora forneça soluções de alta qualidade, sua aplicabilidade prática é limitada em cenários de maior porte. Alguns pontos de melhora em trabalhos futuros seria de justamente utilizar melhores métodos de busca local, e testar *decoders* mais adaptados ao problema de roteamento. Finalmente, gerar outras instâncias adaptadas ao problema real abordado e disponibilizá-las na literatura.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, e da FAPESP (2024/08848-3).

Referências

- Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, 6(2):154–160.
- Bressan, G. M. e de Oliveira Campos, L. A. (2020). Otimização de problemas de roteamento de veículos: soluções heurísticas. *CQD-Revista Eletrônica Paulista de Matemática*.
- Chaves, A. A., Resende, M. G., de Arruda, E. F., e Silva, R. (2024). A random-key optimizer for combinatorial optimization. *arXiv preprint arXiv:2411.04293*.
- Goldbarg, M. C. e Luna, H. P. L. (2005). *Otimização combinatória e programação linear: modelos e algoritmos*. Elsevier.
- Miller, C. E., Tucker, A. W., e Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329.
- Subramanian, A., Drummond, L. M., Bentes, C., Ochi, L. S., e Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899–1911.
- Toth, P. e Vigo, D. (2002). Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, 123(1-3):487–512.