UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE GRADUAÇÃO EM CIÊNCIA DE COMPUTAÇÃO

EDUARDO COSTA PEREIRA

COMPARAÇÃO DO DESEMPENHO DE PACOTES COMPUTACIONAIS NA SOLUÇÃO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM MÚLTIPLOS DEPÓSITOS

Florianópolis
2023

EDUARDO COSTA PEREIRA

COMPARAÇÃO DO DESEMPENHO DE PACOTES COMPUTACIONAIS NA SOLUÇÃO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM MÚLTIPLOS DEPÓSITOS

Trabalho de Conclusão de Curso submetido ao Curso de Bacharelado em Ciência da Computação do Campus Florianópolis da Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de Bacharel em Ciências da Computação.

Orientador(a): Prof. Dr. Pedro Castellucci

Florianópolis
2024

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Pereira, Eduardo Costa
Comparação do desempenho de pacotes computacionais na solução do Problema de Roteamento de Veículos com Múltiplos Depósitos / Eduardo Costa Pereira ; orientador, Pedro Belin Castellucci, 2024.
55 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2024.

Inclui referências.

1. Ciências da Computação. 2. Otimização. 3. Roteamento de Veículos. 4. Pacotes Computacionais. I. Castellucci, Pedro Belin. II. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. III. Título.

Eduardo Costa Pereira

Comparação do desempenho de pacotes computacionais na solução do Problema de Roteamento de Veículos com Múltiplos Depósitos

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de Bacharel e aprovado em sua forma final pelo Curso de Graduação em Ciência da Computação

Florianópolis, 3 de dezembro de 2024

_____

Coordenação do Curso de Graduação em Ciência da Computação

Banca examinadora

_____

Prof. Pedro Belin Castellucci, Dr.

Orientador

Universidade Federal de Santa Catarina

_____

Prof. Álvaro Junio Pereira Franco, Dr.

Avaliador

Universidade Federal de Santa Catarina

_____

Prof. Rafael de Santiago, Dr.

Avaliador

Universidade Federal de Santa Catarina

Florianópolis, 2024

AGRADECIMENTOS

Gostaria de agradecer à minha mãe, Cláudia, que me apoiou durante toda a graduação; ao meu pai, Amarildo, que foi essencial na minha trajetória; à minha irmã, Ana Paula, que sempre esteve ao meu lado quando precisei. Agradeço também ao meu orientador, Pedro, pelos valiosos ensinamentos e orientação ao longo deste percurso. Além disso, tanto eu quanto meu orientador expressamos nosso reconhecimento ao apoio do CNPq (405247/2023-0).

RESUMO

Hoje em dia, os sistemas logísticos possuem grande complexidade, precisando lidar com centenas e até milhares de variáveis incluindo clientes, mercadorias, rotas, motoristas, entre outras. Nesse cenário, os computadores, que atualmente são capazes de testar milhões de cenários em um tempo relativamente curto, são excelentes para auxílio na tomada de decisão e podem ajudar a economizar porcentagens significativas dos custos com transporte. A classe de problemas de Roteamento de Veículos (*Vehicle Routing Problems*, VRPs) possui diversos problemas importantes para a área de logística, incluindo variações do Problema de Roteamento de Veículos (*Vehicle Routing Problem*, VRP), que introduzem janelas de tempo, entrega e recebimento de mercadorias, etc. Este trabalho estuda a variação que introduz múltiplos depósitos, o Problema de Roteamento de Veículos com Múltiplos Depósitos (*Multi-Depot Vehicle Routing Problem*, MDVRP), que possui importância econômica documentada. Mais especificamente, estuda o desempenho de pacotes computacionais de Programação Linear Inteira Mista, comerciais e gratuitos, na solução dessa variante. O resultado desse trabalho é a comparação desses pacotes, utilizando as métricas de melhor solução encontrada, melhor limitante da raiz, melhor limitante inferior e tempo de solução.

Palavras-chave: Otimização. Programação Linear Inteira Mista. Roteamento de Veículos. Pacotes Computacionais.

ABSTRACT

Nowadays, logistics systems exhibit significant complexity, needing to deal with hundreds to even thousands of variables, including customers, goods, routes, drivers, among others. In this scenario, computers, which are currently capable of testing millions of scenarios within a relatively short time, prove to be excellent aids in decision-making and can assist in saving substantial percentages of transportation costs. The class of Vehicle Routing Problems (VRPs) encompasses several critical challenges within the logistics field, including variations of the Vehicle Routing Problem (VRP) that introduce elements like time windows, goods delivery and receipt, and more. This study focuses on the variation that introduces multiple depots, known as the Multi-Depot Vehicle Routing Problem (MDVRP), which holds documented economic importance. More specifically, it examines the performance of mixed-integer linear programming computational packages, both commercial and free, in solving this variant. The outcome of this research is the comparison of these packages, using the metrics of best solution found, root bound, lower bound and solution time.

Keywords: Optimization. Mixed-Integer Linear Programming. Vehicle Routing. Computational Packages.

SUMÁRIO

1 INTRODUÇÃO

A ampliação da complexidade da economia global - que ocorreu nos últimos séculos - naturalmente levou a um aumento na complexidade dos sistemas logísticos, e a necessidade de gerenciar um número maior de mercadorias, motoristas, veículos, clientes, rotas, entre outras variáveis. Nesse contexto, um dos problemas fundamentais da área logística tornou-se o da distribuição física, que consiste basicamente em levar mercadorias de depósitos para clientes, por meio da utilização de veículos que se movem por uma determinada malha de estradas, revendo estratégias de distribuição e canais utilizados para garantir, de modo satisfatório, a disponibilidade dos produtos aos consumidores.

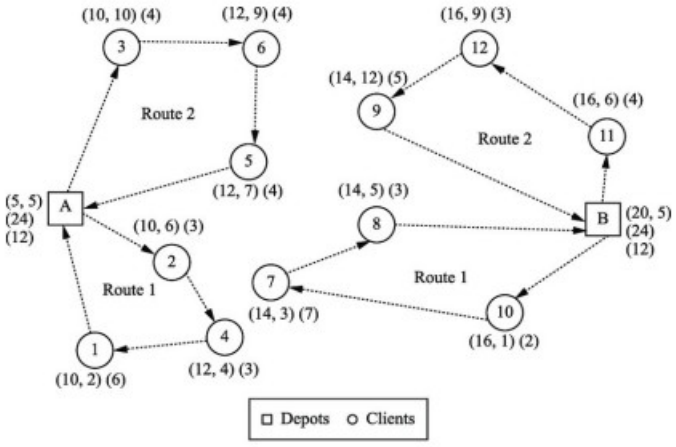
A literatura de Pesquisa Operacional define essa classe de problemas como Roteamento de Veículos (*Vehicle Routing Problems*, VRPs) e a utilização de métodos computacionais para a sua solução já se mostrou capaz de economizar entre 5% a 20% dos custos globais de transporte (TOTH, 2002). O problema fundamental dessa classe é o Problema de Roteamento de Veículos Capacitados (*Capacitated Vehicle Routing Problem*, CVRP), que consiste em um único depósito, uma frota de veículos com capacidade conhecida, clientes com demanda conhecida e um custo associado às arestas do grafo da malha de estradas e que tem por objetivo determinar um conjunto de rotas que cumpram suas restrições (por exemplo, em uma determinada rota, a soma das demandas dos clientes não pode ultrapassar a capacidade do veículo) ao mesmo tempo que tem a sua função custo minimizada.

O poder expressivo do CVRP não é suficiente para abarcar muitas das situações do mundo real, o que levou ao desenvolvimento de variações do problema que introduzem novas complexidades, como janelas de tempo, a possibilidade de entregar e receber mercadorias, a presença de múltiplos depósitos, entre outras. Dentre essas, a variação que introduz múltiplos depósitos ao problema, conhecida como Roteamento de Veículos com Múltiplos Depósitos (*Multi-Depot Vehicle Routing Problem*, MDVRP), possui extensa aplicação no mundo real já documentada, incluindo distribuição de produtos do petróleo (BROWN et al., 1987), entrega de refeições (CASSIDY; BENNETT, 1972), gases industriais (BELL et al., 1983) e reparo de desastres (VIEIRA et al., 2021). Na Figura 1, pode-se ver uma

instância do MDVRP que consiste em 2 depósitos (A, B), 12 clientes e 4 rotas. O MDVRP é o foco de estudo deste trabalho.

A solução do MDVRP é normalmente feita por um de dois caminhos: o primeiro, conhecido como métodos exatos, visa obter a solução ótima para o problema, enquanto o segundo, conhecido como métodos heurísticos, visa obter uma boa solução em um tempo razoável. Para a solução por métodos exatos existem no mercado várias opções de pacotes computacionais, os quais implementam diferentes técnicas para se chegar à melhor solução o mais rápido possível. A variedade de técnicas e a complexa interação entre elas normalmente exige a avaliação empírica do desempenho (GLEIXNER, 2021). Este trabalho tem como objetivo avaliar o desempenho desses pacotes computacionais na solução do MDVRP, visando fornecer informações úteis à tomada de decisões em empresas e instituições que precisam escolher entre um deles.

Figura 1 - Exemplo do MDVRP



Fonte: (DA SILVA JÚNIOR et al., 2011)

1.1 OBJETIVO GERAL

O objetivo geral deste trabalho é avaliar pacotes computacionais baseados em branch-and-cut para a resolução do problema de roteamento de veículos com múltiplos depósitos.

1.2 OBJETIVOS ESPECÍFICOS

• Estudar os algoritmos básicos de programação inteira, utilizados pelos pacotes computacionais.

• Estudar um modelo de programação linear inteira-mista do problema de roteamento de veículos com múltiplos depósitos.

• Implementar o modelo para resolução com pacotes computacionais comerciais e gratuitos.

• Comparar os pacotes utilizando instâncias definidas na literatura.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, primeiramente são abordados os principais conceitos de otimização discreta (2.1), posteriormente são explicados métodos exatos usados para resolução de problemas de programação linear inteira, a enumeração completa (2.2), o *branch-and-bound* (2.3), planos de corte (2.4) e a combinação dos dois anteriores, o *branch-and-cut*. A seção termina com uma análise sobre o Problema de Roteamento de Veículos (2.5).

2.1 OTIMIZAÇÃO DISCRETA

Antes de abordar diretamente o conceito de otimização discreta, é importante tratar do assunto de modelagem matemática. A modelagem matemática é muito usada nas ciências da natureza, em que as leis que regem os fenômenos naturais são observadas e, se possível, transcritas para a linguagem matemática. Esse modelo criado não tem por objetivo representar o fenômeno exatamente como ele é na realidade, mas sim simplificá-lo a alguns elementos essenciais e relacionar estes por meio de expressões matemáticas, que, por mais sucintas que sejam, podem ser usadas para resolver problemas e fazer previsões. Na Física, por exemplo, a modelagem matemática ocorre no estudo do lançamento oblíquo, que consiste em determinar a trajetória de um objeto lançado com uma certa velocidade, formando um ângulo diferente de zero com a horizontal, sob ação exclusiva da força da gravidade. A partir desses parâmetros, é possível chegar a um conjunto de equações que descrevem como a partícula se comporta até atingir o solo.

A área de Pesquisa Operacional se utiliza disso para resolver problemas práticos de logística, de aviação e de outras indústrias. A ideia é estudar o problema e abstrair variáveis e relações matemáticas, usar poder computacional e algoritmos para encontrar soluções para o modelo criado e, então, interpretar o que essas soluções dizem sobre o problema real (ARENALES et al., 2015). Alguns tipos de modelos comumente usados para representar problemas na área de Pesquisa Operacional incluem: programação linear, programação linear inteira mista

(otimização discreta), programação não-linear e programação em redes (ARENALES et al., 2015).

O conceito de otimização discreta está muito relacionado ao de otimização linear. A otimização linear utiliza um tipo de modelo matemático que serve para representar uma grande quantidade de problemas práticos. Formalmente, um problema desse tipo pode sempre ser escrito na seguinte forma padrão (ARENALES et al., 2015):

$$min\ f(x_1, x_2, \ldots, x_n) = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n \quad (1)$$

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \quad (2)$$

$$\vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m$$

$$x_1 \geqslant 0,\ x_2 \geqslant 0,\ \ldots,\ x_n \geqslant 0 \quad (3)$$

Nessa estrutura, a função linear que ocorre em (1), a ser minimizada, é denominada função objetivo, o sistema de equações lineares em (2) determina as restrições do problema e as inequações (3) são chamadas de condições de não-negatividade e impedem que as variáveis assumam valores negativos.

Algumas definições importantes decorrem dessa definição formal, especificamente os conceitos de solução factível, região factível e solução ótima. Uma solução factível é qualquer tupla $(x_1,\ x_2,\ \ldots,\ x_n)$ que satisfaz tanto as restrições do problema quanto às condições de não-negatividade. Uma região factível é definida como o conjunto de todas as soluções factíveis. Uma solução factível $(x_1^*,\ x_2^*,\ \ldots,\ x_n^*)$ é ótima, para um problema de minimização, se $f(x_1^*,\ x_2^*,\ \ldots,\ x_n^*) \leqslant f(x_1,\ x_2,\ \ldots,\ x_n)$ para qualquer solução factível $(x_1,\ x_2,\ \ldots,\ x_n)$.

A seguir é mostrado um exemplo de um modelo desse tipo:

$$min\quad f(x, y) = -y \quad (4)$$

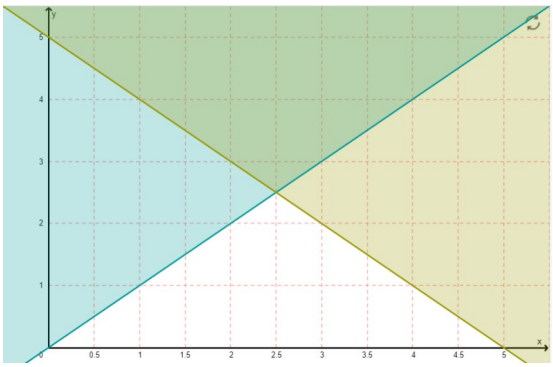
$$-x + y \leqslant 0 \quad (5)$$

$$x + y \leqslant 5 \quad (6)$$

$$x \geqslant 0,\ y \geqslant 0 \quad (7)$$

O problema possui as variáveis x e y, duas restrições e as condições de não-negatividade. Como o problema possui apenas duas variáveis, todas as soluções estarão contidas em $\mathbb{R}^2$, ou seja, no plano. Isso permite que o problema seja visualizado de maneira geométrica, como pode ser visto na Figura 2:

Figura 2 - Problema de programação linear



Fonte: Elaborada pelo autor

Na Figura 2, a parte pintada de azul representa os pontos que não satisfazem a restrição (5), enquanto a parte amarela representa os pontos que não satisfazem a restrição (6). A parte branca acima do eixo x representa a região factível, ou o conjunto de tuplas que satisfazem as duas restrições e a condição de não-negatividade. Além disso, o gradiente da função objetivo dita que a função cresce mais rapidamente na direção do vetor (0, -1), ou, analogamente, que diminui mais rapidamente na direção do vetor (0, 1). Como o objetivo é minimizar a função, o ponto mais “alto” da região factível na direção (0, 1) é aquele que soluciona o problema. Não é difícil perceber que esse ponto é o vértice (2.5, 2.5), com a função objetivo assumindo o valor de -2.5.

A otimização discreta é semelhante à otimização linear, exceto pelo fato que as variáveis não pertencem necessariamente ao conjunto dos reais, podendo pertencer a conjuntos de valores inteiros ou binários, por exemplo. Dependendo dos conjuntos onde as variáveis assumem seus valores, o problema pode ser classificado em algumas classes. Destacam-se as classes de programação linear inteira mista, programação linear inteira e programação binária. Um problema é dito ser de programação linear inteira mista se puder ser escrito no formato (ARENALES et al., 2015):

$$z = max \quad cx + dy \tag{8}$$

$$Ax + Dy \leqslant b \tag{9}$$

$$x \in \mathbb{R}_+^n, y \in \mathbb{Z}_+^p \quad (10)$$

Nesse problema, c, um vetor $(1\times n)$ e d, um vetor $(1\times p)$ definem a função objetivo, com c se aplicando às variáveis reais e d às variáveis inteiras. A, uma matriz $(m\times n)$, define os coeficientes das variáveis reais nas restrições e D, uma matriz $(m\times p)$, define os coeficientes das variáveis inteiras nas restrições. O termo independente nas inequações é representado pelo vetor b $(m\times 1)$. As variáveis x e y pertencem ao espaço de vetores com n componentes reais e ao espaço dos vetores com p componentes inteiras, respectivamente.

O exemplo usado anteriormente pode ser alterado para representar um problema de programação linear inteira mista

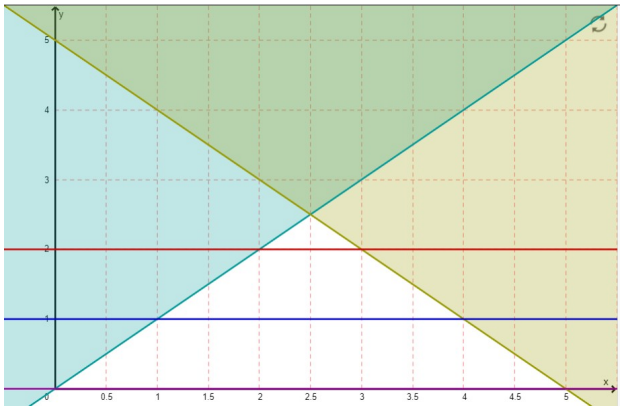
$$min \quad f(x,y) = -y \quad (11)$$

$$-x + y \leqslant 0 \quad (12)$$

$$x + y \leqslant 5 \quad (13)$$

$$x \in \mathbb{R}_+, \; y \in \mathbb{Z}_+ \quad (14)$$

Figura 3 - Problema de programação linear inteira mista



Fonte: Elaborada pelo autor

Em que a região factível é representada pela interseção das retas $y = 0$, $y = 1$ e $y = 2$ com a região branca da Figura 3. Neste novo problema, a solução ótima é qualquer solução factível que pertença à reta vermelha ($y = 2$), com o valor da função objetivo sendo -2, menor que o problema de otimização linear. O valor ótimo ter aumentado não é de surpreender, já que o novo espaço de soluções está contido propriamente no anterior.

Os problemas de programação binária e inteira podem ser definidos de maneira semelhante. No primeiro caso, as variáveis assumem apenas valores binários e, no segundo, apenas valores inteiros.

Embora tenha sido fácil solucionar o exemplo fornecido, isso se deu pela sua simplicidade: poucas restrições e variáveis. A tarefa de solucionar um problema de otimização discreta pode ser difícil em muitos casos. Normalmente, dois caminhos principais são usados para solucionar um problema desse tipo, os métodos exatos e os heurísticos. Os primeiros garantem encontrar a solução ótima, mas em instâncias grandes podem exigir muitos recursos computacionais. Os segundos têm por objetivo apenas encontrar uma solução suficiente para determinada aplicação, portanto não garantem encontrar a melhor solução, mas em instâncias grandes costumam exigir menos tempo de processamento. Na próxima seção serão apresentados alguns métodos exatos e os métodos heurísticos não serão abordados devido ao escopo do trabalho. O leitor interessado em métodos heurísticos pode consultar o livro de Rothlauf (2011).

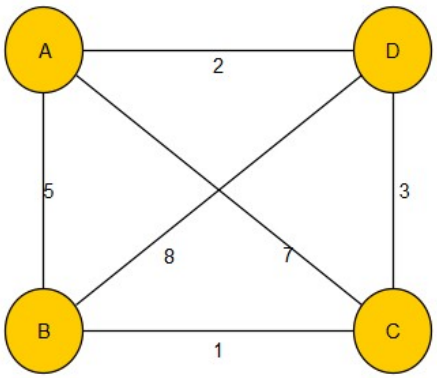
2.2 ENUMERAÇÃO COMPLETA

A enumeração completa é um método que pode ser usado para resolver problemas de otimização quando é possível chegar a um conjunto finito de soluções factíveis. Se o problema for de maximização, então esse método consiste em calcular o valor da função objetivo para todo o conjunto de soluções factíveis e escolher a(s) solução(ões) de maior valor. Se o problema for de minimização, escolhe-se a(s) de menor valor. Nessas condições, é evidente que esse método funciona e que trará a solução ótima, ainda que de maneira custosa, e, frequentemente, proibitiva.

A Figura 4 mostra uma instância do problema do caixeiro-viajante, com 4 cidades, A, B, C e D, que pode ser solucionada usando esse método. O problema do caixeiro viajante clássico consiste em um caixeiro que sai de uma cidade inicial, percorre todas as outras cidades passando por cada uma exatamente uma vez, e retorna à cidade inicial, em uma rota de distância mínima. De maneira mais formal, assume-se que exista um grafo completo $G = (N, E)$, em que N representa o

conjunto das cidades e E o conjunto das arestas do grafo, e que cada aresta tenha um peso associado C_{ij}, $C_{ij} = C_{ji}$, que é a distância entre as cidades i, j. Deve-se encontrar um ciclo hamiltoniano de distância mínima nesse grafo.

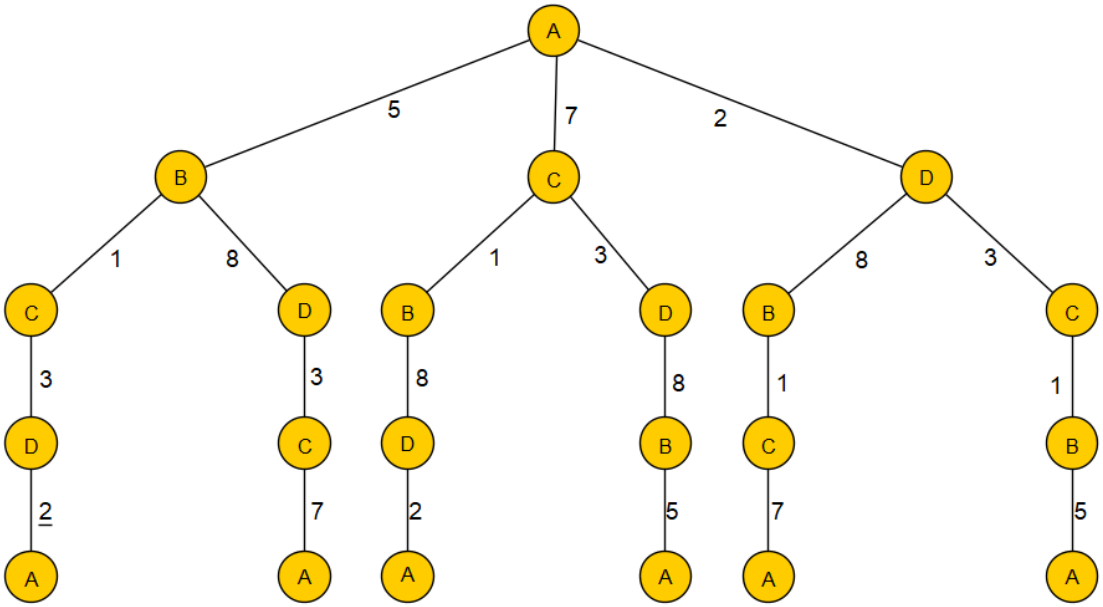
Figura 4 - Instância do caixeiro-viajante



Fonte: Elaborada pelo autor

A árvore da Figura 5 mostra as possíveis rotas para o problema anterior, considerando A como cidade inicial. Nesta árvore, os números nas arestas representam a distância percorrida entre as cidades e cada um dos caminhos da raiz às folhas representa uma rota diferente. Ao somar as arestas de cada desses caminhos é possível concluir que as rotas de distância mínima são (A -> B ->C -> D -> A) e (A -> D -> C -> B -> A), ambas de distância total 11.

Figura 5 - Rotas para a instância do caixeiro-viajante



Fonte: Elaborada pelo autor

Esse método é simples de entender e de implementar, mas é muito ineficiente. Por exemplo, para uma instância do problema do caixeiro-viajante com 20 cidades, seria necessário analisar 19! = 121.645.100.408.832.000 rotas, mesmo que um computador fosse capaz de analisar um bilhão de rotas por segundo, ainda demoraria cerca de 92 anos para resolver o problema. As próximas duas seções descrevem métodos mais eficientes, o *branch-and-bound* e o método de planos de corte.

2.3 BRANCH-AND-BOUND

O *branch-and-bound,* proposto por Land e Doig (1960), é um método de programação inteira e consiste, resumidamente, em dividir um problema de otimização em problemas menores, até que determinadas condições sejam atingidas, por meio da inclusão de restrições que eliminam as soluções ótimas dos problemas relaxados.

Como exemplo, imagine que um determinado problema de programação inteira P, de maximização, seja composto por três variáveis x_1, x_2, x_3 e que a sua

relaxação linear possua como solução ótima a tupla (1, 1.5, 3). O método *branch-and-bound*, nesse caso, dividirá esse problema em dois novos, por meio da inclusão das novas restrições $x_2 \leqslant \lfloor 1.5 \rfloor$ e $x_2 \geqslant \lceil 1.5 \rceil$, o que pode ser feito porque a variável x_2 está definida como inteira no problema original. Dessa forma, o problema original P se transformará nos problemas P_1, idêntico a P exceto pela inclusão da restrição $x_2 \leqslant 1$ e P_2, também idêntico a P, exceto pela inclusão de $x_2 \geqslant 2$. O mesmo procedimento pode ser repetido para os novos problemas, em um processo recursivo que gera uma árvore *branch-and-bound*.

Antes de um problema ser ramificado, alguns testes são executados, considerando sua relaxação linear, para verificar se o problema pode ser "eliminado". O problema P_i pode ser eliminado se passar em um dos seguintes testes, considerando a seguinte notação:

F(P_i) = região factível do problema P_i

F(PL_i) = região factível do problema relaxado PL_i.

z_i = valor ótimo do problema P_i, é um limitante inferior para o problema P.

$\bar{z}_i$ = valor ótimo para o problema PL_i.

x^* = melhor solução encontrada.

z^* = valor de x^*.

1. F(PL_i) = $\emptyset$: Como F(P_i) $\subseteq$ F(PL_i), não há solução factível para P_i.

2. $\bar{z}_i \leq z^*$: Como a melhor solução para PL_i é maior ou igual à melhor solução para P_i, qualquer solução de P_i será menor que a melhor solução encontrada até o momento, logo o problema *i* não precisa ser considerado.

3. A solução ótima de PL_i é inteira: Nesse caso a solução ótima para P_i será a mesma solução de PL_i.

Esse método pode ser especificado mais formalmente pelo algoritmo *branch-and-bound*. Neste algoritmo, extraído do livro do Arenales, mostrado a seguir em 7 passos, um nó ativo, armazenado na lista L, é um problema que ainda não foi eliminado e nem ramificado.

Passo 1) Faça $\bar{z} = \infty$, $z^* = -\infty$, $x^* = \emptyset$, $L = \{P\}$

Passo 2) Se a lista L estiver vazia, prossiga para o passo 6. Caso contrário, selecione um nó ativo i.

Passo 3) Se F(PL_i) for vazia, vá para o passo 2.

Passo 4) Se $z_i \leqslant z^*$, vá para o passo 2.

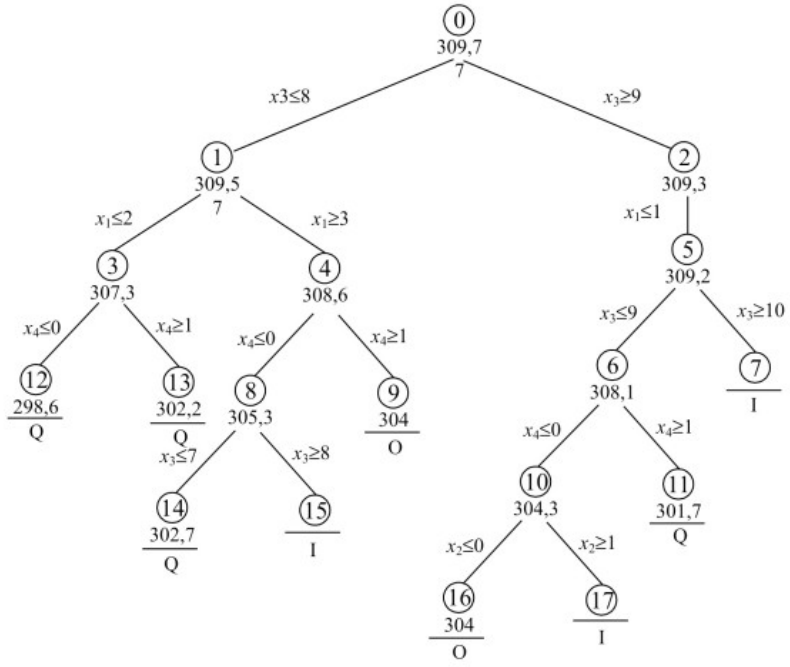
Passo 5) Se a solução ótima x_i para o problema PL_i é inteira e $z_i \geqslant z^*$, atualize x^* e z^*. Depois disso, elimine os nós ativos da lista L cuja solução ótima para o problema relaxado seja menor ou igual a z^* e volte ao passo 2.

Passo 6) Selecione uma variável não inteira da solução ótima de PL_i para dividir o nó i em dois novos problemas, adicione à lista L e vá para o passo 2.

Passo 7) Se $z^* \neq -\infty$ então a solução ótima é x^*. Senão, não existe solução factível.

A Figura 6 mostra um exemplo da árvore *branch-and-bound* após um problema ser resolvido com o uso do método. As folhas da árvore representam problemas que foram eliminados ou que deram origem à melhor solução encontrada.

Figura 6 - Exemplo de árvore *branch-and-bound*

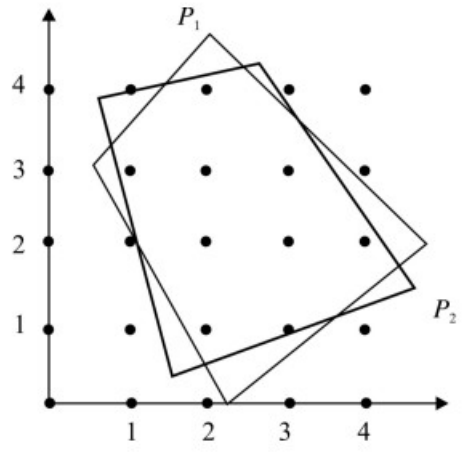


Fonte: (ARENALES et al., 2015)

2.4 PLANOS DE CORTE

Quando um problema de programação inteira é descrito matematicamente, é definido um conjunto de restrições que determinam o conjunto de soluções factíveis para o problema. Na imagem abaixo, as duas regiões, P_1 e P_2, definidas por um conjunto de restrições lineares, determinam um mesmo conjunto de soluções factíveis inteiras, apesar de serem diferentes. Cada uma dessas regiões é chamada de formulação para o problema.

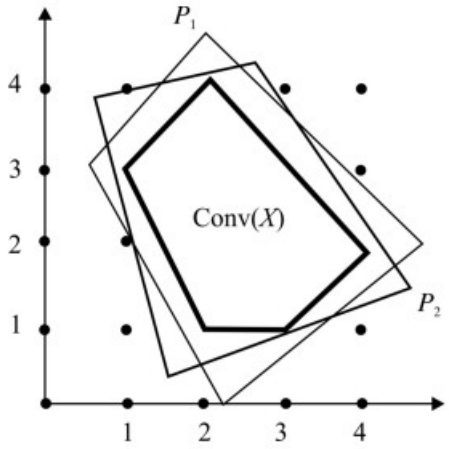
Figura 7 - Formulações para o problema



Fonte: (ARENALES et al., 2015)

Uma formulação qualquer P é dita ser melhor que uma formulação Q se P está contida em Q. Existe uma formulação ideal, a envoltória convexa, que está contida em todas as formulações possíveis para um problema de programação inteira, como pode ser visto na Figura 8, pela região representada por $Conv(X)$.

Figura 8 - Envoltória convexa



Fonte: (ARENALES et al., 2015)

Se a envoltória convexa for obtida para um problema de programação inteira, técnicas de programação linear, como o método simplex, podem ser usadas. Isso porque a solução do problema de programação linear sempre é um ponto extremo da região, que nesse caso será um ponto inteiro, sendo também solução para o problema original.

A ideia da técnica de planos de corte é melhorar a formulação de problemas, obtendo uma região mais próxima da envoltória convexa. Isso é feito por meio do conceito de desigualdade válida. Uma desigualdade $\phi x \leqslant \phi_0$ é dita válida para um conjunto $X \in \mathbb{R}^n$ se $\phi x \leqslant \phi_0$ para todo $x \in X$. Ou seja, a desigualdade é válida se todos os elementos do conjunto X se situam no semi-espaço determinado pela desigualdade.

Existe um procedimento geral, chamado de procedimento de Chvátal-Gomory, para construção de desigualdades válidas para conjuntos do tipo $X = \{x : Ax \leqslant b,\ x \in \mathbb{Z}_+^n\}$, com A sendo uma matriz $(m \times n)$ constituída das colunas $\{a_1,\ a_2,\ \ldots,\ a_n\}$. No próximo parágrafo o desenvolvimento matemático desse procedimento é mostrado.

Sabe-se que $\sum_{j=1}^{n} a_j x_j \leqslant b$ e que, se um vetor $u \in \mathbb{R}_+^m$ for escolhido, poderá ser multiplicado em ambos os lados da inequação que esta continuará válida, pois todos os elementos de u são maiores ou iguais a zero. Logo, obtém-se que $\sum_{j=1}^{n} u^T a_j x_j \leqslant u^T b$

Pode-se também afirmar que $\sum_{j=1}^{n} \lfloor u^T a_j \rfloor x_j \leqslant \sum_{j=1}^{n} u^T a_j x_j$, pois por definição da função piso $\lfloor u^T a_j \rfloor \leqslant u^T a_j$ e $x_j \geqslant 0$. Por transitividade da desigualdade conclui-se que $\sum_{j=1}^{n} \lfloor u^T a_j \rfloor x_j \leqslant u^T b$. Além disso, como tanto x_j quanto o resultado da função piso $\lfloor u^T a_j \rfloor$ são inteiros, conclui-se $\sum_{j=1}^{n} \lfloor u^T a_j \rfloor x_j \leqslant \lfloor u^T b \rfloor$. É possível demonstrar que qualquer desigualdade válida para o conjunto X pode ser obtida pela execução desse procedimento um número finito de vezes (ARENALES et al., 2015).

A explicação dos conceitos de desigualdade válida e procedimento de Chvátal-Gomory permite que o algoritmo de Gomory seja explicado. O algoritmo de Gomory, como é apresentado no livro do Arenales, é composto por 4 passos, explicados a seguir:

Passo 1) Faça $k = 0$ e defina $PL_0 = PL$, com PL sendo a relaxação linear do problema original P.

Passo 2) Resolva o problema linear PL_k

Passo 3) Se a solução obtida para o problema linear for inteira, então trata-se da solução ótima. Se não, prossiga para o passo 4.

Passo 4) Escolha uma restrição adequada (a linha deve satisfazer uma certa condição) e aplique o corte de Gomory. Faça $k = k+1$, adicione a nova restrição ao problema PL_k e prossiga para o Passo 2. É possível provar que esse algoritmo gera uma solução factível após um número finito de passos e que a primeira solução factível gerada é a solução ótima.

O processo de solução utilizado por pacotes computacionais típicos para problemas de otimização linear inteira mista é conhecido como *branch-and-cut*. O *branch-and-cut* é um arcabouço que combina a árvore de *branch-and-bound* com planos de corte (como os de Chvátal-Gomory). Em cada nó da árvore há um potencial para a adição de planos de corte e, consequentemente, uma possível melhoria do limitante superior (para problemas de maximização). Em particular a adição de cortes na raiz da árvore é importante para a aumentar a poda de nós da árvore ao longo do processo de solução. Mais detalhes desse processo de solução podem ser vistos nos livros: Conforti et al. (2014), Arenales et al. (2015) e Wolsey (2020).

Embora todos os pacotes comparados neste trabalho utilizem o branch-and-cut, eles podem diferir nas estratégias utilizadas, como nas técnicas de geração e seleção de cortes, regras de *branching,* etc. O leitor interessado pode consultar o trabalho de Bolusani et al. (2024), que aborda estratégias comumente utilizadas por ferramentas baseadas em *branch-and-cut*, além de detalhes de implementação do SCIP.

2.5 PROBLEMA DE ROTEAMENTO DE VEÍCULOS

A logística têm uma importância fundamental na economia atual, podendo representar até 20% do valor final de produtos (TOTH, 2002). Por exemplo, todos os dias milhões de pessoas compram produtos em *e-commerces* e desejam que o serviço de entrega fornecido seja barato, rápido e de qualidade. O mesmo ocorre em aplicativos de *delivery* de comida. Para que os sistemas de logística cumpram suas expectativas, é necessário que uma quantidade grande de informação referente a veículos, motoristas, rotas, entre outras, sejam processadas e analisadas para gerar um planejamento adequado.

A literatura de Pesquisa Operacional define uma classe de problemas de logística denominada Problemas de Roteamento de Veículos. Os problemas desta classe consistem, de forma geral, em determinar o melhor conjunto de rotas, segundo algum critério, que atenda as demandas dos clientes, utilizando-se de um conjunto de veículos, motoristas, depósitos e uma rede de estradas (TOTH, 2002). Esses problemas possuem extensas aplicações práticas, sendo muito importantes para a economia atual.

Dentre esses problemas, o problema de roteamento de veículos capacitados é um dos mais antigos e estudados. Esse problema consiste em um único depósito, uma frota de veículos com capacidade conhecida, um conjunto de clientes com demanda conhecida e uma rede de estradas por onde os veículos podem trafegar, representada por meio de um grafo com um custo associado a cada aresta. O objetivo é determinar um conjunto de rotas que atenda as demandas dos clientes, cumpra todas as restrições e tenha custo mínimo. Outros problemas dessa classe

introduzem mais complexidades, como janelas de tempo em que os clientes podem ser atendidos ou a definição de um conjunto de depósitos, em vez de apenas um.

O problema de Roteamento de Veículos com Múltiplos Depósitos, abordado neste trabalho, possui aplicações práticas documentadas, incluindo distribuição de produtos do petróleo (BROWN et al., 1987), entrega de refeições (CASSIDY; BENNETT, 1972), gases industriais (BELL et al., 1983) e alívio de desastres (VIEIRA et al., 2021). Formalmente, este problema pode ser definido por um grafo $G = (V, A)$, em que $V = \{1, 2, 3, \ldots, n+w\}$ é o conjunto de vértices, com o número de clientes sendo representado por n e o número de depósitos representado por w, e $A = \{(i, j): i, j \in V, i \neq j\}$ é o conjunto de arestas. O conjunto V é particionado em dois subconjuntos, o conjunto de clientes $V_c = \{1, 2, \ldots, n\}$ e o de depósitos $V_d = \{n+1, n+2, \ldots, n+w\}$. Em cada depósito está disponível uma frota de veículos com capacidade Q e cada cliente $i \in V_c$ possui uma demanda não negativa p_i e uma duração de serviço não negativa t_i. Além disso, cada aresta em A é associada a uma distância d_{ij}. O objetivo nesse problema é determinar o conjunto de rotas com custo mínimo que: (a) Cada rota começa e termina no mesmo depósito, (b) Cada cliente é visitado exatamente uma vez por um único veículo, (c) a soma das demandas dos clientes em cada rota não ultrapassa a capacidade Q do veículo e (d) o tempo total da rota não ultrapassa um determinado limite.

Na Figura 1, pode-se ver uma instância do Problema de Roteamento com Múltiplos depósitos que consiste em 2 depósitos (A, B), 12 clientes e 4 rotas. O primeiro par de números em cada cliente representa as coordenadas e o segundo número, entre parênteses, representa a demanda do cliente. Nos depósitos os pares ordenados representam as coordenadas e os números entre parênteses, (24) e (12), representam a capacidade do depósito e do veículo, respectivamente. É possível observar as condições descritas anteriormente nesse exemplo, cada cliente é atendido exatamente uma vez, a soma das demandas dos clientes não ultrapassa a capacidade do veículo, que nesse caso é 12, e cada rota começa e termina no mesmo depósito.

3 REVISÃO DA LITERATURA

A pesquisa sobre o Problema de Roteamento de Veículos com Múltiplos Depósitos geralmente se divide em duas categorias principais: métodos exatos e métodos heurísticos. Na abordagem desta revisão de literatura, a atenção é direcionada especificamente aos trabalhos relacionados a métodos exatos. A escolha dessa ênfase se justifica pela forte correlação desses métodos com o escopo e os objetivos do presente trabalho.

Na década de 70, os primeiros estudos sobre a distribuição de mercadorias com múltiplos depósitos começaram a surgir. No entanto, foi somente no trabalho de Laporte, Norbert e Arpin (1984) que modelos e procedimentos formais foram propostos para resolver o Problema de Roteamento de Veículos com Múltiplos Depósitos (MONTOYA-TORRES et al, 2015).

Neste trabalho, os autores formularam a versão simétrica do MDVRP como um problema de programação inteira e apresentaram um algoritmo *branch-and-bound* com utilização de relaxação por Programação Linear.

Outros trabalhos pioneiros em métodos exatos incluem Kulkarni e Bhave (1985), Laporte et al. (1988), e Carpaneto, Dell'amico, Fischetti, e Toth (1989). Kulkarni e Bhave propuseram um modelo simples e de fácil entendimento considerando uma frota de veículos com a mesma capacidade de carga e custo máximo. Apesar do trabalho não avaliar o desempenho computacional do modelo, este apresentou uma redução considerável no número de variáveis, sendo posteriormente comentado em (Laporte, 1989).

Em um trabalho mais recente, o artigo de Baldacci e Mingozzi (2009) descreve um método exato para o Problema de Roteamento de Veículos Heterogêneos. Este problema recebe esse nome devido à variação na frota de veículos disponíveis, considerando custos fixos e capacidade. O método proposto pode ser aplicado para resolver o Problema de Roteamento de Veículos Capacitados, o Problema de Roteamento de Veículos Dependente do Local e o Problema de Roteamento de Veículos com Múltiplos Depósitos.

O algoritmo desenvolvido pelos autores é baseado na formulação de particionamento de conjunto e utiliza três tipos de procedimentos de limitantes, os

quais se baseiam em relaxamento por Programação Linear e Lagrangeana. Esses procedimentos reduzem o número de variáveis presentes no problema.

O artigo também apresenta resultados obtidos para instâncias do MDVRP com até 200 clientes e 2-5 depósitos. Os resultados indicam que o limitante inferior obtido pelos autores superou os valores encontrados na literatura. Além disso, o algoritmo demonstrou a capacidade de resolver algumas instâncias dos problemas, incluindo instâncias do MDVRP, pela primeira vez.

Contardo e Martinelli (2014) apresentam um novo método exato para resolver o Problema de Roteamento de Veículos com Múltiplos Depósitos (MDVRP) considerando restrições de capacidade e comprimento de rota. Este método integra a formulação de fluxo de veículos com a técnica de planos de corte, além da formulação de particionamento de conjuntos em conjunto com a estratégia de geração de colunas.

O método proposto faz uso de diversas famílias de desigualdades válidas, possibilitando a obtenção de bons limitantes inferiores para o problema, demonstrando superioridade em relação a métodos do estado da arte nas instâncias estudadas. Vale ressaltar que o artigo alcança um resultado notável ao ser o primeiro, baseado em geração de colunas, a resolver com sucesso a instância *F-n135-k7* do Problema de Roteamento de Veículos Capacitados.

Bettinelli, Ceselli e Righini (2011) propuseram um algoritmo de *branch-and-cut-and-price* para o problema de roteamento de veículos heterogêneos com múltiplos depósitos e janelas de tempo. Esse problema adiciona novas complexidades ao MDVRP, considerando veículos com diferentes capacidades e janelas de tempo nas quais os clientes devem ser atendidos. O estudo foi feito considerando diferentes estratégias de corte e de precificação e as avalia em três diferentes experimentos em três conjuntos de instâncias diferentes.

Ramos, Gomes e Póvoa (2019) propõem uma nova formulação para o MDVRP, chamada de *two-commodity flow*, adaptada de uma formulação proposta por Baldacci, Hadjiconstantinou e Mingozzi (2004) para o CVRP. Para analisar o desempenho da formulação proposta, o artigo a compara com a formulação de três índices, mais tradicional na literatura. Na formulação de três índices, os autores consideram quatro conjuntos de restrições para eliminação de subciclos, a restrição Dantzig–Fulkerson–Johnson, a de Miller–Tucker–Zemlin, a restrição *transit-load* e a restrição de tempo de chegada.

Os resultados foram obtidos através da execução de um conjunto de instâncias, considerando as diferentes formulações, no software comercial CPLEX. A nova formulação proposta no artigo mostrou-se superior à formulação de três índices nos experimentos realizados, precisando, por exemplo, de apenas 129 segundos em média para provar a otimalidade da solução para instâncias de 25 clientes, enquanto a formulação de três índices demorou 1035 segundos em média para chegar ao mesmo resultado. Esta formulação, explicada no capítulo 5 deste documento, será usada neste trabalho por sua ampla utilização em problemas de roteamento.

Além dos trabalhos de Roteamento de Veículos, também foi consultada a literatura acerca de trabalhos de avaliação experimental de pacotes computacionais. Um dos trabalhos mais relevantes nesse contexto é o do MIPLIB. O projeto MIPLIB é conhecido por manter um *benchmark* de diversos pacotes de programação linear inteira-mista. A principal diferença do projeto MIPLIB com o proposto neste trabalho é relacionado ao domínio de aplicações. O MIPLIB seleciona instâncias desafiadoras de diversas áreas do conhecimento, visando uma avaliação abrangente do desempenho dos pacotes computacionais analisados. Os próprios autores reconhecem a limitação de extrapolar esses resultados para situações específicas. Portanto, o objetivo deste trabalho é avaliar pacotes computacionais de forma específica para o Problema de Roteamento de Veículos com Múltiplos Depósitos, preenchendo uma lacuna identificada pelo autor, que não tem conhecimento de outra pesquisa com esse escopo.

Uma pesquisa similar, realizada por Castellucci, Bernardes e Leão (2023), mas para o problema das P-medianas, comparou o desempenho dos pacotes CBC, SCIP, CPLEX e Gurobi, os mesmos que são utilizados neste trabalho. Além da comparação de desempenho entre pacotes, os autores também consideraram duas formulações diferentes nos experimentos.

4 FORMULAÇÃO DOS TRÊS ÍNDICES

O modelo que será empregado nos testes é conhecido como a formulação dos três índices. Essa formulação teve sua origem no trabalho de Golden, Magnanti e Nguyen, de 1977, e posteriormente foi adaptada por Ramos, Gomes e Póvoa (2019) para acomodar uma frota de veículos distribuída entre múltiplos depósitos.

A formulação completa, considerando a função objetivo e as restrições, é mostrada abaixo. O conjunto $V_c = \{1, 2, ..., n\}$ representa os nós clientes, $V_d = \{n+1, n+2, ..., n+w\}$ representa os nós depósitos e o conjunto $V = V_c \cup V_d$ representa todos os nós. O conjunto K_i representa o conjunto de veículos que pertence ao depósito i e o conjunto K representa todos os veículos. Quanto aos parâmetros do modelo, d_{ij} e r_{ij} representam, respectivamente, a distância e o tempo gasto na viagem entre o nó i e j, Q_k representa a capacidade de carga do veículo k, p_i e t_i representam, respectivamente, a demanda e o tempo de serviço do cliente i e T representa o tempo máximo permitido para uma rota. As variáveis de decisão do modelo são as variáveis binárias x_{ijk} que indicam se um veículo k viaja do nodo i para o nodo j.

$$min \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} x_{ijk} d_{ij} \tag{15}$$

$$\sum_{i \in V} \sum_{k \in K} x_{ijk} = 1 \quad \forall j \in V_c \tag{16}$$

$$\sum_{j \in V} \sum_{k \in K} x_{ijk} = 1 \quad \forall i \in V_c \tag{17}$$

$$\sum_{i \in V} x_{ihk} - \sum_{j \in V} x_{hjk} = 0 \quad \forall k \in K, \ \forall h \in V \tag{18}$$

$$\sum_{i \in V_c} \sum_{j \in V} p_i x_{ijk} \leqslant Q_k \forall k \in K \tag{19}$$

$$\sum_{i \in V} \sum_{j \in V} t_i x_{ijk} + \sum_{i \in V} \sum_{j \in V} r_{ij} x_{ijk} \leqslant T \quad \forall k \in K \tag{20}$$

$$\sum_{j \in V_c} x_{ijk} \leqslant 1 \quad \forall k \in K_i, \ \forall i \in V_d \tag{21}$$

$$\sum_{i \in V_c} x_{ijk} \leqslant 1 \quad \forall k \in K_j, \ \forall j \in V_d \tag{22}$$

$$\sum_{i \in V_c} x_{ijk} = 0 \quad \forall j \in V_d, \ \forall k \notin K_j \tag{23}$$

$$\sum_{j \in V_c} x_{ijk} = 0 \quad \forall i \in V_d, \ \forall k \notin K_i \tag{24}$$

$$u_i - u_j + n \times x_{ijk} \leqslant n - 1 \quad 1 \leqslant i \neq j \leqslant n, \forall k \in K \tag{25}$$

$$x_{ijk} \in \{0,1\} \quad \forall i \in V, \ \forall j \in V, \ \forall k \in K \tag{26}$$

A função objetivo (15) define que a distância total percorrida, considerando todos os veículos da frota, deve ser minimizada. As restrições (16) e (17) asseguram que cada cliente será atendido exclusivamente por um único veículo, uma única vez. As restrições (18) garantem que o número de vezes que um veículo “entra” em um nó é o mesmo número de vezes que ele “sai”. Visto que as restrições anteriores estipulam que um veículo visita cada cliente apenas uma vez, essa restrição assegura a continuidade da rota. As restrições (19) garantem que a soma das demandas de cada um dos clientes atendidos por um determinado veículo não ultrapasse a capacidade desse veículo. As restrições (20) determinam que a soma dos tempos de atendimento ao cliente mais o tempo de viagem entre os nodos não ultrapasse um tempo pré-determinado T. As restrições (21) em conjunto com as (22) estabelecem que o veículo deve sair e voltar para o seu depósito uma única vez. As restrições (23) e (24) garantem que um veículo não consiga seguir uma rota que começa e termina em algum depósito que não seja o seu. O artigo de Ramos, Gomes e Póvoa apresenta 4 diferentes restrições para a eliminação de subciclos. Dentre as quatro restrições propostas, a de Miller-Tucker-Zemlin foi escolhida por sua popularidade e compatibilidade com pacotes de programação linear inteira mista, representada na restrição (25). As restrições (26) apenas determinam que as variáveis de decisão são binárias.

5 RESULTADOS DOS EXPERIMENTOS

Nesta seção são apresentados os resultados obtidos nos experimentos realizados com o objetivo de comparar o desempenho dos pacotes computacionais CBC, CPLEX, Gurobi, SCIP. Nos pacotes considerados, a configuração padrão foi usada, foi permitida a utilização de apenas uma *thread* e o tempo de execução foi limitado a no máximo uma hora. A criação do modelo nos pacotes foi feita com a biblioteca PuLP na versão 2.7.0 e Python 3.10[1]. As versões utilizadas foram CBC 2.10, CPLEX 22.1, Gurobi 10.0 e SCIP 9.0. Os experimentos foram executados em computador equipado com o processador Intel Core i7-7700 CPU @ 3.60GHz x 8, 16 GiB de memória RAM e o Ubuntu 22.04.4 foi utilizado como sistema operacional.

O conjunto de instâncias utilizado neste trabalho foi baseado naquele empregado por Ramos, Gomes e Póvoa (2019). O trabalho de Ramos utilizou instâncias de Cordeau, Gendreau e Laporte (1997), mas, como o foco estava em instâncias de pequeno e médio porte, aquelas com mais de cem clientes foram ignoradas. Como após esse filtro poucas instâncias restaram, 10 delas foram modificadas (subconjunto PR) para que tivessem 25 e 40 clientes, criando os novos subconjuntos PR_25 e PR_40. Este trabalho utiliza estas novas instâncias geradas, assim como aquelas com mais de cem clientes.

A Tabela 1 faz uma comparação dos resultados obtidos pelos pacotes em relação à melhor solução encontrada para cada uma das instâncias consideradas. Na tabela, a primeira coluna mostra o nome da instância e as três próximas colunas mostram a quantidade de depósitos, consumidores e veículos dessa instância. A quinta coluna mostra a melhor solução encontrada para a instância de acordo com o artigo de Ramos, Gomes, Póvoa (2019). As quatro próximas colunas mostram a melhor solução encontrada por cada um dos pacotes. Na tabela, os traços são usados para indicar que o pacote não conseguiu encontrar uma solução ou que a informação não foi encontrada no artigo.

Os valores da Tabela 1 mostram uma clara superioridade do pacote Gurobi, que obteve o melhor resultado entre os pacotes em 28 das 53 instâncias consideradas. O segundo melhor desempenho foi obtido pelo CPLEX, que obteve o

1 O código desenvolvido pode ser visualizado no link https://github.com/EduardoC1705/TCC-EduardoCostaPereira

melhor resultado em 11 das 53 instâncias. Dessas 11, em 10 delas o Gurobi obteve o mesmo valor. O SCIP obteve o melhor resultado em 5 instâncias, e em todas elas o Gurobi e o CPLEX também chegaram à mesma solução. O CBC obteve o melhor resultado em apenas uma instância, e os outros três pacotes chegaram ao mesmo resultado.

Tabela 1 - Melhores soluções encontradas pelos pacotes

Instância	Qtd. Depósitos	Qtd. Consumido-res	Qtd. Veículos	Ramos, Gomes, Póvoa (2019)	CPLEX	Gurobi	SCIP	CBC
p01	4	50	16	576.87	789.72	597.07	1047.57	938.38
p02	4	50	8	473.53	855.40	484.24	809.24	1026.39
p03	5	75	15	641.19	1412.78	671.18	_	1364.93
p04	2	100	16	1001.04	2140.08	1128.64	_	2002.22
p05	2	100	10	750.03	1467.48	866.99	_	_
p06	3	100	18	876.5	1889.34	982.79	_	_
p07	4	100	16	881.97	1920.06	995.23	_	1867.91
p08	2	249	28	_	_	_	_	_
p09	3	249	36	_	_	_	_	_
p10	4	249	36	_	_	_	_	_
p11	5	249	30	_	_	_	_	_
p12	2	80	10	1318.95	2854.77	1320.99		_3302.43
p13	2	80	10	1318.95	_	_	_	_
p14	2	80	10	1360.12	_	_	_	_
p15	4	160	20	_	_	4003.56	_	_
p16	4	160	20	_	_	_	_	_
p17	4	160	20	_	_	_	_	_
p18	6	240	30	_	_	_	_	_

p19	6	240	30	_	_	_	_	_
p20	6	240	30	_	_	_	_	_
p21	9	360	45	_	_	_	_	_
p22	9	360	45	_	_	_	_	_
p23	9	360	45	_	_	_	_	_
pr01	4	48	4	861.32	1036.71	_	1167.82	_
pr01_25	4	25	4	590.35	590.35	590.35	598.24	656.16
pr01_40	4	40	4	726.82	922.11	726.82	1036.85	_
pr02	4	96	8	1307.34	_	_	_	_
pr02_25	4	25	8	716.38	716.91	716.98	723.40	1036.15
pr02_40	4	40	8	908.73	1130.15	963.69	1193.79	_
pr03	4	144	12	_	_	_	_	_
pr03_25	4	25	12	575.40	575.40	575.40	575.40	588.41
pr03_40	4	40	12	882.10	1057.56	882.13	1221.99	1160.58
pr04	4	192	16	_	_	_	_	_
pr04_25	4	25	16	703.52	703.52	703.52	703.52	837.14
pr04_40	4	40	16	809.64	_	809.64	1142.18	_
pr05	4	240	20	_	_	_	_	_
pr05_25	4	25	20	481.54	481.54	481.54	481.54	507.67
pr05_40	4	40	20	673.45	689.03	673.45	849.53	1330.22
pr06	4	288	24	_	_	_	_	_
pr06_25	4	25	24	659.56	659.56	659.56	659.56	944.01
pr06_40	4	40	24	743.47	770.38	743.47	1234.87	1830.55
pr07	6	72	6	1089.56	_	_	_	_

pr07_25	6	25	6	684.01	684.01	684.01	684.35	716.38
pr07_40	6	40	6	839.93	969.07	870.77	999.90	_
pr08	6	144	12	_	_	_	_	_
pr08_25	6	25	12	612.48	612.79	612.79	615.71	797.94
pr08_40	6	40	12	755.46	848.76	751.31	1185.37	1426.42
pr09	6	216	18	_	_	_	_	_
pr09_25	6	25	18	707.12	707.12	707.12	709.33	795.42
pr09_40	6	40	18	864.03	1004.93	864.28	1345.88	_
pr10	6	288	24	_	_	_	_	_
pr10_25	6	25	24	518.95	518.95	518.95	518.95	518.95
pr10_40	6	40	24	738.22	754.24	738.45	1199.94	_

A Tabela 2 mostra a média e a mediana obtida em cada um dos pacotes, agrupados por tipo de instância, para as métricas de melhor limitante da raiz, *gap* e tempo de solução. O *gap* mede o quão distante o melhor limitante inferior está do melhor limitante superior (melhor solução encontrada) em percentual. Em termos matemáticos, é o melhor limitante superior menos o melhor limitante inferior dividido pelo melhor limitante superior.

Quanto ao melhor tempo de solução, quase todas as instâncias utilizaram o tempo máximo permitido. O melhor ponto de comparação para essa métrica é o conjunto de instâncias PR_25, que por serem menores tiveram mais ocorrências de tempos inferiores ao tempo limite. Para esse grupo de instâncias o pacote Gurobi se mostrou superior aos seus concorrentes, apresentando uma média e mediana consideravelmente inferior. Os pacotes CPLEX e SCIP apresentaram resultados semelhantes, com uma pequena vantagem para o CPLEX. O CBC apresentou o pior resultado, com tanto a média quanto a mediana igualando o tempo limite.

Quanto ao melhor limitante da raiz, para os conjuntos de instâncias PR_25 e PR_40 os quatro conjuntos apresentaram valores semelhantes, com uma breve vantagem ao CBC. Nos conjuntos P e PR os valores se tornam mais discrepantes, com uma clara vantagem ao pacote Gurobi, que se destacou principalmente no

conjunto P. O segundo melhor resultado apresentado nesses conjuntos foi o do CPLEX, seguido pelo CBC e pelo SCIP, que apresentou o pior resultado.

Tabela 2 - Média e mediana do melhor limitante da raiz, *gap* e tempo de solução agrupado pelos subconjuntos P, PR, PR_25 e PR_40.

Subconjunto	Pacote	Melhor limitante da raiz		Gap		Tempo de solução(s)	
		Média	Mediana	Média	Mediana	Média	Mediana
P	CBC	919.52	579.90	66.60%	67.11%	3600	3600
P	CPLEX	1391.27	1398.82	64.16%	66.87%	3600	3600
P	Gurobi	1646.29	1865.10	35.70%	33.10%	3600	3600
P	SCIP	918.44	579.90	57.20%	57.20%	3600	3600
PR	CBC	1160.75	1214.43	_	_	3600	3600
PR	CPLEX	1262.07	1297.63	21.54%	21.54%	3600	3600
PR	Gurobi	1307.37	1330.76	_	_	3600	3600
PR	SCIP	983.34	928.72	31.58%	31.58%	3600	3600
PR_25	CBC	488.35	495.60	27.24%	26.21%	3600	3600
PR_25	CPLEX	465.02	450.06	3.76%	1.51%	2752	3600
PR_25	Gurobi	465.02	450.06	2.33%	0.00%	1609	798
PR_25	SCIP	465.21	450.79	5.96%	5.72%	3333	3600
PR_40	CBC	610.64	620.84	56.96%	62.63%	3600	3600
PR_40	CPLEX	605.94	616.49	26.93%	27.28%	3600	3600
PR_40	Gurobi	605.94	616.49	11.46%	11.74%	3600	3600
PR_40	SCIP	605.94	616.49	41.73%	38.35%	3600	3600

Em relação ao *gap*, o pacote Gurobi apresentou uma clara vantagem em relação aos seus concorrentes, se destacando principalmente nos conjuntos P e PR_40. O CPLEX apresentou o segundo melhor resultado, seguido pelo SCIP e pelo CBC. Vale ressaltar que apesar do SCIP ter números melhores que o CPLEX no conjunto P, isso ocorreu porque o SCIP encontrou uma solução para apenas duas instâncias nesse conjunto, contra oito do CPLEX. Nessas duas instâncias, o CPLEX apresentou uma média e uma mediana superior ao SCIP. O CBC apresentou o pior resultado, principalmente no conjunto PR_25, no qual se distanciou dos outros três pacotes. O conjunto PR não é bom para comparação dessa métrica pois apenas o SCIP e o CPLEX conseguiram encontrar uma solução e ambos o fizeram para apenas uma instância. Os traços nas linhas do CBC e Gurobi indicam que não foi possível calcular a métrica, visto que esta exige um limitante superior (pelo menos uma solução deve ser encontrada).

6 CONCLUSÕES E TRABALHOS FUTUROS

O Problema de Roteamento de Veículos com Múltiplos Depósitos (Multi-Depot Vehicle Routing Problem, MDVRP) é uma importante variante do Problema de Roteamento de Veículos e possui aplicações práticas documentadas. Algumas formulações de programação linear inteira foram propostas na literatura para esse problema, como, por exemplo, a formulação dos três índices. Essa formulação foi utilizada neste trabalho para comparar o desempenho de pacotes comerciais (CPLEX, Gurobi) e gratuitos (CBC, SCIP) na solução do MDVRP.

Experimentos computacionais foram realizados com um conjunto de 53 instâncias obtido da literatura e as métricas de melhor solução encontrada, melhor limitante da raiz, melhor limitante inferior e tempo de solução foram coletadas e analisadas. No geral, os pacotes computacionais comerciais obtiveram resultados melhores, com o Gurobi se destacando em relação ao CPLEX nesse grupo. No grupo dos gratuitos, o SCIP se sobressaiu em relação ao CBC nas métricas de *gap* e tempo de solução e apresentou um *gap* competitivo com os comerciais para instâncias menores (subconjunto PR_25). Trabalhos futuros podem explorar outros pacotes computacionais ou realizar o estudo para uma formulação diferente do MDVRP.

REFERÊNCIAS

ARENALES, M. N. et al. *Pesquisa operacional*. 2015.

BALDACCI, R.; HADJICONSTANTINOU, E.; MINGOZZI, A. *An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation*. Operations research, v. 52, n. 5, p. 723-738, 2004.

BALDACCI, R.; MINGOZZI, A. *A unified exact method for solving different classes of vehicle routing problems*. Mathematical Programming, v. 120, p. 347-380, 2009.

BELL, W. J. et al. *Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer*. Interfaces, [s. l], v. 13, n. 6, p. 4-23, 1983.

BETTINELLI, A.; CESELLI, A.; RIGHINI, G. *A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows*. Transportation Research Part C: Emerging Technologies, v. 19, n. 5, p. 723-740, 2011

BROWN, G. G. et al. *Real-time, Wide Area Dispatch of Mobil Tank Trucks*. Interfaces, [s. l], v. 17, n. 1, p. 107-120, Feb. 1987.

CARPANETO, G. et al. *A branch and bound algorithm for the multiple depot vehicle scheduling problem*. Networks, v. 19, n. 5, p. 531-548, 1989.

CASSIDY, P. J.; BENNETT, H. S. *TRAMP—a multi-depot vehicle scheduling system.* Journal of the Operational Research Society, [s. l], v. 23, p. 151-163, June 1972.

CASTELUCCI, P. B.; BERNARDES, E. D.; LEÃO, A. A. S. *Revisitando experimentalmente modelos para o problema das p-medianas*. In: ANAIS DO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 2023, São José dos Campos. Anais eletrônicos... Campinas, Galoá, 2023. Acesso em: 27 nov. 2023

CONFORTI, M. et al. *Integer programming models*. Springer International Publishing, 2014.

CORDEAU, J.; GENDREAU, M.; LAPORTE, G. *A tabu search heuristic for periodic and multi depot vehicle routing problems* - Networks: An International Journal, v. 30, n. 2, p. 105-119, 1997.

CONTARDO, C.; MARTINELLI, R. *A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints*. Discrete Optimization, v. 12, p. 129-146, 2014.

DA SILVA JÚNIOR, O. S.; LOPES, L. A.; BERGMANN, U. *A Free Geographic Information System as a Tool for Multi-Depot Vehicle Routing*. Brazilian Journal of Operations & Production Management, [s. l], v. 8, n. 1, p. 103-120, 2011.

GLEIXNER, Ambros. et al. *MIPLIB 2017: data-driven compilation of the 6th mixedinteger programming library*. Mathematical Programming Computation, v. 13, n. 3, p. 443-490, 2021.

GOLDEN, B. L.; MAGNANTI, T. L.; NGUYEN, H. Q. *Implementing vehicle routing algorithms*. Networks, v. 7, n. 2, p. 113-148, 1977.

KULKARNI, R. V.; BHAVE, P. R. *Integer programming formulations of vehicle routing problems*.European journal of operational research, v. 20, n. 1, p. 58-67, 1985.

LAND, A. H.; DOIG A. G. *An Automatic Method of Solving Discrete Programming Problems*. Econometrica, v. 28, n. 3, 1960, p. 497–520.

LAPORTE, G.. *Optimal solutions to capacitated multidepot vehicle routing problems*. Congressus Nemerantium, v. 4, p. 283-292, 1984.

LAPORTE, G. NOBERT, Y.; TAILLEFER, S. *Solving a family of multi-depot vehicle routing and location-routing problems*. Transportation science, v. 22, n. 3, p. 161-172, 1988.

LAPORTE, G. *Integer programming formulations for the multi-depot vehicle routing problem: Comments on a paper by Kulkarni and Bhave*. European Journal of Operational Research, v. 38, n. 2, p. 228-237, 1989.

MONTOYA-TORRES, J. R. et al. *A literature review on the vehicle routing problem with multiple depots*. Computers & Industrial Engineering, v. 79, p. 115-129, 2015.

RAMOS, T. R. P.; GOMES, M. I.; PÓVOA, A. P. B. *Multi-depot vehicle routing problem: a comparative study of alternative formulations*. International Journal of Logistics Research and Applications, v. 23, n. 2, p. 103-120, 2020.

ROTHLAUF, F. et al.*Design of modern heuristics: principles and application*. Berlin: Springer, 2011.

TOTH, Paolo; VIGO, Daniele (Ed.). *The vehicle routing problem*. Philadelphia: Society for Industrial and Applied Mathematics, 2002.

VIEIRA, Y. E. M.; DE MELLO BANDEIRA, R. A.; DA SILVA JÚNIOR, O. S. *Multidepot vehicle routing problem for large scale disaster relief in drought scenarios: The case of the Brazilian northeast region*. International Journal of Disaster Risk Reduction, v. 58, p. 102193, 2021.

WOLSEY, L. A. *Integer programming*. John Wiley & Sons, 2020

BOLUSANI, S. et al. "The SCIP Optimization Suite 9.0." *arXiv preprint arXiv:2402.17702* (2024).

APÊNDICE A – RESULTADOS DOS EXPERIMENTOS

Tabela 3 - Resultados para o pacote Gurobi

Instância	Melhor limitante da raiz	Melhor solução encontrada	Melhor limitante inferior	Tempo de solução(s)
p01	376.76	597.07	416.10	3600
p02	376.76	484.24	417.55	3600
p03	473.66	671.18	481.48	3600
p04	579.90	1128.64	587.19	3600
p05	579.90	866.99	580.04	3600
p06	579.90	982.79	579.98	3600
p07	579.90	995.23	579.98	3600
p08	1874.63	_	1874.63	3600
p09	1872.84	_	1872.83	3600
p10	1870.63	_	1870.63	3600
p11	1874.63	_	1877.11	3600
p12	932.55	1320.99	952.55	3600
p13	932.55	_	959.28	3600
p14	932.55	_	952.55	3600
p15	1865.10	4003.56	1865.10	3600
p16	1865.10	_	1865.27	3600
p17	1865.10	_	1865.10	3600
p18	2797.65	_	2797.65	3600
p19	2797.65	_	2797.65	3600

Instância	Melhor limitante da raiz	Melhor solução encontrada	Melhor limitante inferior	Tempo de solução(s)
p20	2797.65	_	2797.65	3600
p21	_	_	_	_
p22	4196.47	_	4196.47	3600
p23	4196.47	_	4196.47	3600
pr01	704.57	_	819.92	3600
pr01_25	432.48	590.35	590.31	84
pr01_40	606.68	726.82	701.69	3600
pr02	928.72	_	1116.79	3600
pr02_25	421.24	716.98	683.39	3600
pr02_40	641.57	963.69	844.79	3600
pr03	1264.50	_	1286.17	3600
pr03_25	467.64	575.40	575.35	298
pr03_40	704.70	882.13	790.36	3600
pr04	1330.76	_	1350.45	3600
pr04_25	586.22	703.52	703.52	147
pr04_40	626.30	809.64	758.85	3600
pr05	1441.74	_	1441.74	3600
pr05_25	376.00	481.54	481.50	1102
pr05_40	532.03	673.45	598.42	3600
pr06	1745.80	_	1745.80	3600

Instância	Melhor limitante da raiz	Melhor solução encontrada	Melhor limitante inferior	Tempo de solução(s)
pr06_25	518.76	659.56	659.49	2872
pr06_40	593.77	743.47	671.48	3600
pr07	_	_	_	_
pr07_25	548.15	684.01	683.95	294
pr07_40	659.67	870.77	746.71	3600
pr08	1164.35	_	1182.45	3600
pr08_25	414.38	612.79	550.78	3600
pr08_40	482.62	751.31	612.81	3600
pr09	1488.00	_	1506.88	3600
pr09_25	512.38	707.12	647.06	3600
pr09_40	643.22	864.28	749.55	3600
pr10	1697.90	_	1718.40	3600
pr10_25	373.01	518.95	518.90	495
pr10_40	568.86	738.45	624.77	3600

Tabela 4 - Resultados para o pacote CBC

Instância	Melhor limitante da raiz	Melhor solução encontrada	Melhor limitante inferior	Tempo de solução(s)
p01	376.76	938.38	377.45	3599
p02	376.76	1026.39	377.41	3600
p03	473.66	1364.93	474.20	3593
p04	579.87	2002.22	579.87	3600
p05	579.89	_	580.00	3592
p06	579.90	_	579.90	3570
p07	579.90	1867.91	579.90	3577
p08	_	_	_	_
p09	_	_	_	_
p10	_	_	_	_
p11	_	_	_	_
p12	946.69	3302.43	946.84	3598
p13	932.55	_	932.90	3597
p14	932.55	_	932.55	3596
p15	1865.10	_	1865.10	3404
p16	1865.10	_	1865.10	3600
p17	1865.10	_	1865.10	3600
p18	_	_	_	_
p19	_	_	_	_

Instância	Melhor limitante da raiz	Melhor solução encontrada	Melhor limitante inferior	Tempo de solução(s)
p20	–	–	–	–
p21	–	–	–	–
p22	–	–	–	–
p23	–	–	–	–
pr01	798.37	–	803.96	3600
pr01_25	434.54	656.16	498.65	3600
pr01_40	613.28	–	617.59	3600
pr02	935.98	–	935.98	3595
pr02_25	444.85	1036.15	484.01	3600
pr02_40	641.57	–	646.06	3600
pr03	1264.50	–	1264.50	3594
pr03_25	501.48	588.41	522.62	3600
pr03_40	731.38	1160.58	743.58	3600
pr04	1330.76	–	1330.76	3600
pr04_25	586.22	837.14	599.18	3600
pr04_40	628.40	–	641.35	3600
pr05	1441.74	–	1441.74	3600
pr05_25	385.43	507.67	401.60	3600
pr05_40	532.03	1330.22	537.14	3600
pr06	–	–	–	–

Instância	Melhor limitante da raiz	Melhor solução encontrada	Melhor limitante inferior	Tempo de solução(s)
pr06_25	531.78	944.01	550.08	3600
pr06_40	597.84	1830.55	610.26	3600
pr07	862.27	_	862.27	3599
pr07_25	563.98	716.38	598.29	3600
pr07_40	667.25	_	674.63	3600
pr08	1164.35	_	1164.35	3556
pr08_25	414.38	797.94	472.74	3600
pr08_40	482.62	1426.42	490.19	3600
pr09	1488.00	_	1488.00	3377
pr09_25	531.09	795.42	536.80	3600
pr09_40	643.22	_	647.20	3600
pr10	_	_	_	_
pr10_25	489.73	518.95	502.79	3600
pr10_40	568.86	_	573.85	3600

Tabela 5 - Resultados para o pacote SCIP

Instância	Melhor limitante da raiz	Melhor solução encontrada	Melhor limitante inferior	Tempo de solução (s)
p01	376.76	1047.57	383.42	3600
p02	376.76	809.24	396.46	3600
p03	473.66	_	473.66	3600
p04	579.90	_	579.90	3600
p05	579.90	_	581.61	3600
p06	579.90	_	579.90	3600
p07	579.90	_	579.90	3600
p08	_	_	_	_
p09	_	_	_	_
p10	_	_	_	_
p11	_	_	_	_
p12	932.55	_	932.55	3600
p13	932.55	_	932.55	3600
p14	932.55	_	932.55	3600
p15	1865.10	_	1865.10	3600
p16	1865.10	_	1865.10	3600
p17	1865.10	_	1865.10	3600
p18	_	_	_	_
p19	_	_	_	_

Instância	Melhor limitante da raiz	Melhor solução encontrada	Melhor limitante inferior	Tempo de solução (s)
p20	–	–	–	–
p21	–	–	–	–
p22	–	–	–	–
p23	–	–	–	–
pr01	704.80	1167.82	798.92	3600
pr01_25	433.36	598.24	569.75	3600
pr01_40	606.68	1036.85	663.19	3600
pr02	928.72	–	928.72	3600
pr02_25	421.24	723.40	648.83	3600
pr02_40	641.57	1193.79	751.70	3600
pr03	1264.50	–	1264.50	3600
pr03_25	468.22	575.40	542.89	3600
pr03_40	704.70	1221.99	762.07	3600
pr04	–	–	–	3600
pr04_25	586.55	703.52	703.52	2839
pr04_40	626.30	1142.18	696.07	3600
pr05	–	–	–	3600
pr05_25	376.00	481.54	440.69	3600
pr05_40	532.03	849.53	578.75	3600
pr06	–	–	–	–

Instância	Melhor limitante da raiz	Melhor solução encontrada	Melhor limitante inferior	Tempo de solução (s)
pr06_25	518.84	659.56	621.41	3600
pr06_40	593.77	1234.87	617.33	3600
pr07	854.35	_	902.14	3600
pr07_25	548.15	684.35	655.94	3600
pr07_40	659.67	999.90	752.69	3600
pr08	1164.35	_	1164.35	3600
pr08_25	414.38	615.71	543.12	3600
pr08_40	482.62	1185.37	486.06	3600
pr09	_	_	0.00	3600
pr09_25	512.38	709.33	648.15	3600
pr09_40	643.22	1345.88	662.62	3600
pr10	_	_	_	_
pr10_25	373.01	518.95	518.95	1690
pr10_40	568.86	1199.94	585.88	3600

Tabela 6 - Resultados para o pacote CPLEX

Instância	Melhor limitante da raiz	Melhor solução encontrada	Melhor limitante inferior	Tempo de solução(s)
p01	376.76	789.72	379.39	3600
p02	376.76	855.40	384.02	3600
p03	473.66	1412.78	473.82	3600
p04	579.90	2140.08	579.94	3600
p05	579.90	1467.48	580.45	3600
p06	579.90	1889.34	580.23	3600
p07	579.90	1920.06	579.90	3600
p08	1874.63	_	1874.63	3600
p09	1872.83	_	1872.83	3600
p10	1870.63	_	1870.63	3600
p11	1874.63	_	1874.63	3600
p12	932.55	2854.77	934.05	3600
p13	932.55	_	954.97	3600
p14	932.55	_	932.55	3600
p15	1865.10	_	1865.10	3600
p16	1865.10	_	1865.10	3600
p17	1865.10	_	1865.10	3600
p18	2797.65	_	2797.65	3600
p19	2797.65	_	2797.65	3600

Instância	Melhor limitante da raiz	Melhor solução encontrada	Melhor limitante inferior	Tempo de solução(s)
p20	2797.65	_	2797.65	3600
p21	_	_	_	_
p22	_	_	_	3600
p23	_	_	_	_
pr01	704.57	1036.71	813.37	3600
pr01_25	432.48	590.35	590.35	1281
pr01_40	606.68	922.11	670.57	3600
pr02	928.72	_	1065.79	3600
pr02_25	421.24	716.91	674.68	3600
pr02_40	641.57	1130.15	717.03	3600
pr03	1264.50	_	1277.41	3600
pr03_25	467.64	575.40	575.40	1966
pr03_40	704.70	1057.56	765.45	3600
pr04	1330.76	_	1330.76	3600
pr04_25	586.22	703.52	691.70	3600
pr04_40	626.30	_	713.31	3600
pr05	1441.74	_	1441.74	3600
pr05_25	376.00	481.54	458.73	3600
pr05_40	532.03	689.03	566.83	3600
pr06	1745.80	_	1745.80	3600

Instância	Melhor limitante da raiz	Melhor solução encontrada	Melhor limitante inferior	Tempo de solução(s)
pr06_25	518.76	659.56	650.77	3600
pr06_40	593.77	770.38	661.14	3600
pr07	854.35	_	964.86	3600
pr07_25	548.15	684.01	684.01	1664
pr07_40	659.67	969.07	748.62	3600
pr08	1164.35	_	1164.35	3600
pr08_25	414.38	612.79	538.86	3600
pr08_40	482.62	848.76	525.88	3600
pr09	1488.00	_	1488.00	3600
pr09_25	512.38	707.12	622.64	3600
pr09_40	643.22	1004.93	657.61	3600
pr10	1697.90	_	1697.90	3600
pr10_25	373.01	518.95	518.95	1010
pr10_40	568.86	754.24	575.80	3600

APÊNDICE B – Artigo SBC

Comparação do desempenho de pacotes computacionais na solução do Problema de Roteamento de Veículos com Múltiplos Depósitos

Eduardo Costa Pereira

Departamento de Informática - Universidade Federal de Santa Catarina (UFSC)

eduardo.c.p@grad.ufsc.br

Abstract. *This work evaluates the performance of commercial and free Mixed Integer Linear Programming computational packages in solving the Multi-Depot Vehicle Routing Problem (MDVRP), using the three-index formulation. The CBC, CPLEX, Gurobi and SCIP packages were compared in the metrics of best solution found, root bound, lower bound and solution time.*

Resumo. *Este trabalho avalia o desempenho de pacotes computacionais de Programação Linear Inteira Mista, comerciais e gratuitos, na solução do Problema de Roteamento de Veículos com Múltiplos Depósitos (Multi-Depot Vehicle Routing Problem, MDVRP), utilizando a formulação dos três índices. Os pacotes CBC, CPLEX, Gurobi e SCIP foram comparados nas métricas de melhor solução encontrada, melhor limitante da raiz, melhor limitante inferior e tempo de solução.*

INTRODUÇÃO

A literatura de pesquisa operacional define uma classe de problemas, chamada de Problemas de Roteamento de Veículos (*Vehicle Routing Problems*, VRP) que inclui diversos problemas de roteamento com algumas diferenças na definição. As aplicações práticas para Problemas de Roteamento de Veículos são diversas, envolvendo, por exemplo, coleta de resíduos sólidos, limpeza de ruas e definição de rotas para ônibus escolares (Toth, 2002). O problema fundamental desta classe é o Problema de Roteamento de Veículos Capacitados (*Capacitated Vehicle Routing Problem*, CVRP) que consiste em um conjunto de veículos que deve servir um conjunto de clientes, por meio de rotas que começam e terminam no depósito, visitam os clientes uma única vez de tal forma que a soma das demandas dos clientes ao longo de uma rota não ultrapasse a capacidade do veículo.

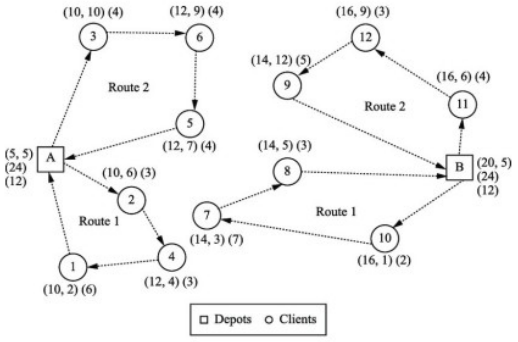
Os outros problemas dessa classe se assemelham a esse, mas com algumas complexidades acrescentadas. Por exemplo, o Problema de Roteamento de Veículos com Janelas de Tempo (*Vehicle Routing Problem With Time Windows*, VRPTW) estabelece uma janela de tempo em que pode ser realizado o atendimento para cada um dos clientes. O Problema de Roteamento com Múltiplos Depósitos (*Multi-Depot Vehicle Routing Problem*, MDVRP), abordado neste artigo, adiciona a complexidade da

existência de múltiplos depósitos disponíveis para realizar o atendimento dos clientes.

PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM MÚLTIPLOS DEPÓSITOS

O Problema de Roteamento de Veículos com Múltiplos Depósitos pode ser definido por um grafo $G=(V,A)$, em que $V=\{1,2,3,...,n+w\}$, é o conjunto de vértices e $A=\{(i,j):\ i,j \in V,\ i \neq j\}$ é o conjunto de arestas. O conjunto de todos os vértices, V, é a união dos conjuntos de clientes $V_c=\{1,2,3,...,n\}$ e de depósitos $V_d=\{n+1,n+2,n+3,...,n+w\}$. Cada um dos clientes possui uma demanda não negativa p_i e uma duração de serviço não negativa t_i. Os depósitos dispõem de uma frota de veículos com capacidade Q. Cada uma das arestas do conjunto A é associada à uma distância d_{ij}, a distância a ser percorrida na viagem direta de i para j. O objetivo nesse problema é determinar o conjunto de rotas com custo mínimo que cumpram os requisitos: (a) As rotas começam e terminam no mesmo depósito, (b) os clientes são visitados exatamente uma vez por um único veículo, (c) em uma rota a soma das demandas dos clientes não ultrapassa e capacidade Q do veículo e (d) o tempo total da rota não ultrapassa um determinado limite.

Figura 1 - Instância do MDVRP



Fonte: (DA SILVA JÚNIOR et al., 2011)

A Figura 1, mostrada acima, representa uma instância do Problema de Roteamento de Veículos com Múltiplos Depósitos com 2 depósitos (A, B), 12 clientes e 4 rotas. Um par ordenado aparece próximo aos depósitos e clientes, representa suas coordenadas, os números ao lado, que aparecem nos nós clientes, representam a demanda, enquanto os números abaixo das coordenadas, que aparecem nos depósitos, representam a capacidade do depósito e dos veículos. As rotas mostradas na figura cumprem os requisitos a, b, c mencionados, mas o tempo das rotas não foi considerado.

FORMULAÇÃO DOS TRÊS ÍNDICES

A formulação dos três índices teve origem no trabalho de Golden, Magnanti e Nguyen, de 1977, e foi posteriormente adaptada no trabalho de Ramos, Gomes e Póvoa (2019) para acomodar uma frota de veículos distribuída entre múltiplos depósitos. Na formulação completa, mostrada abaixo, o conjunto $V_c = \{1,2,3,\ldots,n\}$ representa os nós clientes, $V_d = \{n+1,n+2,n+3,\ldots,n+w\}$ representa os nós depósitos e o conjunto $V = V_c \cup V_d$ representa todos os nós. O conjunto K_i representa o conjunto de veículos que pertence ao depósito i e o conjunto K representa todos os veículos. Quanto aos parâmetros do modelo, d_{ij} e r_{ij} representam, respectivamente, a distância e o tempo gasto ao percorrer o arco ij, Q_k representa a capacidade de carga do veículo k, p_i e t_i representam, respectivamente, a demanda e o tempo de serviço do cliente i e T o tempo máximo que uma rota pode ter. As variáveis de decisão do modelo são as variáveis binárias x_{ijk} que indicam se um veículo k percorre o arco ij.

$$min \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} x_{ijk} d_{ij} \tag{1}$$

$$\sum_{i \in V} \sum_{k \in K} x_{ijk} = 1 \quad \forall j \in V_c \tag{2}$$

$$\sum_{j \in V} \sum_{k \in K} x_{ijk} = 1 \quad \forall i \in V_c \tag{3}$$

$$\sum_{i \in V} x_{ihk} - \sum_{j \in V} x_{hjk} = 0 \quad \forall k \in K, \ \forall h \in V \tag{4}$$

$$\sum_{i \in V_c} \sum_{j \in V} p_i x_{ijk} \leqslant Q_k \forall k \in K \tag{5}$$

$$\sum_{i \in V} \sum_{j \in V} t_i x_{ijk} + \sum_{i \in V} \sum_{j \in V} r_{ij} x_{ijk} \leqslant T \quad \forall k \in K \tag{6}$$

$$\sum_{j \in V_c} x_{ijk} \leqslant 1 \quad \forall k \in K_i, \ \forall i \in V_d \tag{7}$$

$$\sum_{i \in V_c} x_{ijk} \leqslant 1 \quad \forall k \in K_j, \ \forall j \in V_d \tag{8}$$

$$\sum_{i \in V_c} x_{ijk} = 0 \quad \forall j \in V_d, \ \forall k \notin K_j \tag{9}$$

$$\sum_{j \in V_c} x_{ijk} = 0 \quad \forall i \in V_d, \ \forall k \notin K_i \tag{10}$$

$$u_i - u_j + n \times x_{ijk} \leqslant n - 1 \quad 1 \leqslant i \neq j \leqslant n, \forall k \in K \tag{11}$$

$$x_{ijk} \in \{0,1\} \quad \forall i \in V, \ \forall j \in V, \ \forall k \in K \tag{12}$$

A função objetivo (1) determina que a distância total percorrida, considerando todos os veículos da frota, deve ser minimizada. As restrições (2) e (3) asseguram que cada cliente será atendido exclusivamente por um único veículo, uma única vez. As

restrições (4) garantem que o veículo “sai” de um nó o mesmo número de vezes que ele “entra”. As restrições (5) garantem que o respeito a capacidade máxima do veículo. As restrições (6) determinam que a soma dos tempos de atendimento ao cliente mais o tempo gasto em viagem não ultrapasse um tempo T. As restrições (7) em conjunto com as (8) estabelecem que o veículo deve sair e voltar para o seu depósito uma única vez. As restrições (9) e (10) garantem que as rotas comecem e terminem no depósito do veículo. O artigo de Ramos, Gomes e Póvoa apresenta 4 diferentes restrições para a eliminação de subciclos. Dentre as restrições propostas, a de Miller-Tucker-Zemlin foi escolhida por sua popularidade e compatibilidade com pacotes de programação linear inteira mista e pode ser visualizada na restrição (11). As restrições (12) apenas estabelecem que as variáveis de decisão são binárias.

DESCRIÇÃO DOS EXPERIMENTOS

Um conjunto de 53 instâncias do Problema de Roteamento de Veículos com Múltiplos Depósitos foi utilizado nos experimentos para avaliar o desempenho dos pacotes computacionais CPLEX, CBC, Gurobi e SCIP. A formulação dos três índices, conforme definida na seção anterior, foi a utilizada nos experimentos.

O conjunto de instâncias utilizado neste trabalho foi baseado no utilizado por Ramos, Gomes e Póvoa (2019). O trabalho de Ramos utilizou instâncias de Cordeau, Gendreau e Laporte (1997), mas, como o foco estava em instâncias de pequeno e médio porte, instâncias com mais de cem clientes foram ignoradas. Como após esse filtro poucas instâncias restaram, um conjunto de 10 foram modificadas (subconjunto PR) para que tivessem 25 e 40 clientes, criando os novos subconjuntos PR_25 e PR_40. Este trabalho utiliza estas novas instâncias, assim como as com mais de cem clientes.

O computador utilizado nos experimentos possuía o processador Intel Core i7-7700 CPU @ 3.60GHz x 8, 16 GiB de memória RAM e o Ubuntu 22.04.4 como sistema operacional. A criação dos modelos nos pacotes foi feita por meio da biblioteca PuLP na versão 2.7.0 e Python 3.10. Os pacotes estavam na configuração padrão com o número de *threads* limitado à um e tempo de execução máximo de 1 hora, nas seguintes versões: CBC 2.10, CPLEX 22.1, Gurobi 10.0 e SCIP 9.0.

A tabela 1, levando em consideração todo o conjunto de 53 instâncias em que os experimentos foram conduzidos, mostra a pontuação obtida por cada um dos pacotes de acordo com o seguinte critério: Para cada instância, se a melhor solução encontrada por um determinado pacote for igual ou melhor que as soluções encontradas pelos outros pacotes, então o pacote vencedor recebe 1 ponto. Se houver empate na melhor solução encontrada, todos os empatados recebem 1 ponto e se nenhum pacote conseguir encontrar uma solução, pontos não são distribuídos.

Tabela 1 - Número de vezes que cada pacote supera ou iguala todos os seus concorrentes quanto à melhor solução encontrada

Gurobi	CPLEX	SCIP	CBC
28	11	5	1

A Tabela 2 mostra a média e a mediana para as métricas de melhor limitante da raiz, *gap* e tempo computacional para os subconjuntos de instâncias P, PR, PR_25 e PR_40.

No subconjunto P, os pacotes comerciais CPLEX e Gurobi se destacam em relação aos pacotes gratuitos na métrica de melhor limitante da raiz. Na métrica de gap o Gurobi se destaca, apresentando um gap significativamente melhor que os outros três pacotes.

Tabela 2 - Média e mediana das métricas de melhor limitante da raiz, gap e tempo de solução para os subconjuntos P, PR, PR_25 e PR_40

Subconjunto	Pacote	Melhor limitante da raiz		Gap		Tempo de solução(s)	
		Média	Mediana	Média	Mediana	Média	Mediana
P	CBC	919.52	579.90	66.60%	67.11%	3600	3600
P	CPLEX	1391.27	1398.82	64.16%	66.87%	3600	3600
P	Gurobi	1646.29	1865.10	35.70%	33.10%	3600	3600
P	SCIP	918.44	579.90	57.20%	57.20%	3600	3600
PR	CBC	1160.75	1214.43	_	_	3600	3600
PR	CPLEX	1262.07	1297.63	21.54%	21.54%	3600	3600
PR	Gurobi	1307.37	1330.76	_	_	3600	3600
PR	SCIP	983.34	928.72	31.58%	31.58%	3600	3600
PR_25	CBC	488.35	495.60	27.24%	26.21%	3600	3600
PR_25	CPLEX	465.02	450.06	3.76%	1.51%	2752	3600
PR_25	Gurobi	465.02	450.06	2.33%	0.00%	1609	798
PR_25	SCIP	465.21	450.79	5.96%	5.72%	3333	3600
PR_40	CBC	610.64	620.84	56.96%	62.63%	3600	3600
PR_40	CPLEX	605.94	616.49	26.93%	27.28%	3600	3600

PR_40	Gurobi	605.94	616.49	11.46%	11.74%	3600	3600
PR_40	SCIP	605.94	616.49	41.73%	38.35%	3600	3600

No subconjunto PR, na métrica de melhor limitante da raiz os pacotes comerciais apresentam valores semelhantes, superando os pacotes gratuitos, com o pacote SCIP se destacando negativamente. A métrica de *gap* é pouco relevante para esse subconjunto, porque apenas os pacotes CPLEX e SCIP conseguiram encontrar uma solução para alguma instância, e fizeram isso para apenas uma instância. Os pacotes Gurobi e CBC não encontraram uma solução para qualquer instância.

No subconjunto PR_25, o pacote CBC se distancia do restante, apresentando um gap em torno de 27% enquanto os outros três pacotes apresentam um *gap* próximo dos 3%. Na métrica de melhor limitante da raiz todos apresentaram valores muito semelhantes.

No subconjunto PR_40 todos os pacotes apresentaram valores semelhantes quanto à métrica de melhor limitante da raiz, mas na métrica *gap* houve bastante discrepância entre os valores, com o Gurobi novamente se destacando, com uma média/mediana por volta de 11%, seguido pelo CPLEX com 27%, SCIP com 40% e CBC com 60%, aproximadamente.

CONCLUSÃO

Para realizar este estudo, foram conduzidos experimentos computacionais utilizando um conjunto de 53 instâncias provenientes da literatura. Foram analisadas as métricas de melhor solução encontrada, melhor limitante da raiz, melhor limitante inferior e o tempo de solução. De maneira geral, os *solvers* comerciais obtiveram resultados superiores, com o Gurobi destacando-se frente ao CPLEX. Entre os *solvers* gratuitos, o SCIP apresentou desempenho superior ao CBC em termos de *gap* e tempo de solução, além de alcançar um *gap* competitivo com os *solvers* comerciais nas instâncias menores (subconjunto PR_25). Como trabalho futuro, pode-se explorar outros pacotes computacionais ou realizar o estudo para uma formulação alternativa do MDVRP.

REFERÊNCIAS

CORDEAU, J.; GENDREAU, M.; LAPORTE, G. *A tabu search heuristic for periodic and multi depot vehicle routing problems* - Networks: An International Journal, v. 30, n. 2, p. 105-119, 1997.

DA SILVA JÚNIOR, O. S.; LOPES, L. A.; BERGMANN, U. *A Free Geographic Information System as a Tool for Multi-Depot Vehicle Routing*. Brazilian Journal of Operations & Production Management, [s. l], v. 8, n. 1, p. 103-120, 2011.

GOLDEN, B. L.; MAGNANTI, T. L.; NGUYEN, H. Q. *Implementing vehicle routing algorithms*. Networks, v. 7, n. 2, p. 113-148, 1977.

RAMOS, T. R. P.; GOMES, M. I.; PÓVOA, A. P. B. *Multi-depot vehicle routing problem: a comparative study of alternative formulations*. International Journal of Logistics Research and Applications, v. 23, n. 2, p. 103-120, 2020.

TOTH, Paolo; VIGO, Daniele (Ed.). *The vehicle routing problem*. Philadelphia: Society for Industrial and Applied Mathematics, 2002.