Aplicação da Decomposição Lógica de Benders para o Problema de Roteamento de Veículos com Múltiplos Depósitos

Integrante(s): José Eduardo Saroba Bieco
E-mail(s): jose.bieco@usp.br

Resumo

O Problema de Roteamento de Veículos com Múltiplos Depósitos (MDVRP) é uma variação do Problema de Roteamento de Veículos (VRP). Este estudo compara à Decomposição Lógica de Benders (LBBD) com a Programação Inteira Mista (MIP) para resolver o MDVRP. Foram realizados testes em diferentes instâncias, variando o número de clientes, depósitos e veículos. Os resultados indicam que, embora o MIP tenha apresentado soluções mais rápidas, a LBBD alcançou menores custos totais, especialmente em cenários de alta complexidade. Para problemas complexos, nos quais a qualidade da solução é prioritária, concluiu-se que a LBBD se mostra mais adequada.

Palavras-chave: Decomposição de Benders, Roteamento de Veículos, Múltiplos depósitos

1 Introdução e trabalhos relacionados

O Problema de Roteamento de Veículos com Múltiplos Depósitos (MDVRP, do inglês *Multi-Depot Vehicle Routing Problem*) é uma das variações mais complexas do Problema de Roteamento de Veículos (VRP), que tem como objetivo minimizar as rotas percorridas pelos veículos e garantir que todos os clientes sejam atendidos.

No contexto de operações logísticas, onde empresas precisam gerenciar múltiplos centros de distribuição, a solução eficiente do MDVRP pode gerar reduções significativas de custos operacionais e melhorias na eficiência do atendimento ao cliente (Surekha and Sumathi (2011)).

Uma das abordagens mais exploradas é o uso de Algoritmos Genéticos (GA), que se destacam por sua eficiência em resolver problemas combinatórios complexos. Pesquisas como as de Surekha and Sumathi (2011) e Mahmud and Haque (2019) ressaltam que devido às características estocásticas e à capacidade de otimização dos GAs, esses algoritmos são amplamente aplicados para encontrar soluções tanto exatas quanto aproximadas para o problema.

Em Surekha and Sumathi (2011) os autores propõem um método em que os clientes são inicialmente agrupados com base na proximidade aos depósitos, e em seguida, as rotas são otimizadas usando o método de economia de *Clarke e Wright* [1], seguido pela aplicação do GA para refinar as rotas. Essa abordagem mostrou-se eficaz, especialmente quando comparada com técnicas presentes no estado da arte em termos de comprimento das rotas, distância total e tempo computacional.

O método desenvolvido por Mahmud and Haque (2019) consiste também em três partes: a geração da primeira população utilizando um algoritmo de clusterização para agrupar os clientes, em seguida aplica-se um GA para otimizar as rotas de atendimento para cada depósito e por fim, o algoritmo é utilizado novamente para construir a estrutura de rede que conecta todas as rotas obtidas. Esta abordagem foi avaliada em comparação com métodos existentes, como o *GenClust*, e mostrou-se promissora para problemas complexos e de grande escala.

Outra abordagem empregada para encontrar soluções ótimas para o MDVRP são os métodos exatos. Contardo and Martinelli (2014) propuseram uma solução que combina a abordagem de fluxo de veículos com a de partição de conjunto. Os autores introduziram uma nova família de desigualdades válidas para evitar ciclos arbitrários e utilizaram o limite inferior calculado na formulação de fluxo de veículos para eliminar arestas não promissoras. O algoritmo demonstrou eficácia ao resolver algumas instâncias anteriormente não resolvidas. Além disso, para as instâncias que não puderam ser resolvidas, o método gerou limites inferiores mais fortes do que os obtidos por técnicas anteriores (Jayarathna et al. (2021)).

Com base nos trabalhos mencionados anteriormente, a Tabela 1 apresenta a quantidade de instâncias, bem como os números mínimo e máximo de clientes, depósitos e veículos considerados em cada pesquisa.

Tabela 1: Resumo das Instâncias, Clientes e Depósitos por Pesquisa

Pesquisa	**Instâncias**	**Clientes**	**Depósitos**	**Veículos**
[1]	5	50 – 100	2 – 5	20 – 35
[2]	10	20 – 250	1 – 5	Não informado
[3]	17	50 – 360	2 – 9	2 – 14

[1] Surekha and Sumathi (2011), [2] Mahmud and Haque (2019), [3] Contardo and Martinelli (2014)

Este trabalho explora duas metodologias para resolver o MDVRP: a Decomposição Lógica de

[1] Trata-se de uma heurística iterativa de construção baseada em uma função gulosa de inserção.

Benders e a Programação Inteira Mista (MIP). A Decomposição de Benders é muito utilizada em para problemas de grande escala, pois permite a divisão do problema original em subproblemas menores, facilitando a obtenção de soluções eficientes (Rahmaniani et al. (2017)).

A disposição das demais seções esta organizada da seguinte forma: a seção 2 especifica o MDVRP. Em seguida, a seção 3 aborda o método de Decomposição de Benders. A seção 4 descreve o ambiente de teste, as instâncias utilizadas e a análise dos resultados. E a última seção, 5, resume as principais conclusões e propõe sugestões para trabalhos futuros.

2 Definição do problema

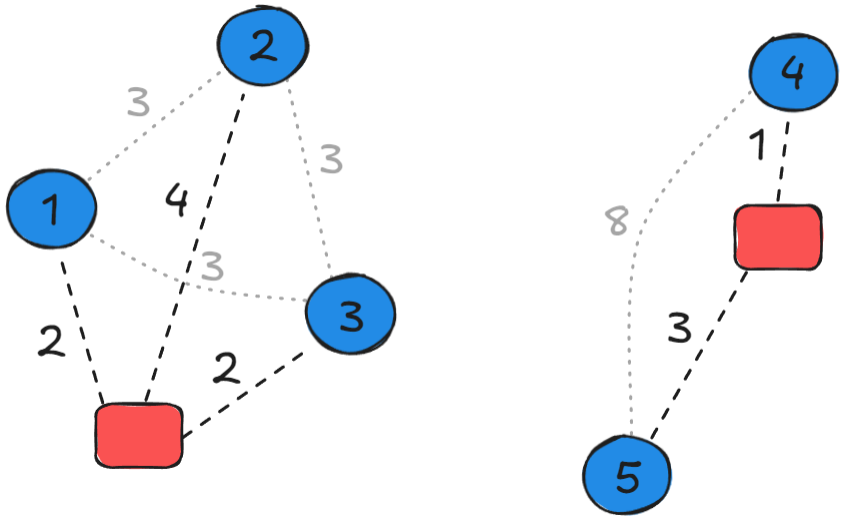
O MDVRP trata-se de um problema de otimização cujo objetivo é determinar as rotas ótimas para múltiplos veículos, partindo de diferentes depósitos, para atender às demandas de clientes localizados em regiões distintas, garantindo o retorno ao depósito de origem (Surekha and Sumathi (2011)).

Pesquisas como a proposta por Castro et al. (2024) introduzem penalidades ou custos adicionais a certas ações com o objetivo de restringir ou tornar mais realistas as soluções obtidas. Por exemplo, pode-se incluir um custo para a utilização de um depósito, ou aplicar uma penalidade por atrasos nas entregas entre clientes, etc.

A solução busca minimizar o custo global, que inclui tanto o número de veículos utilizados quanto a distância percorrida, enquanto assegura que todas as demandas dos clientes sejam completamente atendidas (Mahmud and Haque (2019)).

A Figura 1 ilustra um exemplo do MDVRP com 2 depósitos e 5 clientes, apresentando a localização dos depósitos e clientes, bem como as distâncias entre eles. Observa-se que cada cliente foi previamente alocado ao depósito mais próximo. Deve ser considerado também o custo de 5 unidades para a utilização de cada veículo.

Figura 1: Posição dos clientes depósitos e custos



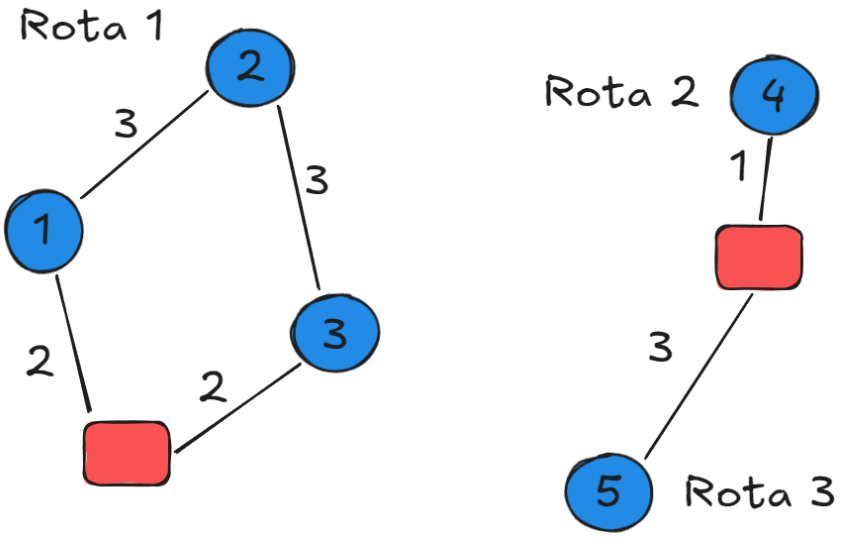
Fonte: próprio autor

Os depósitos são representados por retângulos vermelhos, enquanto os clientes são indicados por círculos azuis. As linhas tracejadas mostram as distâncias entre os clientes e os depósitos, e as linhas pontilhadas indicam as distâncias entre os próprios clientes.

As rotas ótimas são determinadas pela análise das combinações que envolvem partir de um depósito d, seguir uma sequência u de clientes e retornar ao ponto inicial. A Figura 2 ilustra essas

rotas, que minimizam o custo global ao mostrar as rotas mais curtas entre os clientes e a alocação mínima de veículos necessária.

Figura 2: Rotas ótimas para cada depósito



Fonte: próprio autor

Para modelar matematicamente as rotas ótimas descritas anteriormente, conforme apresentado por Contardo and Martinelli (2014), o MDVRP pode ser formulado como um problema de programação inteira mista. Seja $G = (V, A)$ um grafo completo, onde $V = \{i \in I, j \in J\}$ representa o conjunto de nós, com I indicando os depósitos e J os clientes. O conjunto $A = \{(i, j) : i, j \in V, i \neq j\}$ denota os arcos que conectam todos os pares de nós. Define-se K como o conjunto de veículos disponíveis, V_i como a capacidade do depósito $i \in I$, e d_j como a demanda do cliente $j \in J$.

De acordo com a formulação proposta por Surekha and Sumathi (2011) e considerando a inclusão de uma penalidade para cada veículo alocado, o modelo de otimização pode ser representado da seguinte maneira:

- **Conjuntos:**
  - I: Conjunto de depósitos.
  - J: Conjunto de clientes.
  - K: Conjunto de veículos.
- **Parâmetros:**
  - N: Número de veículos.
  - C_{ij}: Distância entre os pontos i e j.
  - V_i: Capacidade máxima do depósito i.
  - d_j: Demanda do cliente j.
  - Q_k: Capacidade do veículo k.
  - P: Penalidade associada à alocação de um veículo.
- **Variáveis de Decisão:**
  - x_{ijk}: Binária e indica se o veículo k percorre diretamente do ponto i para o ponto j.
  - z_{ij}: Binária e indica se o cliente j é alocado ao depósito i.
  - U_{lk}: Variável auxiliar utilizada para eliminar subciclos.

– y_{ki}: Binária e indica se o veículo k está associado ao depósito i.

$$x_{ijk}, z_{ij}, y_{ki} \in \{0,1\}, \qquad \forall i,j \in J, \forall k \in K \tag{1}$$

$$u_{lk} \geq 0, \qquad \forall l \in J, \forall k \in K \tag{2}$$

2.1 Função Objetivo

O objetivo do modelo é minimizar o custo total das rotas percorridas por todos os veículos, levando em consideração a penalidade aplicada para cada veículo utilizado.

$$\text{Minimizar} \quad \sum_{k\in K}\sum_{i\in I}\sum_{j\in J} C_{ij} \cdot x_{ijk} + \sum_{k\in K}\sum_{i\in I} P \cdot y_{ki} \tag{3}$$

Neste contexto, o termo $C_{ij} \cdot x_{ijk}$ representa o custo de percorrer o arco (i,j) na rota, onde x_{ijk} é uma variável binária que indica se o arco (i,j) é utilizado.

2.2 Restrições

O modelo considera as seguintes restrições:

- **Atribuição de Rota:**

$$\sum_{k\in K}\sum_{i\in I\cup J} x_{ijk} = 1, \quad \forall j \in J \tag{4}$$

  Essa restrição assegura que cada cliente seja atendido exatamente uma vez, ou seja, cada cliente é atribuído a uma única rota.

- **Capacidade do Veículo:**

$$\sum_{j\in J}\sum_{i\in I\cup J} d_j \cdot x_{ijk} \leq Q_k, \quad \forall k \in K \tag{5}$$

  Garante que a demanda total dos clientes alocados a um veículo não ultrapasse a capacidade deste.

- **Eliminação de subciclos:**

$$U_{lk} - U_{jk} + (|J| - 1) \cdot x_{ijk} \leq |J| - 2, \quad \forall i,j \in J, \forall k \in K \tag{6}$$

  Utilizada para assegurar que cada veículo complete seu percurso, partindo e retornando ao depósito, sem formar ciclos internos indesejados.

- **Conservação de Fluxo:**

$$\sum_{j\in I\cup J} x_{ijk} - \sum_{j\in I\cup J} x_{jik} = 0, \quad \forall i \in I, k \in K \tag{7}$$

  Essa restrição mantém o equilíbrio entre o número de veículos que chegam e partem de cada nó, garantindo a conservação do fluxo em clientes e depósitos.

- **Atendimento de Depósito:**

$$\sum_{i\in I} z_{ij} \leq 1, \quad \forall j \in J \tag{8}$$

  Assegura que cada rota inicie e termine em um depósito, respeitando as condições de operação dos veículos.

Devido ao fato de ser um problema NP-difícil, os métodos exatos não são adequados para obter soluções ótimas em todas as instâncias (Surekha and Sumathi (2011)). Resolver o MDVRP envolve tomar decisões em múltiplos níveis, como a alocação de veículos e clientes aos depósitos e a definição das rotas para o atendimento aos clientes, o que contribui para a complexidade do problema.

3 Solução proposta

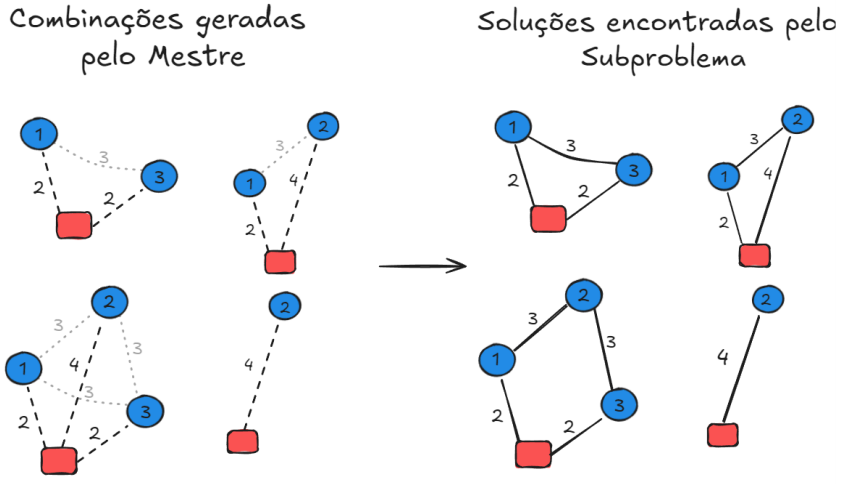
A decomposição de Benders é um método de otimização amplamente utilizado para resolver problemas complexos que envolvem variáveis complicadoras. Essas variáveis, quando fixadas temporariamente, simplificam o problema original, permitindo uma convergência mais rápida para a solução ótima (Rahmaniani et al. (2017)).

O método divide o problema em duas partes: o problema mestre (MP) e o subproblema. O MP aborda a parte mais difícil do problema e é resolvido iterativamente, enquanto o subproblema, de menor complexidade, gera cortes de factibilidade e/ou de otimalidade, que refinam progressivamente a solução do MP (Hooker (2023)).

Uma extensão do método tradicional é a decomposição de Benders baseada em lógica (LBBD), eficaz para problemas em que as variáveis complicadoras são melhor tratadas por modelos lógicos ou combinatórios (Hooker (2023)). Diferentemente da abordagem clássica, que utiliza programação linear para resolver os subproblemas, a LBBD permite que subproblemas sejam modelados e resolvidos por técnicas como programação inteira, programação por restrições ou outros métodos especializados (Hooker and Ottosson (2003)).

No contexto deste estudo, a LBBD é aplicada ao MDVRP, dividindo-o em um problema mestre que foca na alocação de veículos aos depósitos e subproblemas que se concentram na definição das menores rotas para atender os clientes. A Figura 3 ilustra a aplicação da decomposição para a solução apresentada pela Figura 2

Figura 3: Representação do Mestre e do Subproblema



Fonte: próprio autor

A seguir, são apresentadas as formulações matemáticas para o problema mestre e os subproblemas, detalhando como a LBBD estrutura e resolve esses componentes para alcançar uma solução ótima.

3.1 Descrição das Variáveis e Índices

- **Índices:**
  - $i, j \in C$: Índices dos clientes.
  - $k \in K$: Índice dos veículos.
  - $d \in D$: Índice dos depósitos.
  - n: Número de nós (clientes + depósito) no subproblema.
- **Parâmetros:**
  - demanda_i: Demanda do cliente i.
  - Q_d: Capacidade máxima do depósito d.
  - $\text{custo}_{i,j}$: Custo de viagem entre os nós i e j.
  - V: Número máximo de veículos disponíveis por depósito.
  - P: Penalidade associada à alocação de um veículo.
- **Variáveis de Decisão:**
  - $y_{k,d} \in \{0, 1\}$: Variável binária que indica se o veículo k está alocado ao depósito d.
  - $z_{i,k,d} \in \{0, 1\}$: Variável binária que indica se o cliente i é atendido pelo veículo k do depósito d.
  - $\alpha_{k,d} \geq 0$: Variável contínua associada ao custo total de uma rota específica para o veículo k do depósito d.
  - $x_{i,j} \in \{0, 1\}$: Variável binária que indica se o arco entre os nós i e j é percorrido.
  - u_i: Variável contínua usada para eliminação de subciclos no subproblema.

3.2 Problema Mestre

O problema mestre, descrito pela Equação 9, tem como objetivo minimizar o custo total da operação, composto por dois componentes principais: o custo associado à distância percorrida pelos veículos, representado por $\alpha_{k,d}$, e uma penalidade fixa P [2] para cada veículo k alocado ao depósito d.

Função Objetivo:

$$\text{Minimizar} \quad \sum_{k \in K} \sum_{d \in D} (\alpha_{k,d} \cdot y_{k,d} + P \cdot y_{k,d}) \tag{9}$$

Domínios das Variáveis:

$$y_{k,d} \in \{0, 1\}, \qquad \forall k \in K, \forall d \in D \tag{10}$$

$$z_{i,k,d} \in \{0, 1\}, \qquad \forall i \in C, \forall k \in K, \forall d \in D \tag{11}$$

$$\alpha_{k,d} \geq 0, \qquad \forall k \in K, \forall d \in D \tag{12}$$

As variáveis $y_{k,d}$ e $z_{i,k,d}$ são binárias, representando as decisões de alocação de veículos e atendimento de clientes, respectivamente. A variável $\alpha_{k,d}$ é contínua e não negativa, representando o custo associado à distância percorrida pelo veículo k para atender os clientes do depósito d.

[2] O valor para P foi definido como 1000 após uma análise inicial dos resultados.

3.2.1 Restrições do Problema Mestre

As restrições a seguir asseguram que cada cliente seja atendido, que as capacidades dos veículos e depósitos sejam respeitadas, e que a alocação de veículos ocorra de maneira eficiente.

- **Atendimento de clientes:**

$$\sum_{k \in K} \sum_{d \in D} z_{i,k,d} = 1, \quad \forall i \in C \tag{13}$$

  Cada cliente i deve ser atendido por exatamente um veículo k de algum depósito d.

- **Capacidade do veículo:**

$$\sum_{i \in C} \text{demanda}_i \cdot z_{i,k,d} \leq Q_d \cdot y_{k,d}, \quad \forall k \in K, \forall d \in D \tag{14}$$

  A carga total alocada ao veículo k no depósito d não pode exceder a capacidade Q_d do depósito.

- **Limite de veículos por depósito:**

$$\sum_{k \in K} y_{k,d} \leq V, \quad \forall d \in D \tag{15}$$

  O número de veículos alocados ao depósito d não pode exceder V.

- **Custo de atendimento:**

$$\alpha_{k,d} \geq \text{custo}_{i,d} \cdot z_{i,k,d}, \quad \forall i \in C, \forall k \in K, \forall d \in D \tag{16}$$

  A variável $\alpha_{k,d}$ deve ser maior ou igual ao custo associado ao atendimento do cliente i pelo veículo k no depósito d.

- **Vinculação de alocação:**

$$z_{i,k,d} \leq y_{k,d}, \quad \forall i \in C, \forall k \in K, \forall d \in D \tag{17}$$

  Um cliente só pode ser atendido por um veículo alocado ao depósito d se o veículo k estiver realmente alocado a esse depósito.

3.2.2 Pseudo-Código do Problema Mestre

O Algoritmo 1 ilustra o processo iterativo de solução do problema mestre. A cada iteração, são tomadas decisões sobre as variáveis de alocação e atendimento com base nos custos atuais, e essas decisões são refinadas ao longo das iterações, com a adição de cortes de Benders gerados pelos subproblemas.

3.3 Subproblema

O subproblema, representado pela Equação 18, tem como objetivo minimizar a distância percorrida pelos veículos designados a cada depósito para atender os clientes.

Função Objetivo:

$$\text{Minimizar} \quad \sum_{(i,j) \in A} \text{custo}_{i,j} \cdot x_{i,j} \tag{18}$$

O termo $\text{custo}_{i,j} \cdot x_{i,j}$ representa o custo associado ao arco (i, j) percorrido. Este subproblema corresponde a um problema de roteamento de veículos (VRP), com o objetivo de otimizar as rotas

Algorithm 1 Algoritmo para Resolver o Problema Mestre

Require: Conjunto de clientes C, veículos K, depósitos D

Require: Parâmetros de capacidade, matriz de custos, penalidade

Inicializar:

$GerarSolucaoInicial()$

$SolucaoAtual \leftarrow SolucaoInicial$

while Iteração < MaxIterações **e** Não Convergência **ou** Tempo limite de execução **do**

Resolver Subproblema:

for cada combinação de cliente, depósito, veículo **do**

$ResolverSubproblema(clientes, depósito, veículo)$

end for

Atualizar Solução do Problema Mestre:

$AtualizarSolucaoMestre()$

Verificar Convergência:

if SoluçãoAtual satisfaz condições de otimalidade **then**

$Parar$

end if

end while

de forma a minimizar os custos, respeitando as restrições de capacidade dos veículos. As soluções desses subproblemas geram cortes de Benders que refinam a solução do problema mestre.

Domínios das Variáveis:

$$x_{i,j} \in \{0,1\}, \qquad \forall (i,j) \in A \tag{19}$$

$$u_i \geq 0, \qquad \forall i \in N \tag{20}$$

A variável binária $x_{i,j}$ indica se o arco (i,j) é utilizado, enquanto u_i é uma variável contínua utilizada para eliminar subciclos no problema de roteamento.

3.3.1 Restrições do Subproblema

As restrições do subproblema asseguram que o percurso de cada veículo atenda a todos os clientes atribuídos, respeitando as capacidades dos veículos.

- **Visita Única por Veículo:**

$$\sum_{j \in N} x_{i,j} = 1, \quad \forall i \in N \tag{21}$$

Cada nó (cliente ou depósito) deve ser visitado exatamente uma vez por um veículo.

- **Saída Única por Veículo:**

$$\sum_{i \in N} x_{i,j} = 1, \quad \forall j \in N \tag{22}$$

Cada nó deve ser deixado exatamente uma vez por um veículo.

- **Eliminação de Subciclos:**

$$u_i - u_j + (n-1) \cdot x_{i,j} + (n-3) \cdot x_{j,i} \leq n-2, \quad \forall (i,j) \in A, i \neq j \tag{23}$$

Garantir a eliminação de subciclos nas rotas.

- **Direção Única dos Arcos:**

$$x_{i,j} + x_{j,i} \leq 1, \quad \forall (i,j) \in A \tag{24}$$

Cada arco entre dois nós i e j pode ser percorrido em apenas uma direção.

3.3.2 Pseudo-Código do Subproblema

O subproblema, modelado como um problema de roteamento de veículos, determina as rotas ótimas que cada veículo deve seguir a partir do depósito ao qual foi alocado no problema mestre. O Algoritmo 2 descreve o processo de solução do subproblema, considerando as restrições de capacidade dos veículos e as demandas dos clientes. As soluções obtidas no subproblema geram cortes de Benders, que são incorporados ao problema mestre para melhorar a solução global.

Algorithm 2 Algoritmo para Resolver o Subproblema

Require: Conjunto de clientes C, veículo K, depósito D
Require: Matriz de custos, demanda, capacidade
  Inicializar:
  $GerarSolucaoInicialSubproblema()$
  $SolucaoAtual \leftarrow SolucaoInicialSubproblema$
  $Mudanca \leftarrow \infty$
  for cada combinação de clientes **do**
    $ResolverTSP(SolucaoAtual)$
    if SolucaoAtual é Viável **then**
      $GerarCortesDeOtimalidade()$
    end if
    $Mudanca \leftarrow AtualizarSolucaoSubproblema()$
  end for

3.4 Cortes de Benders

Existem dois principais tipos de cortes: os cortes de factibilidade, que eliminam regiões inviáveis do espaço de soluções quando o subproblema indica inviabilidade na solução atual do modelo mestre, e os cortes de otimalidade, que ajustam a função objetivo do MP quando a solução é viável, mas ainda não ótima (Rahmaniani et al. (2017)).

No modelo apresentado, os cortes de otimalidade são utilizados para refinar a estimativa de custo total fornecida pelo problema mestre. Eles são gerados com base nas informações das soluções do subproblema e ajustam a função objetivo do MP para aproximá-la da solução ótima (Hooker (2023)). O corte de otimalidade implementado neste trabalho assegura que a função objetivo do problema mestre reflita corretamente os custos identificados no subproblema.

- **Corte de Otimalidade:**

$$\alpha_{k,d} \geq \text{objetivo}_{\text{tsp}} - \text{objetivo}_{\text{tsp}} \cdot \sum_{i \in C} (1 - z_{i,k,d}), \quad \forall k \in K, \forall d \in D \tag{25}$$

Esse corte é adicionado iterativamente ao problema mestre até que a solução ótima seja alcançada.

3.4.1 Pseudo-Código para Geração e Adição de Cortes de Benders

O processo de interação entre o problema mestre e os subproblemas, focando na geração e adição de cortes de otimalidade, é ilustrado pelo pseudo-código 3. Após a solução preliminar do problema mestre, todos os subproblemas são resolvidos, e os cortes de otimalidade gerados são incorporados ao problema mestre para refinar a solução.

Com a formulação do problema mestre, dos subproblemas e a definição dos cortes de Benders estabelecida, a próxima seção consiste na implementação da solução.

Algorithm 3 Algoritmo para Gerar e Adicionar Cortes de Otimalidade de Benders

Require: Solução do Subproblema (TSP), Parâmetros do Problema Mestre

Gerar Corte de Otimalidade:

$CorteOtimalidade \leftarrow GerarCorteDeOtimalidade(SolucaoTSP)$ ▷ Gera o corte de otimalidade com base na solução do TSP

Adicionar Corte ao Problema Mestre:

$AdicionarCorteAoMestre(CorteOtimalidade)$ ▷ Incorpora o corte gerado ao problema mestre

4 Experimentos computacionais

Os modelos para a decomposição lógica de Benders e para a programação inteira mista foram implementados utilizando o *solver* Gurobi, versão 11.0.3, com licença acadêmica, na linguagem de programação Python, versão 3.9.0. O ambiente computacional utilizado é um notebook com sistema operacional Windows 11 de 64 bits, equipado com uma CPU AMD Ryzen 5 7535HS (3,30 GHz) e 8 GB de memória RAM.

Para avaliar o desempenho da implementação, foram elaboradas 15 instâncias de teste, classificadas como pequenas, médias e grandes, com 5 instâncias em cada categoria. A geração dessas instâncias foi baseada em um conjunto original disponível no site *neo.lcc* [3].

Das informações presentes nos arquivos originais, foram utilizadas as seguintes: número de veículos, número de clientes, número de depósitos, capacidade máxima de um veículo, identificação dos clientes, coordenadas x e y, e demanda de cada cliente. Vale ressaltar a matriz de custo foi baseada na distância euclidiana, calculada a partir das coordenadas dos clientes e depósitos.

Foram selecionadas as 5 primeiras instâncias do conjunto, identificadas como *p01*, *p02*, *p03*, *p04* e *p05*. Para cada uma dessas instâncias originais, foram geradas 3 novas instâncias, correspondentes a cada uma das classificações (pequena, média e grande).

A escolha dos parâmetros para cada classificação foi baseada em uma análise das instâncias originais, levando em conta o número de depósitos, veículos e clientes. Os parâmetros adotados, apresentados na Tabela 2, foram definidos considerando a complexidade do MDVRP, que é um problema NP-difícil, o que implica em um aumento exponencial do tempo de resolução à medida que o número de variáveis cresce. Vale destacar que os parâmetros foram extraídos da primeira linha de cada instância, onde essas configurações são especificadas.

Tabela 2: Configurações das Instâncias Utilizadas

Instância	**Veículos**	**Clientes**	**Depósitos**	**Capacidade**
Pequena	4	15	2	80
Média	4	25	4	80
Grande	4	50	4	80

Com as configurações definidas, iniciou-se o processo de geração das novas instâncias. A partir de cada instância original, foram extraídos subconjuntos de clientes e depósitos conforme os parâmetros estabelecidos para cada classificação. Por exemplo, a instância *p01* original possui 4 veículos, 50 clientes e 4 depósitos; para gerar a versão pequena, foram selecionados os 15 primeiros clientes e os 2 primeiros depósitos, enquanto as informações restantes foram removidas. Repetiu-se esse processo para todas as classificações das 5 instâncias selecionadas.

Todas as instâncias geradas, bem como as instâncias originais e a implementação dos modelos, estão disponíveis no repositório *GitHub* [4].

[3] Disponível em: https://neo.lcc.uma.es/vrp/vrp-instances/multiple-depot-vrp-instances/, Acesso em: 01/09/2024.

[4] Disponível em: https://github.com/JoseBieco/MDVRP-LBBD, Acesso em: 01/09/2024.

Com as instâncias devidamente geradas e os modelos implementados, foram realizados testes comparativos para avaliar o desempenho das abordagens de decomposição de Benders e de programação inteira mista (MIP). Cada instância foi submetida a ambos os modelos, com um tempo máximo de execução de 30 minutos para cada teste. As tabelas 3 e 4 apresentam os resultados obtidos para cada modelo.

Tabela 3: Resultados Obtidos com a Decomposição de Benders

Instância	**Classe**	**Objetivo**	**Tempo (s)**	**Dist. Percorrida**	**Veículos**	**Soma das Cap.**
p01	Pequena	4200,44	22,52	483,99	4	258
p01	Média	6444,06	1800,28	585,36	6	424
p01	Grande	11206,67	1800,23	1458,04	10	777
p02	Pequena	4168,36	2,13	440,74	4	258
p02	Média	6217,1	450,38	548,1	6	424
p02	Grande	11165,95	1800,25	1366,59	10	777
p03	Pequena	4190,22	2,61	426,29	4	315
p03	Média	7394,15	1800,37	569,07	7	491
p03	Grande	14424,61	1800,48	1527,55	13	974
p04	Pequena	3202,8	4,32	452,06	3	206
p04	Média	5489,22	1801,62	748,95	5	332
p04	Grande	11260,46	1801,4	1727,81	10	721
p05	Pequena	3167,08	2,61	405,53	3	206
p05	Média	5365,47	1800,33	804,65	5	332
p05	Grande	11390,51	1803,37	1887,9	10	721

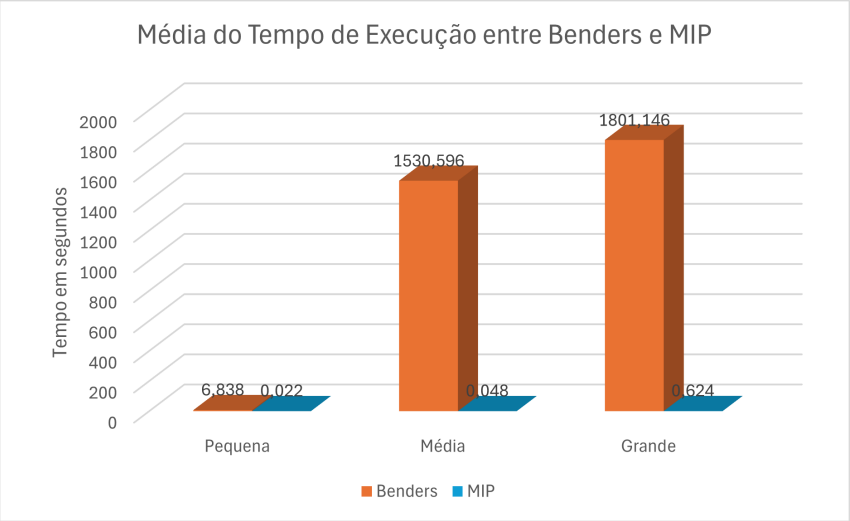
Tabela 4: Resultados Obtidos com o MIP

Instância	**Classe**	**Objetivo**	**Tempo (s)**	**Dist. Percorrida**	**Veículos**	**Soma das Cap.**
p01	Pequena	4751,74	0,03	427,67	4	258
p01	Média	6981,33	0,06	1002,55	6	424
p01	Grande	11496,32	0,62	1876,21	10	777
p02	Pequena	4751,74	0,02	427,67	4	258
p02	Média	6981,33	0,05	1002,55	6	424
p02	Grande	11496,32	0,41	1876,21	10	777
p03	Pequena	4696,49	0,02	524,98	4	315
p03	Média	8035,2	0,04	1068,06	7	491
p03	Grande	14395,54	0,28	1843,49	13	973
p04	Pequena	3735,18	0,02	488,8	3	206
p04	Média	6132,14	0,05	929,56	5	332
p04	Grande	11627,28	0,87	1946,78	10	721
p05	Pequena	3707,36	0,02	540,5	3	206
p05	Média	6045,38	0,04	863,39	5	332
p05	Grande	11640,04	0,94	2057,31	10	721

Durante a realização dos experimentos computacionais, foram adotadas várias métricas para comparar o desempenho dos modelos implementados, observadas nas Tabelas 3 e 4. As principais métricas consideradas foram:

- **Tempo Computacional**: Medido em segundos, representa o tempo total necessário para que cada modelo encontre uma solução, ou até que atinja o tempo limite de 30 minutos.
- **Qualidade da Solução**: Avaliada pelo valor da função objetivo, que neste caso específico

Figura 4: Média do Tempo de Execução entre Benders e MIP



Fonte: próprio autor

corresponde ao custo total da operação. Menores valores da função objetivo indicam soluções de melhor qualidade, uma vez que o problema é de minimização.

Essas métricas foram escolhidas para considerar tanto a eficiência computacional quanto a eficácia na minimização dos custos totais.

4.1 Análise dos Resultados

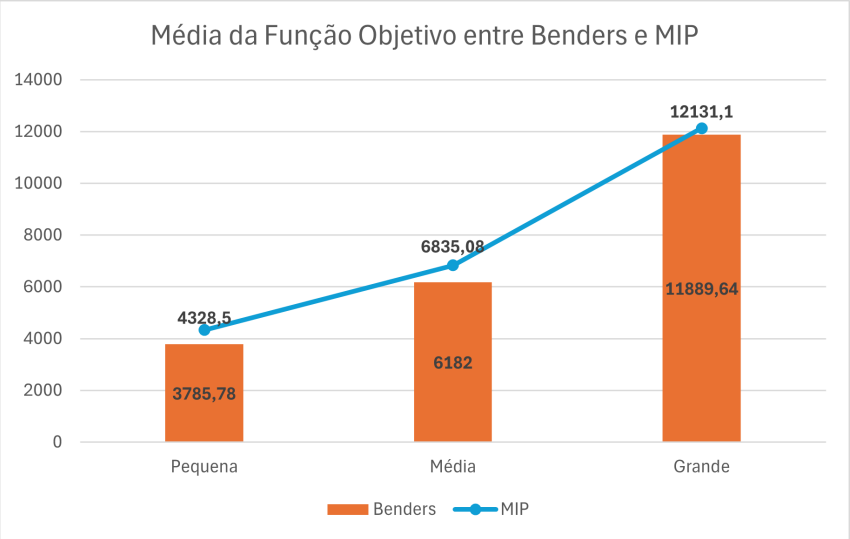
O modelo MIP demonstrou um tempo de execução significativamente menor em comparação com a decomposição de Benders, com a maioria das instâncias sendo resolvidas em poucos segundos. Em contraste, o tempo de execução da decomposição de Benders frequentemente atingiu o limite de 30 minutos (1800 segundos) em várias instâncias médias e grandes, evidenciando uma maior dificuldade em alcançar a convergência dentro do tempo disponível. O Gráfico 4 ilustra as médias de diferença de tempo de execução entre os dois métodos, destacando a vantagem do MIP em termos de velocidade.

Em termos de função objetivo, a decomposição de Benders apresentou soluções com valores menores em relação ao MIP. O Gráfico 5 apresenta a comparação das médias dos valores de função objetivo entre Benders e MIP.

Na decomposição de Benders, para as instâncias pequenas, o menor número de veículos permitiu uma otimização eficiente das rotas, resultando em menores valores da função objetivo em comparação ao MIP. O método se beneficiou do menor número de variáveis, o que facilitou a convergência para uma solução próxima do ótimo. No entanto, à medida que o número de veículos aumentou, a complexidade do problema também cresceu, fazendo com que o tempo necessário para alcançar uma solução de alta qualidade aumentasse significativamente. Em várias instâncias médias e grandes, o Benders atingiu o limite de tempo permitido (30 minutos) sem atingir a convergência total, apresentando pequenos gaps. Mesmo assim, a decomposição de Benders conseguiu obter soluções com menores custos totais, evidenciando sua eficácia em minimizar a função objetivo mesmo sob maior complexidade.

Em contraste, o modelo MIP obteve soluções rapidamente nas instâncias pequenas, mas com valores de função objetivo superiores aos do Benders, indicando que, embora eficiente em tempo, não foi tão eficaz em minimizar custos. Com o aumento do número de veículos nas instâncias médias e grandes, o MIP manteve sua rapidez, mas os valores da função objetivo também aumentaram,

Figura 5: Média da Função Objetivo entre Benders e MIP



Fonte: próprio autor

sugerindo uma menor qualidade das soluções em problemas mais complexos.

5 Conclusão e trabalhos futuros

Após a análise dos resultados, constatou-se que a decomposição de Benders demonstrou um desempenho superior em termos de qualidade da solução, especialmente em cenários de alta complexidade. Embora o modelo de programação inteira mista (MIP) tenha se destacado por sua rapidez na obtenção de soluções, ele não se mostrou tão eficaz quanto o Benders na minimização dos custos totais. Esses resultados sugerem que, em contextos onde a qualidade da solução é prioritária, a decomposição de Benders é a abordagem mais adequada, mesmo que isso implique em tempos de execução mais longos.

Para trabalhos futuros, recomenda-se explorar a aplicação de heurísticas para gerar soluções iniciais, o que pode potencialmente acelerar a convergência da decomposição de Benders, reduzindo o tempo de execução sem comprometer a qualidade da solução. Além disso, a investigação de novos cortes que possam ser integrados ao modelo pode contribuir para melhorar ainda mais a eficiência do método, tornando-o mais robusto em situações de alta complexidade. A análise de outras variantes ou implementação de técnicas híbridas, também representa uma oportunidade promissora para avançar na solução de problemas de otimização de grande escala.

Referências

Margarita P Castro, Merve Bodur, and Amer Shalaby. Incorporating service reliability in multi-depot vehicle scheduling. *arXiv preprint arXiv:2407.00836*, 2024.

Claudio Contardo and Rafael Martinelli. A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization*, 12:129–146, 2014.

John Hooker. *Logic-based benders decomposition: theory and applications*. Springer Nature, 2023.

John N Hooker and Greger Ottosson. Logic-based benders decomposition. *Mathematical Programming*, 96(1):33–60, 2003.

DGND Jayarathna, GHJ Lanel, and ZAMS Juman. Survey on ten years of multi-depot vehicle routing problems: mathematical models, solution methods and real-life applications. *Ideas Spread*, 2021. doi: https://doi.org/10.30560/sdr.v3n1p36.

Nafiz Mahmud and Md Mokammel Haque. Solving multiple depot vehicle routing problem (mdvrp) using genetic algorithm. In *2019 International conference on electrical, computer and communication engineering (ECCE)*, pages 1–6. IEEE, 2019.

Ragheb Rahmaniani, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259 (3):801–817, 2017.

Paneerselvam Surekha and Sai Sumathi. Solution to multi-depot vehicle routing problem using genetic algorithms. *World Applied Programming*, 1(3):118–131, 2011.