Trabalho de Conclusão de Curso

Uma Meta-heurística Híbrida para o Problema da Cobertura de Disco de Raio Fixo

Mateus da Silva Batista
msb@ic.ufal.br

Orientador:

Prof. Dr. Rian Gabriel Santos Pinheiro

Maceió, Novembro de 2024

Mateus da Silva Batista

Uma Meta-heurística Híbrida para o Problema da Cobertura de Disco de Raio Fixo

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação do Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

Prof. Dr. Rian Gabriel Santos Pinheiro

Maceió, Novembro de 2024

Catalogação na Fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária: Sâmela Rouse de Brito Silva – CRB-4 – 6023

B333u Batista, Mateus da Silva.
Uma meta-heurística híbrida para o problema da cobertura de disco de raio fixo / Mateus da Silva Batista. – Maceió, 2024.
38 f. : il., grafs. e tabs. color.

Orientador: Rian Gabriel Santos Pinheiro.
Monografia (Trabalho de conclusão de curso em Engenharia de Computação) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2024.

Bibliografia: f. 35-38.

1. Meta heurística. 2. Discos de raio fixo. 3. Cobertura (Redes de computadores). I. Título.

CDU: 004.023

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL
Instituto de Computação – IC
Campus A. C. Simões - Av. Lourival de Melo Mota, BL 12
Tabuleiro do Martins, Maceió/AL - CEP: 57.072-970
Telefone: (082) 3214-1401

Trabalho de Conclusão de Curso - TCC

Formulário de Avaliação

Nome do Aluno

M	A	T	E	U	S		D	A		S	I	L	V	A		B	A	T	I
S	T	A																	

Nº de Matrícula

1	7	2	1	2	7	9	1					-	

Título do TCC (Tema)

Uma Meta-heurística Híbrida para o Problema da Cobertura de Disco de Raio Fixo

Banca Examinadora

Prof. Dr. Rian Gabriel Santos Pinheiro
Nome do Orientador

Documento assinado digitalmente
gov.br RIAN GABRIEL SANTOS PINHEIRO
Data: 29/11/2024 16:11:37-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Bruno Costa e Silva Nogueira
Nome do Professor

Documento assinado digitalmente
gov.br BRUNO COSTA E SILVA NOGUEIRA
Data: 29/11/2024 16:48:24-0300
Verifique em https://validar.iti.gov.br

Alfredo Lima Moura Silva
Nome do Professor

Documento assinado digitalmente
gov.br ALFREDO LIMA MOURA SILVA
Data: 29/11/2024 16:17:59-0300
Verifique em https://validar.iti.gov.br

Data da Defesa

29 / 11 / 2024

Nota Obtida

9 (NOVE)

Observações

Coordenador do Curso De Acordo

Documento assinado digitalmente
gov.br JOBSON DE ARAUJO NASCIMENTO
Data: 29/11/2024 20:15:56-0300
Verifique em https://validar.iti.gov.br

Assinatura

Agradecimentos

Agradeço, em primeiro lugar, à minha mãe, a pessoa mais importante da minha vida, pelo amor e dedicação incondicionais. Ao meu pai, pelo apoio constante em todas as etapas da minha jornada, e ao meu irmão-amigo, Lucas, cuja amizade e parceria são inestimáveis.

Sou profundamente grato ao professor Rian, que foi essencial nesta caminhada e é, sem dúvida, uma das melhores pessoas que já conheci. Agradeço também a todos os meus amigos da universidade e do laboratório, que tornaram essa experiência ainda mais enriquecedora e marcante.

Um agradecimento especial aos professores Chico Potiguar e Márcio Cavalcante, cujas aulas de matemática, especialmente de cálculo, não só ensinaram, mas também motivaram e inspiraram.

A todos vocês, que de alguma forma contribuíram para a realização deste trabalho e para o meu crescimento pessoal e acadêmico, deixo meu mais sincero agradecimento

Resumo

O problema de cobertura na geometria computacional possui aplicações em uma ampla gama de áreas, incluindo redes de comunicação sem fio, planejamento de localização de instalações, robótica, processamento de imagens e aprendizado de máquina. Além disso, ele desempenha um papel crucial na otimização da alocação de recursos e no design de infraestrutura, abordando desafios como a minimização de custos enquanto garante cobertura total em aplicações como torres de celular, pontos de acesso redes sem fio. Ademais, este problema é fundamental na logística, para a otimização de centros de distribuição, e na defesa militar, para o posicionamento estratégico de sistemas de defesa antimísseis ou transmissores de rádio. Sua ampla aplicabilidade reforça sua importância no enfrentamento de desafios econômicos, sociais e tecnológicos, destacando a necessidade de metodologias inovadoras e eficientes. Este trabalho apresentaa meta-heurística híbrida *Construct, Merge, Solve, and Adapt (CMSA)* para o Problema de Cobertura de Discos de Raio Fixo, que é $\mathcal{NP}$-difícil. O objetivo é determinar o conjunto mínimo de discos, juntamente com suas localizações, necessário para cobrir todos os pontos dados. A meta-heurística proposta foi avaliada por meio de testes experimentais, com a análise comparativa focada no número de discos necessários e nos tempos de execução computacional. Os resultados experimentais demonstram que a meta-heurística CMSA supera significativamente os métodos mais avançados da literatura, obtendo consistentemente as melhores soluções em quase todos os casos de teste, com tempos de execução competitivos.

Palavras-chave: geometria computacional; otimização combinatória; cobertura de disco; meta-heurística; CMSA.

Abstract

The coverage problem in computational geometry has applications across a wide range of fields, including wireless communication networks, facility location planning, robotics, image processing, and machine learning. It plays a crucial role in optimizing resource allocation and infrastructure design, addressing challenges such as cost minimization while ensuring complete coverage in applications like cell towers and wireless access points. Furthermore, this problem is fundamental in logistics for optimizing distribution centers and in military defense for the strategic positioning of anti-missile defense systems or radio transmitters. Its broad applicability underscores its importance in tackling economic, social, and technological challenges, highlighting the need for innovative and efficient methodologies. This work presents a hybrid meta-heuristic *Construct, Merge, Solve, and Adapt (CMSA)* for the Fixed Radius Disk Coverage Problem, which is $\mathcal{NP}$-hard. The objective is to determine the minimum set of disks, along with their locations, required to cover all given points. The proposed CMSA meta-heuristic was evaluated through experimental tests. The comparative analysis focused on the number of disks required and computational execution times. The experimental results demonstrate that the CMSA meta-heuristic significantly outperforms the most advanced methods in the literature, consistently obtaining the best solutions in almost all test cases, with competitive execution times.

Keywords: computational geometry; combinatorial optimization; disk coverage; meta-heuristic; CMSA.

Conteúdo

1 Introdução **1**
- 1.1 Motivação . . . 2
- 1.2 Objetivo . . . 2
  - 1.2.1 Objetivos Específicos . . . 3
- 1.3 Organização do Trabalho . . . 3

2 Trabalhos Relacionados **4**

3 Fundamentação teórica **7**
- 3.1 NP-completude . . . 8
- 3.2 Abordagens Exatas . . . 10
  - 3.2.1 Programação matemática . . . 10
  - 3.2.2 Problema de Programação Linear . . . 10
  - 3.2.3 Problema de Programação Linear Inteira . . . 11
- 3.3 Branch-and-Bound . . . 11
- 3.4 Abordagens Aproximadas . . . 12
  - 3.4.1 Meta-heurística . . . 12
  - 3.4.2 CMSA . . . 14

4 Metodologia **17**
- 4.1 Caracterização do problema . . . 18

5 CMSA **21**
- 5.1 CMSA Aplicado ao PCDRF . . . 21
- 5.2 Construct . . . 22
- 5.3 Merge . . . 24
- 5.4 Solve . . . 24
- 5.5 Adapt . . . 25

6 Resultado e Discussão **26**
- 6.1 Ambiente e ferramentas . . . 26
- 6.2 Parâmetros . . . 26
- 6.3 Instâncias . . . 27
- 6.4 Resultados experimentais . . . 28
  - 6.4.1 FastCover . . . 28
  - 6.4.2 DR . . . 29
- 6.5 Discussão . . . 29

7 Considerações Finais **33**

Referências bibliográficas **34**

Lista de Figuras

3.1 Exemplos de abordagens. 8
3.2 Árvore de enumeração do *branch-and-bound*. 12
3.3 Representação de um ótimo local e um ótimo global. 13
4.1 Etapas metodológicas. 17
4.2 Exemplo de entrada 20
4.3 Solução viável 20
4.4 Solução ótima 20
5.1 CMSA aplicado ao PCDRF 22
6.1 Solução do CMSA para bairros de Maceió 31
6.2 Solução do *FastCover* para bairros de Maceió 31

Lista de Tabelas

2.1 Classificação dos artigo pela a abordagem proposta 6
2.2 Categorização dos artigos selecionados por problema tratado 6
6.1 Parâmetros utilizados nos experimentos. 27
6.2 Resultados para instâncias do TSP 30
6.3 Resultados para instâncias sintéticas 32
6.4 Resultados para instâncias reais 32

1

Introdução

A geometria computacional é um campo de estudo rico e diversificado, que abrange uma ampla gama de problemas e aplicações. O foco central desse campo reside no desenvolvimento de algoritmos e estruturas de dados eficientes para a resolução de problemas geométricos [de Berg et al., 2000]. Além disso, a geometria computacional desempenha um papel crucial na resolução de problemas práticos em fabricação e otimização geométrica em espaços euclidianos, o que demanda o desenvolvimento de novas ferramentas e algoritmos eficientes capazes de lidar com a complexidade computacional em alta velocidade [Shamos and Hoey, 1975].

Dentro desse contexto, a cobertura geométrica constitui uma família de problemas de otimização de grande interesse e amplamente pesquisados no âmbito da geometria computacional. Esse tema tem sido objeto de estudo contínuo há várias décadas, refletindo sua importância e aplicabilidade [Friederich et al., 2023]. Um dos problemas centrais nessa área é o problema de cobertura de pontos, que é clássico e fundamental na geometria computacional. Este problema possui aplicações em diversas áreas, como logística, localização, visão computacional e aprendizado de máquina.

O problema investigado no presente trabalho é conhecido como Problema de Cobertura de Discos de Raio Fixo, uma importante questão dentro da geometria computacional. Em síntese, o PCDRF visa determinar o número mínimo de discos de raio fixo necessários para cobrir um dado conjunto de pontos. Diferentemente de outros problemas de cobertura, a localização dos centros dos discos é uma variável contínua, o que implica que eles podem ser posicionados em qualquer ponto do plano, e não apenas em um conjunto discreto de locais. Essa característica adiciona um nível de complexidade adicional, exigindo abordagens mais sofisticadas para a sua resolução eficaz.

1.1 Motivação

O Problema de Cobertura de Discos de Raio Fixo (PCDRF) possui uma ampla gama de aplicações, abrangendo redes sem fio, localização de instalações, robótica, processamento de imagens e aprendizado de máquina [Friederich et al., 2023]. Este problema desempenha um papel crucial na otimização da alocação de recursos e no planejamento de infraestrutura em cenários que exigem cobertura espacial.

O PCDRF oferece contribuições valiosas para problemas de localização de instalações. Uma aplicação típica, por exemplo, consiste em determinar o número mínimo de transmissores necessários para garantir cobertura total aos clientes em redes sem fio. Nesse caso, os clientes são modelados como pontos, e os transmissores como discos [Hu, 2013]. Essa abordagem permite projetar redes mais eficientes, equilibrando custo e desempenho.

Iniciativas de segurança pública também se beneficiam do PCDRF, especialmente na seleção de locais ideais para a instalação de câmeras de segurança. Os discos correspondem às áreas de monitoramento das câmeras, enquanto os pontos representam os locais que necessitam de vigilância. O objetivo é maximizar a cobertura, minimizando os custos, garantindo a segurança de espaços críticos como centros comerciais, praças públicas e bairros residenciais de alto risco.

No setor de logística e transporte, o PCDRF auxilia no planejamento estratégico da localização de centros de distribuição. Nesse cenário, os discos representam as áreas de atendimento de cada centro, enquanto os pontos correspondem aos destinos de entrega. Otimizando a localização dos centros de distribuição, as organizações podem minimizar os custos de transporte e armazenamento, garantindo entregas rápidas e eficientes.

Além disso, o PCDRF tem aplicações no contexto militar e de defesa, como no posicionamento de sistemas de defesa antimísseis para proteger locais estratégicos ou na colocação de transmissores de rádio para cobrir o maior número possível de vilarejos [Panov et al., 2022]. Esses casos destacam a utilidade do PCDRF na solução de questões críticas de segurança nacional.

Em resumo, o Problema de Cobertura por Discos de Raio Fixo funciona como uma estrutura versátil para lidar com uma ampla gama de desafios econômicos, sociais e técnicos. Sua relevância em diversos setores reforça a importância do desenvolvimento de novas abordagens e metodologias para ampliar sua aplicabilidade e eficácia [Friederich et al., 2023]. O estudo do PCDRF continua sendo uma área de significativo interesse acadêmico e prático.

1.2 Objetivo

Este trabalho tem como objetivo geral desenvolver uma meta-heurística para a resolução do Problema da Cobertura de Discos de Raio Fixo (PCDRF), visando avançar o estado da arte nessa área.

1.2.1 Objetivos Específicos

Para alcançar o objetivo geral proposto, foram estabelecidos os seguintes objetivos específicos:

- Revisar e identificar abordagens existentes na literatura para a resolução do PCDRF;
- Definir e caracterizar detalhadamente o problema, incluindo sua formulação matemática e restrições;
- Desenvolver uma meta-heurística para o PCDRF, considerando suas especificidades;
- Realizar experimentos computacionais e, por meio de análise estatística, avaliar o desempenho do método proposto em comparação com abordagens existentes.

1.3 Organização do Trabalho

No Capítulo 2, é apresentada uma revisão da literatura relacionada ao PCDRF. Os trabalhos analisados são classificados de acordo com as abordagens propostas (aproximação, heurística, meta-heurística) e os tipos específicos de problemas de cobertura tratados.

O Capítulo 3 fornece a base teórica para o estudo. São introduzidos os conceitos de problemas de otimização combinatória, explorada a noção de $\mathcal{NP}$-completude e discutidas estratégias comuns de solução. Apresentam-se técnicas de programação matemática, como programação linear e programação linear inteira, além de métodos aproximados, heurísticas e metaheurísticas.

No Capítulo 4, descreve-se a metodologia adotada, caracterizada como um estudo quantitativo e descritivo. Um fluxograma ilustra as etapas metodológicas. Além é realizado a caracterização do problema.

O Capítulo 6 apresenta os resultados da avaliação experimental do CMSA. São descritos o ambiente computacional e as ferramentas utilizadas, especificados os parâmetros configurados no algoritmo CMSA e detalhadas as instâncias de teste. Os resultados obtidos são analisados, evidenciando o desempenho superior do CMSA em comparação com métodos de referência.

Por fim, o Capítulo 7 sintetiza as principais conclusões, destacando a eficácia do CMSA na resolução do PCDRF. São discutidas as limitações do trabalho, como os desafios relacionados a instâncias de grande escala e a sensibilidade à calibração de parâmetros. Também são indicadas direções para pesquisas futuras.

2

Trabalhos Relacionados

Na literatura, é comum encontrar algoritmos aproximativos para o Problema da Cobertura de Discos de Raio Fixo (PCDRF). No entanto, esses algoritmos frequentemente apresentam uma elevada razão de aproximação, o que implica que, em certos casos, a solução gerada pode se distanciar consideravelmente da solução ótima.

Hochbaum and Maass [1985] introduziram a *shifting strategy*, uma técnica baseada no princípio de divisão e conquista. Essa abordagem divide o problema em faixas ou grades e resolve-o para uma divisão inicial do plano. Posteriormente, essas faixas são deslocadas várias vezes, criando novas subdivisões e, consequentemente, novas soluções. Ao final, todas as soluções obtidas são comparadas, e a melhor é selecionada. Esse método é um *Polynomial-Time Approximation Scheme* (PTAS) que opera em tempo $O\left(\ell^d(\ell\sqrt{d})^d(2n)^{d(\ell\sqrt{d})^d+1}\right)$, com um fator de aproximação de $\left(1+\frac{1}{\ell}\right)^d$. Em que, ℓ representa o número de deslocamentos, n é a quantidade de pontos, e d é o diâmetro dos discos.

Gonzalez [1991] aborda o problema de cobrir um conjunto de pontos em um espaço multidimensional com o menor número possível de hipercubos ortogonais de tamanho fixo. No artigo, o autor demonstra como adaptar algoritmos aproximativos para a cobertura por discos, estendendo os resultados para métricas euclidianas.

Fu et al. [2007] foca no PCDRF, apresentando um algoritmo com complexidade de tempo quase linear e uma razão de aproximação de 2,8334, o que significa que a solução é garantida em, no máximo, 2,8334 vezes o ótimo. Ghasemalizadeh and Razzazi [2012] investigam o problema de cobertura com discos limitados, uma variante do PCDRF onde o número de discos é restrito. Os autores propõem um PTAS com o objetivo de maximizar o número de pontos cobertos.

O problema de Cobertura com Discos Unitários (CDU), um caso particular do PCDRF, foi examinado por Liu and Lu [2014]. Os autores propõe um algoritmo de aproximação que restringe os centros dos discos a linhas paralelas, melhorando significativamente o tempo de execução e os requisitos de armazenamento.

Basappa et al. [2015] exploram duas variantes do problema CDU: o problema de cobertura com discos unitários discretos e o problema de cobertura de regiões retangulares.

Biniaz et al. [2017] propuseram um algoritmo simples e rápido de aproximação para o problema CDU, alcançando uma aproximação de 4 vezes em $O(n \log n)$. A adaptabilidade do algoritmo a outros espaços métricos e ao problema de cobertura com esferas unitárias em três dimensões também é discutida.

A variante multidimensional do problema, envolvendo esferas de raio unitário, é abordada em Dumitrescu et al. [2020]. Os autores consideram um algoritmo *online*, onde os pontos são revelados incrementalmente e devem ser cobertos à medida que aparecem.

Panov et al. [2022] examinam algoritmos para minimizar o número de discos de raio fixo necessários para cobrir um conjunto de pontos. O estudo propõe duas abordagens: uma que visa a localização que cobre o maior número de pontos descobertos e outra que foca na maximização da cobertura de pontos distantes. Testes comparativos em diferentes conjuntos de dados mostram que o segundo algoritmo, baseado na abordagem *MaxiMinMax*, frequentemente produz resultados superiores.

Viana [2022] apresenta a meta-heurística CMSA como uma solução para o Problema de Cobertura de Pontos com Discos Ponderados (PCPDP). O PCPDP consiste em determinar a configuração ótima de discos, cada um com custos e raios de cobertura distintos, para cobrir um conjunto de pontos em um plano, minimizando o custo total. Este problema difere notavelmente do PCDRF. No PCPDP, os discos são limitados em quantidade, possuem raios e custos variados, e são especificados como parte da entrada do problema. O objetivo é minimizar o custo total utilizando os discos fornecidos para garantir a cobertura completa dos pontos. Em contraste, o PCDRF busca identificar o número mínimo de discos, todos com um raio idêntico, necessário para cobrir todos os pontos.

Friederich et al. [2023] tratam do CDU. Os autores deste artigo implementam e comparam o desempenho prático de vários algoritmos CDU, incluindo um novo algoritmo de 7-aproximação chamado *FastCover*, que executa em $O(n)$. O método envolve discretizar os pontos do plano associando cada ponto (x_i, y_i) a inteiros a, b, que formam uma grade de tamanho $\sqrt{2}$, onde $a = \left\lfloor \frac{x_i}{\sqrt{2}} \right\rfloor$ e $b = \left\lfloor \frac{y_i}{\sqrt{2}} \right\rfloor$. A partir dessa discretização, é possível determinar um disco unitário $D(a,b)$ que circunscreve a célula $\sigma(a,b)$ correspondente.

Em outra contribuição, Wang et al. [2023] propuseram um algoritmo prático denominado *Hierarchical Service Facility (HSF)*, voltado para a otimização da distribuição de serviços públicos. O problema associado é formulado como um problema de *Hierarchical Discrete Unit Disk Cover*, resolvido por meio de um algoritmo guloso.

Chen et al. [2024] propuseram uma heurística denominada Redução Dinâmica (*DR*) para o PCDRF. O método *DR* funciona selecionando aleatoriamente um ponto como o primeiro centro de disco. Os centros subsequentes são escolhidos com base na maior distância $D_{p,x}$, de forma que os centros permaneçam esparsos. Em seguida, calcula-se o disco de maior raio. Se o raio desse maior círculo for menor ou igual a r, um dos centros é removido, e o processo continua

até encontrar uma configuração na qual não seja mais possível remover centros sem que o raio do menor disco ultrapasse r.

Por fim, Uma abordagem meta-heurística baseada em *simulated annealing* para maximizar a cobertura de pontos com um número fixo de discos é apresentada por Tomova et al. [2024].

Abordagem	**Referências**
Aproximativo	Friederich et al. [2023], Gonzalez [1991], Liu and Lu [2014], Hochbaum and Maass [1985], Biniaz et al. [2017], Dumitrescu et al. [2020], Fu et al. [2007], Ghasemalizadeh and Razzazi [2012], Basappa et al. [2015]
Heurística	Chen et al. [2024], Wang et al. [2023], Panov et al. [2022]
Meta-heurística	Tomova et al. [2024], Viana [2022]

Tabela 2.1: Classificação dos artigo pela a abordagem proposta

Na Tabela 2.1, os artigos revisados são classificados em três grupos distintos: algoritmos de aproximação, algoritmos heurísticos e algoritmos meta-heurísticos. Da mesma forma, a Tabela 2.2 categoriza os problemas abordados em três categorias: Raio Unitário, que analisa casos em que o raio do disco de cobertura é fixado em um valor unitário; Raio Fixo, que considera cenários nos quais o raio pode assumir qualquer valor real positivo; e Discos Limitados, que aborda problemas em que o número de discos de cobertura é restringido por um limite pré-definido.

Problema	**Referências**
Raio Unitário	Friederich et al. [2023], Liu and Lu [2014], Dumitrescu et al. [2020], Biniaz et al. [2017], Basappa et al. [2015], Wang et al. [2023]
Raio Fixo	Chen et al. [2024], Gonzalez [1991], Hochbaum and Maass [1985], Fu et al. [2007], Panov et al. [2022], Viana [2022]
Disco limitado	Ghasemalizadeh and Razzazi [2012], Tomova et al. [2024]

Tabela 2.2: Categorização dos artigos selecionados por problema tratado

3

Fundamentação teórica

Muitos problemas práticos e teóricos em ciência da computação, engenharia e diversas outras áreas podem ser formulados como problemas de otimização combinatória. Um exemplo disso é a otimização do uso de recursos escassos, como tempo, dinheiro, mão de obra e materiais, aplicável a setores como manufatura, transporte, saúde, finanças e logística. Para resolver esses problemas, é necessário maximizar ou minimizar uma determinada função com uma ou várias variáveis, garantindo que todas as restrições sejam satisfeitas.

Um problema de otimização pode ser descrito de forma genérica da seguinte maneira:

$$\begin{aligned} \max \quad & f(x) \\ \text{sujeito a} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m. \end{aligned}$$

em que $f : \mathbb{R}^n \to \mathbb{R}$ e $g_i : \mathbb{R}^n \to \mathbb{R}$ para todo $i = 1, \ldots, m$. Note que, neste caso, todas as funções são contínuas. Uma solução é dita viável se ela satisfaz todas as restrições, ou seja, todas as inequações $g_i(x) \leq 0$ são válidas. A função objetivo f associa a toda solução um número real que indica o valor ou a qualidade da solução. Insta salientar que grande parte dos problemas de otimização combinatória são classificados como $\mathcal{NP}$-difíceis na literatura e, até o momento, não se conhecem algoritmos que os resolvam em tempo polinomial [Souza, 2024].

Um exemplo de famoso de um problema de otimização é o Problema do Caixeiro Viajante (PCV). O PCV é visa é encontrar a menor rota que um caixeiro viajante deve percorrer para visitar um conjunto de cidades e retornar à cidade de origem, passando por cada cidade exatamente uma vez. O espaço de busca para o PCV com n cidades é de $(n-1)!$ possíveis rotas, considerando que a rota pode ser iniciada em qualquer cidade e a direção pode ser invertida sem alterar a solução. Por exemplo, para $n = 10$ cidades, o número de rotas possíveis é 362.880, enquanto para $n = 20$, esse número salta para aproximadamente $1{,}216 \times 10^{18}$. Resolver o PCV por enumeração torna-se inviável rapidamente à medida que n aumenta, pois o tempo de computação cresce de forma fatorial. Mesmo utilizando supercomputadores, resolver o PCV por enumeração para $n = 50$ cidades levaria um tempo impraticável, da ordem de bilhões de anos.

Outro problema é o Problema da Mochila. Neste problema, você tem uma mochila com capacidade limitada e um conjunto de itens, cada um com um peso e um valor. O objetivo é selecionar os itens que maximizem o valor total carregado na mochila, sem exceder sua capacidade. O espaço de busca para o Problema da Mochila com n itens é de 2^n combinações possíveis, já que cada item pode estar ou não na mochila. Por exemplo, para $n = 30$ itens, existem 1,073,741,824 possíveis combinações, e para $n = 50$, o número de combinações sobe para $1{,}125 \times 10^{15}$. A abordagem de enumeração se torna rapidamente impraticável, pois o tempo necessário para avaliar todas as combinações cresce exponencialmente com o número de itens. Resolver o problema da mochila por enumeração para $n = 100$ itens exigiria mais combinações do que o número de átomos no universo observável, tornando a solução exata inviável.

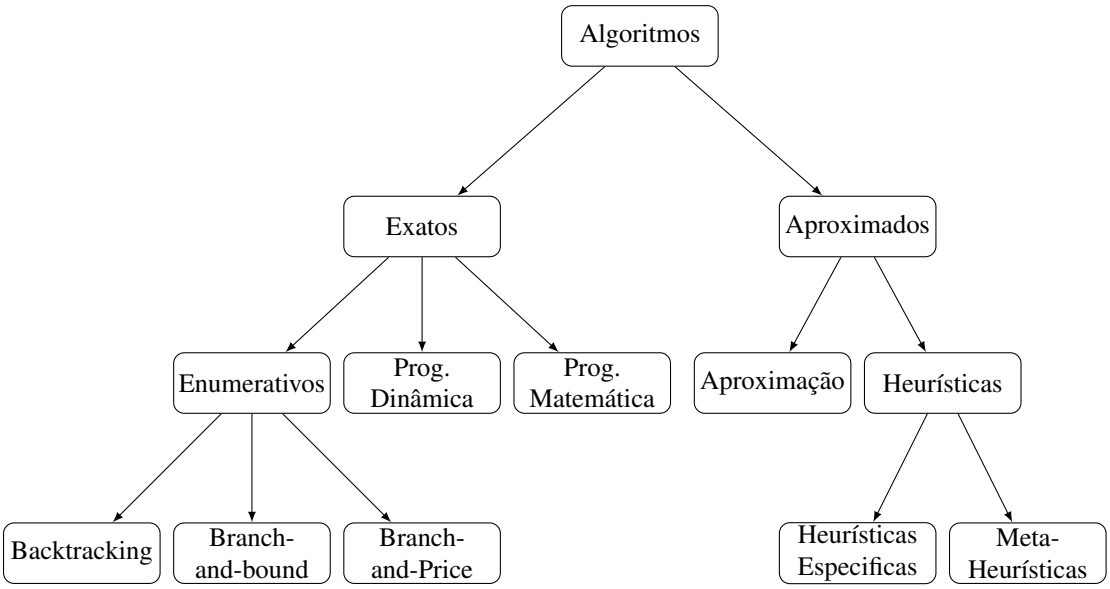

Figura 3.1: Exemplos de abordagens.
Fonte: Elaborado pelo autor.

As abordagens para resolver esses problemas podem ser divididas em duas categorias: exatas e aproximadas (ver figura 3.1). As abordagens exatas garantem a obtenção da melhor solução possível, enquanto as abordagens aproximadas não asseguram a descoberta da solução ótima, mas buscam encontrar soluções em tempo computacional viável.

3.1 NP-completude

Entende-se como problema de decisão um problema computacional cuja solução pode ser expressa como uma função booleana, retornando verdadeiro ou falso, dependendo da entrada. Nesse contexto, há milhares de problemas desse tipo para os quais, apesar de se conhecer um algoritmo eficiente para verificar soluções, ainda não se conhece nenhum algoritmo eficiente que os resolva[1], sem que se tenha provado de forma conclusiva que tais algoritmos não possam

[1]Considera um algoritmo eficiente quando sua complexidade é um polinômio no tamanho de sua entrada.

existir.

Nesse contexto, surge como questão fundamental o Problema P vs. NP, que consiste em determinar se todo problema computacional cuja solução pode ser verificada eficientemente também pode ser resolvido eficientemente. Trata-se de uma das questões abertas mais proeminentes da matemática e ciência da computação [Garey and Johnson, 1979].

Para definir as classes $\mathcal{P}$ e $\mathcal{NP}$ mencionadas, é fundamental estabelecer um modelo computacional formal. Uma das razões para essa necessidade é garantir a robustez dessas classes, ou seja, assegurar que suas definições sejam independentes do modelo computacional específico utilizado. O modelo computacional padrão empregado na teoria da computabilidade, e consequentemente na definição de $\mathcal{P}$ e $\mathcal{NP}$, é a Máquina de Turing [Turing, 1936]. Para evitar formalismo segue as definições informais das principais classes da teoria da NP-completude.

- **Classe $\mathcal{P}$:**
  - Consiste no conjunto de todos os problemas de decisão que podem ser resolvidos por um algoritmo em tempo polinomial.
- **Classe $\mathcal{NP}$:**
  - Refere-se ao conjunto de problemas de decisão para os quais, se a resposta for "sim", existe uma prova ou certificado que pode ser verificado em tempo polinomial por um algoritmo.
- **Problemas $\mathcal{NP}$-Completos:**
  - Um problema é dito $\mathcal{NP}$-completo se satisfaz duas condições:
    1. Está em $\mathcal{NP}$.
    2. Todo problema em $\mathcal{NP}$ pode ser reduzido a ele em tempo polinomial.
- **Problemas $\mathcal{NP}$-Difíceis:**
  - Um problema é dito $\mathcal{NP}$-difícil se satisfaz a segunda condição para $\mathcal{NP}$-completude: Todo problema em $\mathcal{NP}$ pode ser reduzido a ele em tempo polinomial.

Partindo desses conceitos, pode-se dizer que o Problema P vs. NP consiste em determinar se a classe $\mathcal{NP}$ é igual á classe $\mathcal{P}$. A maioria dos cientistas da computação acredita que $\mathcal{P} \neq \mathcal{NP}$, não existe, contudo, uma prova formal para sustentar essa conjectura [Cook, 2000].

O primeiro problema a ser provado como $\mathcal{NP}$-completo foi o Problema de Satisfatibilidade Booleana (PTB), demonstrado por Stephen Cook em seu artigo de 1971 [Cook, 1971], *The Complexity of Theorem Proving Procedures*. Foi justamente a prova de Cook de que o PTB é $\mathcal{NP}$-completo que inaugurou a teoria da $\mathcal{NP}$-completude.

A resolução desta questão tem implicações profundas para a ciência da computação e áreas afins. Diversos problemas de otimização em áreas como logística, planejamento de produção

e bioinformática são $\mathcal{NP}$-completos. A descoberta de um algoritmo eficiente para resolver esse tipo de problema revolucionaria essas áreas, permitindo a obtenção de soluções ótimas para problemas atualmente considerados intratáveis. Por exemplo, a segurança da internet, que dá suporte a transações financeiras e comunicações confidenciais, depende de pressupostos de complexidade computacional, como a dificuldade de fatorar números grandes ou de quebrar o padrão de criptografia DES (*Data Encryption Standard*) [Cook, 1971]. Se P fosse igual a NP, a criptografia poderia ser quebrada em minutos, comprometendo a segurança da internet.

3.2 Abordagens Exatas

Métodos exatos garantem a otimalidade. Nesta seção serão abordados alguns métodos exatos.

3.2.1 Programação matemática

A programação matemática é uma área fundamental da otimização, que busca encontrar a melhor solução para um problema dentre um conjunto de alternativas viáveis. Essa melhor solução é definida por uma função objetivo, que deve ser maximizada ou minimizada, respeitando um conjunto de restrições.

Diferentes modelos de Programação Matemática podem ser usados para modelar e resolver diferentes problemas de otimização. Basicamente, eles podem ser divididos em Linear e Não Linear; ou em modelos de Programação Contínua, Mista e Inteira.

3.2.2 Problema de Programação Linear

Um Problema de Programação Linear pode ser definido da seguinte forma:

$$\max \quad \sum_{i=1}^{n} c_i x_i \tag{1}$$

$$\text{s.a.} \quad \sum_{i=1}^{n} a_{ij} x_i \leq b_j, \quad j = 1,2,\ldots,m \tag{2}$$

$$x_i \geq 0, \quad i = 1,2,\ldots,n \tag{3}$$

em que c_i, a_{ij} e b_j são dados (números reais) e x_i representa as variáveis de decisão. A função linear a ser maximizada (1) é denominada função objetivo. As restrições (2) determinam o domínio do problema. Por fim, as restrições (3) são conhecidas como restrições de não negatividade ou restrições triviais.

3.2.3 Problema de Programação Linear Inteira

Um problema de Programação Inteira (PI) pode ser representado da seguinte forma genérica:

$$\max \quad c^T x \tag{4}$$
$$\text{s.a.} \quad Ax \leq b \tag{5}$$
$$x \in \mathbb{Z}^n \tag{6}$$

Em que x é um vetor de variáveis de decisão. O objetivo é maximizar a função $c^T x$, enquanto que as variáveis devem atender ao conjunto de restrições lineares (5). A dificuldade em resolver um problema de PI da forma (4) – (6) encontra-se nas restrições de integralidade (6), que tornam o problema $\mathcal{NP}$-difícil Karp [1972].

3.3 Branch-and-Bound

O *branch-and-bound* é um método de enumeração inteligente que utiliza estratégias de poda para evitar percorrer todo o espaço de busca. O algoritmo funciona particionando recursivamente o espaço de busca em subproblemas (*branching*) e eliminando soluções inviáveis ou não promissoras (*bounding*). Cada nó em uma árvore representa um subproblema, e apenas os subproblemas considerados promissores são explorados.

No contexto da Programação Linear Inteira (PLI), o *branch-and-bound* é aplicado da seguinte forma [Rothlauf, 2011]:

1. Inicialmente, relaxa-se o problema original e resolve-se por programação linear, obtendo um limite (superior ou inferior) para o valor ótimo da função objetivo.

2. Seleciona-se uma variável de decisão x_i que não é inteira e ramifica-se em dois subproblemas adicionando as restrições $x_i \leq \lfloor x_i \rfloor$ e $x_i \geq \lceil x_i \rceil$. Esses subproblemas são resolvidos recursivamente.

3. Um subproblema é podado se:

   - Todas as variáveis de decisão são inteiras e o valor da função objetivo melhora o limite atual.
   - O subproblema não possui solução viável (poda por inviabilidade).
   - A função objetivo do subproblema é pior do que o limite atual (poda por limite).

4. O algoritmo continua expandindo e podando subproblemas até que todas as possibilidades tenham sido exploradas ou eliminadas.

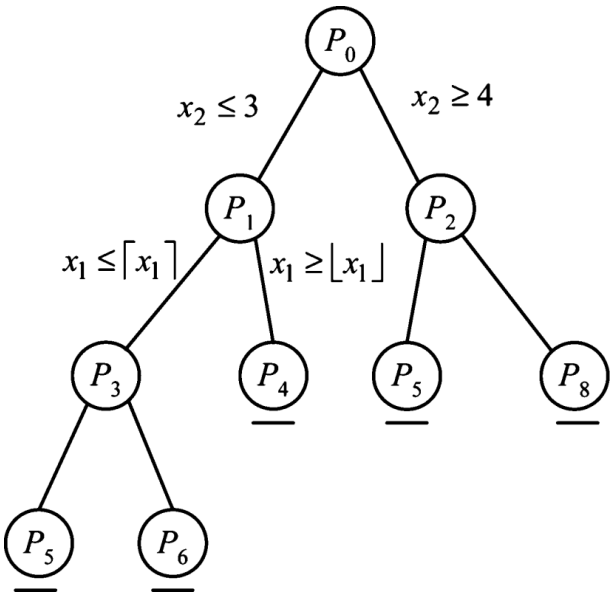

Figura 3.2: Árvore de enumeração do *branch-and-bound*.
Fonte: Yu and Lim [2007].

A implementação do *branch-and-bound* pode variar quanto à estratégia de ramificação, ordem de exploração e regras de poda. Além de PLI, o método também é aplicável a problemas não lineares.

Para facilitar sua aplicação em diversos problemas de otimização, foram desenvolvidos pacotes de software como o CPLEX[2], Gurobi[3] e LINDO[4].

3.4 Abordagens Aproximadas

Os algoritmos aproximados são divididos em duas categorias: algoritmos de aproximação e heurísticas. Os algoritmos de aproximação fornecem garantias quanto à qualidade das soluções encontradas, embora, na maioria das vezes, a qualidade das aproximações esteja distante das soluções ótimas [Talbi, 2009]. Por outro lado, as heurísticas não oferecem nenhuma garantia em relação à qualidade das soluções.

As heurísticas podem ser subdivididas em heurísticas específicas e meta-heurísticas. As heurísticas específicas são abordagens ad hoc para um determinado problema, enquanto as meta-heurísticas possuem caráter geral.

3.4.1 Meta-heurística

Meta-heurísticas são métodos de otimização que buscam encontrar boas soluções para problemas de otimização, sem garantir a otimalidade. Elas são algoritmos de propósito geral, com

[2] https://www.ibm.com/analytics/cplex-optimizer
[3] https://www.gurobi.com/
[4] https://www.lindo.com/

mecanismos para evitar ficarem presas em ótimos locais [Souza, 2024](ver 3.3).

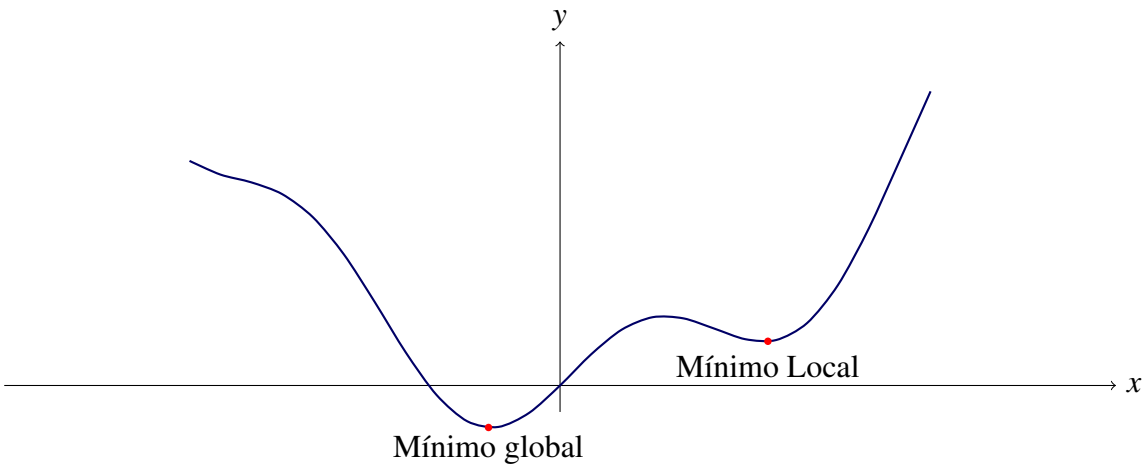

Figura 3.3: Representação de um ótimo local e um ótimo global.
Fonte: Elaborado pelo autor.

As meta-heurísticas diferem entre si principalmente pelo método utilizado para evitar essas armadilhas dos ótimos locais. Elas são classificadas em duas categorias, de acordo com o princípio adotado para explorar o espaço de soluções: busca local e busca populacional [Souza, 2024].

Nesse contexto, destacam-se as meta-heurísticas de busca local: *Greedy Randomized Adaptive Search Procedure* (GRASP), *Variable Neighborhood Descent* (VND), *Reactive Search Optimization* (RSO) e *Large Neighborhood Search* (LNS); e as populacionais: *Particle Swarm Optimization* (PSO), *Differential Evolution* (DE), *Bee Colony Optimization* e *Cuckoo Search*.

Um tipo bem-sucedido de meta-heurística são as meta-heurísticas híbridas, que combinam meta-heurísticas com diferentes métodos. Normalmente, quatro tipos distintos de combinações são consideradas:

1. Meta-heurísticas com outras meta-heurísticas;
2. Meta-heurísticas com métodos de programação matemática;
3. Meta-heurísticas com técnicas de aprendizado de máquina.

Nos últimos anos, o interesse por meta-heurísticas híbridas cresceu significativamente no domínio da otimização. Os algoritmos híbridos têm se destacado ao proporcionar os melhores desempenhos em diversos problemas de otimização, tanto clássicos quanto aplicados ao mundo real [Talbi, 2009]. A integração de diferentes técnicas resulta em algoritmos cada vez mais robustos e eficientes.

As meta-heurísticas que combinam métodos de programação matemática são chamadas de matheurísticas [Maniezzo et al., 2009, Boschetti and Maniezzo, 2022]. Elas integram técnicas de meta-heurísticas com métodos de Programação Matemática , aproveitando o poder da

modelagem matemática para guiar a busca por soluções de maneira mais eficiente. Ao incorporar componentes de MP, as matheurísticas superam algumas das limitações das meta-heurísticas tradicionais, proporcionando soluções mais precisas e robustas em problemas em que uma abordagem puramente heurística seria insuficiente.

3.4.2 CMSA

O algoritmo CMSA (*Construct, Merge, Solve and Adapt*), proposto por Christian Blum [Blum et al., 2016], é uma matheurística projetada para resolver problemas de otimização combinatória. A essência do CMSA reside na aplicação iterativa de métodos exatos, como resolvedores de Programação Linear Inteira (ILP), a subinstâncias derivadas das instâncias originais do problema.

O CMSA opera em um ciclo de quatro etapas principais: *construct*, *merge*, *solve* e *adapt*. Para aplicar o CMSA a um problema específico, é fundamental definir o conjunto C de componentes da solução, que são os elementos que compõem as soluções viáveis para o problema em questão. A definição precisa de C depende da natureza do problema, a qual geralmente sugere uma escolha natural para esses componentes.

Parâmetros do CMSA

O CMSA utiliza vários parâmetros que influenciam seu comportamento e desempenho. Os principais são:

- **Número de Construções de Solução por Iteração (n_a)**: Define quantas soluções são construídas probabilisticamente em cada iteração do algoritmo.
- **Idade Máxima ($\mathbf{age_{max}}$)**: Determina o número máximo de iterações consecutivas que um componente de solução pode permanecer na subinstância C' sem estar presente na solução ILP.
- **Limite de Tempo ($t_{\mathbf{limit}}$)**: Especifica o tempo máximo, em segundos, que o CMSA tem para encontrar a melhor solução.
- **Parâmetros do Resolvedor ILP**: Parâmetros específicos do resolvedor ILP, como ênfase em heurísticas, uso de *warm start* e critério de parada, que podem ser ajustados para otimizar o desempenho.

Descrição do Algoritmo

O pseudocódigo do CMSA é apresentado no Algoritmo 1.

Algoritmo 1 CMSA(C, n_a, $\text{age}_{\max}$, t_{limit})

$S_{\text{bsf}} \leftarrow \emptyset$; $C' \leftarrow \emptyset$
while tempo limite não for excedido **do**
  for $i \leftarrow 1$ **to** n_a **do**
    $S \leftarrow \text{ProbabilisticSolution}(C)$
    for all $c \in S$ **e** $c \notin C'$ **do**
      $\text{age}[c] \leftarrow 0$; $C' \leftarrow C' \cup \{c\}$
    end for
  end for
  $S_{\text{ILP}} \leftarrow \text{SolveSubinstance}(C', t_{\text{ILP}})$
  if $F(S_{\text{ILP}}) < F(S_{\text{bsf}})$ **then**
    $S_{\text{bsf}} \leftarrow S_{\text{ILP}}$
  end if
  Adapt(C', S_{ILP}, $\text{age}_{\max}$)
end while
Retornar S_{bsf}

O CMSA executa um ciclo iterativo que se repete até que um limite de tempo seja atingido. Cada iteração consiste nas seguintes etapas:

1. **Construção de Soluções**:
   - O algoritmo gera n_a soluções não necessariamente viáveis utilizando uma heurística probabilística.
2. **Mesclagem**:
   - Os componentes das soluções geradas são adicionados à subinstância C'.
3. **Resolução da Subinstância**:
   - Um resolvedor exato, como um resolvedor ILP, é aplicado à subinstância C' por um tempo máximo de t_{ILP} segundos.
   - O resolvedor busca a melhor solução (S_{ILP}) dentro do espaço de busca delimitado por C'.
4. **Adaptação da Subinstância**:
   - A subinstância C' é atualizada com base na solução S_{ILP} encontrada pelo resolvedor.
   - Componentes de solução que não contribuem para S_{ILP} têm sua idade incrementada.
   - Componentes com idade superior a $\text{age}_{\max}$ são removidos de C' para evitar o crescimento excessivo da subinstância.

Com esse ciclo iterativo, o CMSA busca melhorar continuamente a qualidade das soluções, combinando a exploração aleatória do espaço de soluções com a exploração intensiva de subinstâncias promissoras por meio de métodos exatos.

4

Metodologia

A pesquisa caracteriza-se como um estudo descritivo de natureza quantitativa, fundamentado em métodos de pesquisa bibliográfica e experimental. A metodologia adotada seguiu uma sequência lógica de etapas, representada no Fluxograma da Figura 4.1, detalhando desde a revisão bibliográfica até a análise dos resultados obtidos.

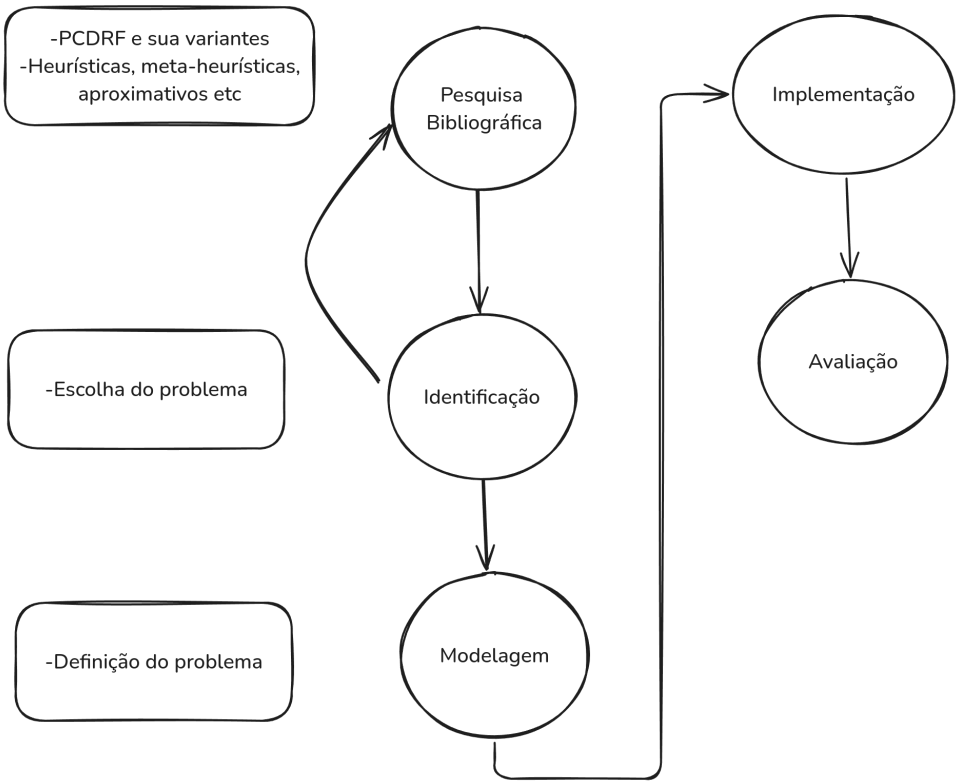

Figura 4.1: Etapas metodológicas.

O fluxograma ilustra o fluxo de atividades metodológicas que foram essenciais para a construção e validação do modelo proposto. A seguir, cada etapa é detalhada para esclarecer sua importância e o papel que desempenha na estrutura da pesquisa:

- **Pesquisa Bibliográfica**: Realizou-se uma extensa revisão da literatura nas principais bases de dados acadêmicas, incluindo *Scopus*, *Web of Science*, *Lens* e *IEEE*, além de periódicos especializados em geometria computacional. Esta etapa teve o objetivo de identificar abordagens atuais e lacunas na resolução do Problema de Cobertura de Discos de Raio Fixo (PCDRF), conforme discutido na Seção 2.
- **Identificação de Abordagens**: Com base na pesquisa bibliográfica, as abordagens existentes para resolver o PCDRF foram catalogadas e analisadas. Identificou-se a ausência de meta-heurísticas específicas e limitações nas heurísticas convencionais. Essa análise orientou o desenvolvimento da meta-heurística proposta.
- **Modelagem do Problema**: O problema foi caracterizado e formulado matematicamente, considerando suas restrições e parâmetros específicos.
- **Implementação do Algoritmo**: A meta-heurística CMSA foi então desenvolvida e implementada. Este processo incluiu desde a programação do algoritmo até os ajustes necessários para garantir sua aplicabilidade ao PCDRF.
- **Avaliação Experimental**: Por fim, testes experimentais foram realizados para verificar o desempenho da CMSA em comparação com métodos existentes. A análise dos resultados, descrita em seções posteriores, permite avaliar a eficácia e eficiência da abordagem proposta.

4.1 Caracterização do problema

O Problema de Cobertura de Discos de Raio Fixo é um problema bem conhecido em otimização combinatória. Este problema consiste em, dado um conjunto de pontos no plano, encontrar o menor conjunto possível de discos que cubra todos esses pontos [Hochbaum and Maass, 1985]. Esse problema é conhecido por ser $\mathcal{NP}$-difícil [Fowler et al., 1981].

PROBLEMA DE COBERTURA DE DISCO DE RAIO FIXO

ENTRADA: Um conjunto finito de pontos P no plano euclidiano $\mathbb{E}^2$ e um número real positivo r.

OBJETIVO: Determinar o conjunto mínimo D de discos de raio r, incluindo suas localizações, tal que todos os pontos de P estejam contidos em pelo menos um disco de D.

A dificuldade do problema reside em encontrar uma solução ótima em um espaço de busca contínuo, o que torna a enumeração de todas as soluções possíveis computacionalmente inviável. O problema pode ser modelado matematicamente, com a formulação não linear proposto por Chen et al. [2024]:

$$\min \quad k \tag{1}$$

$$\text{s.a.} \quad k \in \{1,\ldots,n\} \tag{2}$$

$$\sum_{j=1}^{k} z_{ij} \geq 1, \quad i = 1,\ldots,n \tag{3}$$

$$d(p_i,c_j)z_{ij} \leq r, \quad i = 1,\ldots,n, \quad j = 1,\ldots,k \tag{4}$$

$$z_{ij} \in \{0,1\}, \quad i = 1,\ldots,n, \quad j = 1,\ldots,k \tag{5}$$

em que

- k é o número de discos;
- z_{ij} é uma variável binária tal que $z_{ij} = 1$ se o ponto i for coberto pelo disco j, caso contrário, $z_{ij} = 0$;
- $p_i = (a_i, b_i)$ representa as coordenadas do ponto i;
- $c_j = (x_j, y_j)$ é o centro do disco j;
- $d(p_i, c_j)$ é a distância euclidiana entre o ponto i e o centro do disco j;
- r é o raio do disco.

O modelo busca minimizar o número de discos necessários para cobrir todos os pontos. A função objetivo na equação (1) visa minimizar a quantidade de discos k. As restrições (2) define que k deve ser um número inteiro, variando entre 1 e n, onde n é o número total de pontos. As restrições (3) assegura que cada ponto i seja coberto por pelo menos um disco j, pois, para cada ponto, a soma das variáveis binárias z_{ij} ao longo dos discos deve ser maior ou igual a 1. As restrições (4) garante que se um ponto é associado a um disco ($z_{ij} = 1$), então a distância euclidiana $d(p_i, c_j)$ entre um ponto i e o centro do disco j não exceda o raio r do disco, assegurando assim que o ponto esteja dentro da área de cobertura do disco. Finalmente, as restrições (5) especifica que z_{ij} é uma variável binária, que assume o valor 1 se o ponto i for associado j, e 0 caso contrário.

A entrada para o problema é ilustrada na Figura 4.2, onde os pontos estão dispostos no plano euclidiano. As Figuras 4.3 e 4.4 apresentam duas soluções potenciais. A solução apresentada na Figura 4.3 utiliza quatro discos. É importante notar que, embora alguns pontos sejam cobertos por vários discos, isso não invalida a solução, pois o problema exige apenas que todos os pontos sejam cobertos. Por outro lado, a solução mostrada na Figura 4.4 emprega apenas um único disco, tornando-se uma solução mais otimizada ao atingir o objetivo de minimizar o número de discos usados.

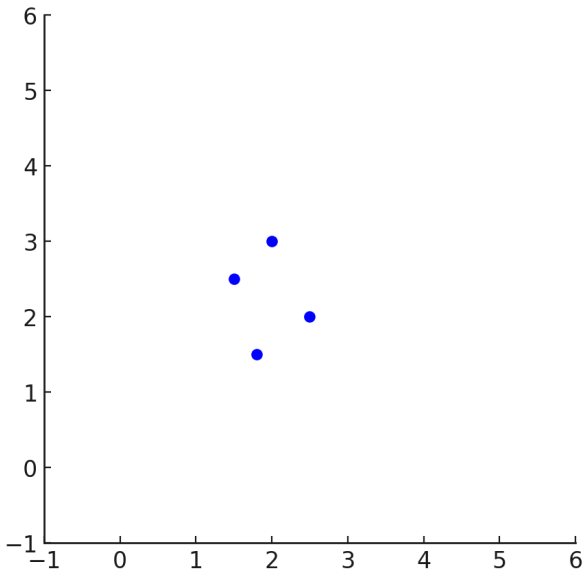

Figura 4.2: Exemplo de entrada

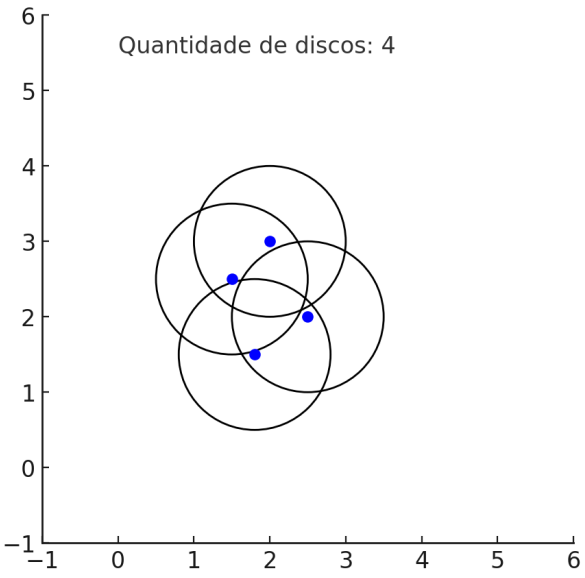

Figura 4.3: Solução viável

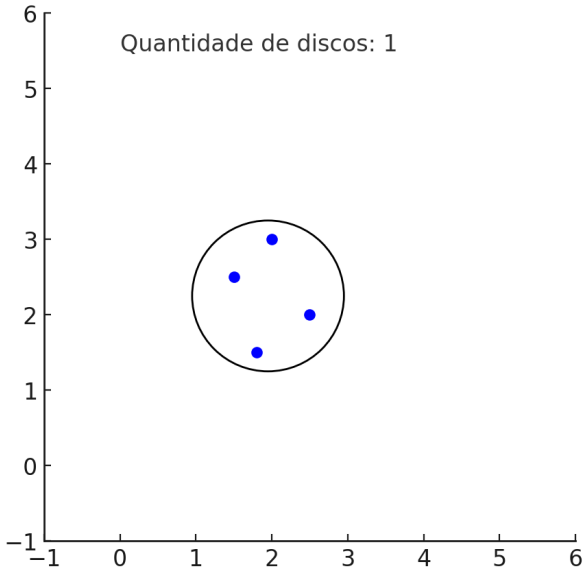

Figura 4.4: Solução ótima

5

CMSA

O CMSA (*Construct, Merge, Solve and Adapt*) é uma *matheuristic* (meta-heurística híbrida). Sua ideia principal consiste na aplicação iterativa de uma abordagem exata, como um resolvedor de programação linear inteira, a subinstâncias das instâncias originais do problema a serem resolvidas. Em cada iteração, o CMSA gera probabilisticamente várias soluções para a instância do problema abordado. Os componentes dessas soluções são adicionados a uma subinstância, denotada como C', que é então passada para um resolvedor exato que fornece a melhor solução encontrada. Com base na solução encontrada, a subinstância incumbente é adaptada. Componentes de solução aparentemente inúteis são removidos de C' para não prejudicar o desempenho do resolvedor na próxima iteração.

5.1 CMSA Aplicado ao PCDRF

Define-se o CMSA com os conjuntos C, C', S e a função $f(S)$ de avaliação. C define todos os componentes que podem fazer parte de uma solução para o problema, C' é um subconjunto de C que é iterativamente expandido e adaptado, S é um subconjunto de C que representa uma solução para o problema, $f(S)$ é uma função que avalia a qualidade de uma solução.

Um componente é definido como um par ordenado $c = (d, PC) \in C$, em que d representa o centro do disco e PC o conjunto de pontos cobertos pela a componente c. C engloba todas as componentes que podem ser utilizadas para formar as soluções do PCDRF. Uma solução válida $S \subseteq C$ para o PCDRF consiste em um subconjunto de componentes, em que, para cada ponto $p \in P$, existe ao menos um componente de C que cobre este ponto.

O conjunto S consiste nos componentes gerados na iteração atual e representa uma solução viável para o problema original, em que cada posição é coberta por pelo menos um componente.

A função

$$f : S \to \mathbb{R}$$

é definida como a cardinalidade do conjunto S, ou seja, a quantidade de discos usados para cobrir os pontos. Matematicamente:

$$f(S) = |S|$$

em que $|S|$ representa o número de elementos em S.

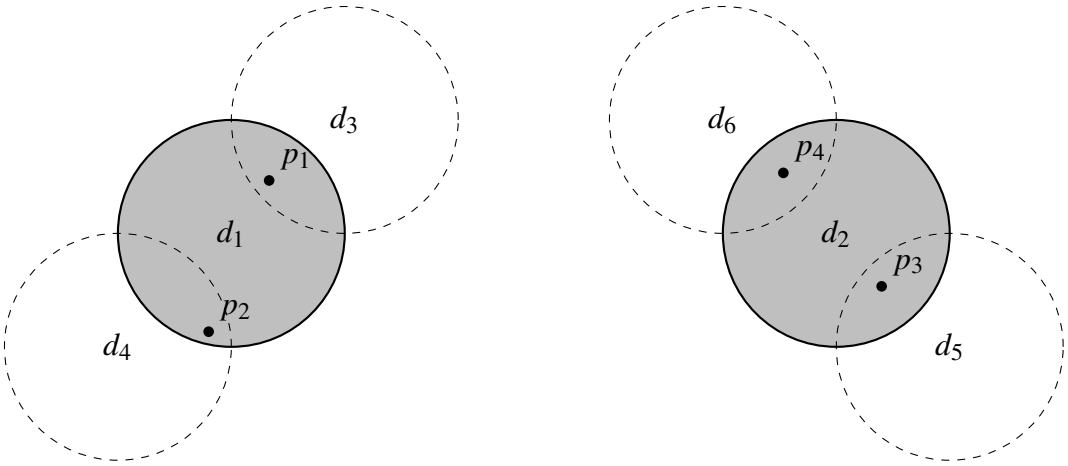

Figura 5.1: CMSA aplicado ao PCDRF
Fonte: Elaborado pelo autor.

Por exemplo, na Figura 5.1, tem-se $P = \{p_1, p_2, p_3, p_4\}$, um conjunto de pontos, e $C' = \{c_1, c_2, c_3, c_4, c_5, c_6\}$, uma subinstância. As componentes são:

$$\begin{aligned}
c_1 &= (d_1, \{p_1, p_2\}),\\
c_2 &= (d_2, \{p_4, p_3\}),\\
c_3 &= (d_3, \{p_1\}),\\
c_4 &= (d_4, \{p_2\}),\\
c_5 &= (d_5, \{p_3\}),\\
c_6 &= (d_6, \{p_4\}).
\end{aligned}$$

$S = \{c_1, c_2\}$ é uma solução do problema, pois cobre todos os pontos, com o valor da solução $F(S) = 2$, correspondente à quantidade de componentes.

5.2 Construct

A etapa *Construct* é o ponto de partida de cada iteração no algoritmo CMSA. O principal objetivo desta etapa é gerar múltiplas soluções iniciais para o PCDR. Nela, utiliza-se o método probabilístico `ProbabilisticSolution` para construir essas soluções. A geração de múltiplas soluções, utilizando métodos aleatórios, contribui para a diversificação da busca, explorando diferentes regiões do espaço de soluções. As soluções construídas nesta etapa fornecem os

componentes que serão utilizados para formar a subinstância C', que será posteriormente resolvida pelo resolvedor exato na etapa *Solve*.

A etapa Construct é implementada pelo o Algoritmo 2. Este algoritmo inicia com o conjunto total de pontos e utiliza uma abordagem de divisão e conquista para criar as componentes.

Inicialmente, o algoritmo define um conjunto vazio S para armazenar os componentes resultantes e inicializa o `Quadrante`, que é definido usando os valores de borda dos pontos, com todos os pontos. Uma fila é utilizada para gerenciar os quadrantes. Enquanto a fila não estiver vazia (todos os pontos não forem cobertos), o algoritmo processa o quadrante no início da fila.

Se o `Quadrante` contém apenas um ponto, um componente c é criado utilizando a função `CreateComponente`, e adicionado ao conjunto S. Se a diagonal do `Quadrante` for menor ou igual a $2r$, em que r é o raio fixo dos discos, então todos os pontos estão suficientemente perto. Nesse caso, um componente c é criado e adicionado a S.

Se nenhuma das condições acima for satisfeita, isto indica que o `Quadrante` é muito grande e que os pontos estão muito dispersos para serem cobertos por um único disco. O algoritmo, então, divide o `Quadrante` aleatoriamente em quatro subquadrantes menores q_1, q_2, q_3 e q_4, que são adicionados à filar. No final, o algoritmo retorna o conjunto S, que contém todos os componentes criados.

O Algoritmo 3 cria os componentes utilizando o ponto médio do `Quadrante` como o centro do disco e busca os pontos, organizados em uma *kd-tree*, que estão a uma distância menor ou igual a r deste centro.

Algoritmo 2 ProbabilisticSolution(`Pontos`, `r`)

$S \leftarrow \{\}$
`Quadrante` $\leftarrow$ `Pontos`
Adiciona `Quadrante` à `fila`
while `fila` não está vazia **do**
    Retira `Quadrante` da `fila`
    if `Quadrante` tem um ponto **then**
        $c \leftarrow$ `CreateComponente`(`Quadrante`, `r`)
        $S \leftarrow S \cup \{c\}$
    else if $Quadrante_{\text{diagonal}} \leq 2 \times$ `r` **then**
        $c \leftarrow$ `CreateComponente`(`Quadrante`, `r`)
        $S \leftarrow S \cup \{c\}$
    else
        Particiona `Quadrante` aleatoriamente em q_1, q_2, q_3, q_4
        Adiciona q_1, q_2, q_3, q_4 à `fila`
    end if
end while
return S

Algoritmo 3 CreateComponente(`Quadrante`, `r`)

1: $d \leftarrow$ Ponto médio do `Quadrante`
2: $CP \leftarrow$ Pontos que estão a distancia `r` de d
3: $c \leftarrow (d, CP)$
4: **return** c

5.3 Merge

A etapa *Merge* no algoritmo CMSA é em que as soluções geradas na fase *Construct* são adicionadas para formar a subinstância C'. Esta subinstância, essencialmente um subconjunto do problema original, é então resolvida usando um resolvedor exato na etapa *Solve*.

5.4 Solve

É a etapa em que um resolvedor exato é aplicado à subinstância C' para encontrar a melhor solução possível (S_{ILP}) dentro do conjunto de componentes de solução disponíveis. Para aplicação do solver, foi elaborada a seguinte formulação inteira:

$$\text{Minimizar} \quad \sum_{i \in C} x_i \tag{5.1}$$

$$\text{Sujeito a} \quad \sum_{i \in PC_j} x_i \geq 1, \quad \forall j \in P \tag{5.2}$$

$$x_i \in \{0,1\}, \quad \forall i \in C \tag{5.3}$$

- x_i — indica que a componente $i \in C$ é usada
- CP_j — é o conjunto das componentes que cobrem o ponto $j \in P$
- C — é o conjunto das componentes
- P — é o conjunto de pontos

A equação (5.1) representa a função objetivo que minimiza a quantidade de componentes. A restrição (5.2) obriga que todos os pontos sejam cobertos por pelo menos uma componente, e a restrição (5.3) define o domínio da variável x_i.

O resolvedor ILP é aplicado ao modelo gerado, buscando encontrar a solução ótima (ou a melhor solução possível dentro do tempo restante) para a subinstância. A solução encontrada pelo resolvedor, S_{ILP}, é uma solução válida para o problema original, pois utiliza apenas os componentes permitidos em C'. Se o resolvedor ILP não encontrar a solução ótima dentro do limite de tempo, a melhor solução encontrada até o momento é utilizada.

5.5 Adapt

A etapa *Adapt* do CMSA é a fase em que a subinstância C' é modificada com base na solução S_{ILP} retornada pelo resolvedor exato. O objetivo principal da etapa *Adapt* é controlar o tamanho e a composição da subinstância, buscando um equilíbrio entre exploração e intensificação.

Os componentes em C' que não fazem parte da solução S_{ILP} têm suas idades incrementadas. A idade de um componente indica o número de iterações consecutivas em que ele pertenceu à subinstância C' sem ser incluído na solução S_{ILP}. Componentes com idade acima de um limite pré-definido (age_{max}) são removidos da subinstância C'. Esse mecanismo visa evitar que componentes que parecem não contribuir para boas soluções continuem a ser considerados, o que poderia prejudicar a eficiência do resolvedor exato em iterações subsequentes. Todos os componentes presentes na solução S_{ILP} têm suas idades zeradas e são mantidos na subinstância C'.

6

Resultado e Discussão

6.1 Ambiente e ferramentas

Todos os testes foram executados em um computador equipado com um processador Intel® Core™ i9-10900KF, 32 GB de RAM e sistema operacional Ubuntu 20.04.5 LTS. Os algoritmos foram implementados na linguagem C++ e compilados com o GNU G++ versão 14.0.0, utilizando a flag de otimização `-O3`.

6.2 Parâmetros

Os parâmetros do CMSA foram calibrados através da ferramenta irace[1]. O *irace* [López-Ibáñez et al., 2016] é um pacote *open-source* implementado em `R` que tem o objetivo de configurar automaticamente hiperparâmetros de algoritmos de otimização. Funciona recebendo um grupo de instâncias e de parâmetros com seus respectivos domínios e utilizando, em seguida, ferramentas estatísticas para escolher uma configuração de parâmetros adequada. Na tabela 6.1 mostra os domínios avaliados e valor escolhido pelo o *irace*.

- n_a: Representa o número máximo de iterações da etapa construtiva. Neste caso, n_a está definido como *2*, portanto a construção das soluções ocorre por até duas iterações.
- age_{max}: Este parâmetro define a vida útil máxima que um componente pode atingir antes de ser excluído do processo. Assim que um componente atinge essa idade, ele é removido do conjunto de soluções. Neste contexto, $age_{\max}$ está definido como 1, então os componentes são descartados após a conclusão de uma única iteração.

[1] https://mlopez-ibanez.github.io/irace/

- `cplex_emphasis`: Esse parâmetro controla a ênfase do CPLEX durante o processo de otimização. Ele permite ajustar o foco do algoritmo para atender a diferentes objetivos, como melhorar a viabilidade ou a qualidade das soluções. Neste experimento, o parâmetro foi configurado como *3*, o que significa que o CPLEX enfatiza a melhoria do melhor bound encontrado[2].

- `cplex_warmstart`: Quando `cplex_warmstart` está configurado como `true`, o CPLEX utiliza uma solução de partida que pode acelerar a busca pela solução ótima. Neste caso, o valor está definido como `false`, logo o CPLEX não emprega uma solução inicial.

- `cplex_abort`: Este parâmetro utiliza um mecanismo de callback para encerrar o solucionador CPLEX assim que identifica uma solução superior à atual. Essa abordagem é particularmente eficaz na redução do tempo de processamento quando uma solução aceitável é encontrada rapidamente. Nesse contexto, `cplex_abort` está configurado como `true`, ativando essa funcionalidade.

Tabela 6.1: Parâmetros utilizados nos experimentos.

Parâmetro	**Domínio**	**Valor escolhido**
n_a	$\{1,\ldots,10\}$	2
$age_{\max}$	$\{1,\ldots,10\}$	1
`cplex_emphasis`	{0, 1, 2, 3, 4}	3
`cplex_warmstart`	{true, false}	false
`cplex_abort`	{true, false}	true

6.3 Instâncias

1. Instâncias da TSP-Lib: Quatro instâncias da biblioteca do Problema do Caixeiro Viajante (TSP-Lib) [Reinelt, 1995], nomeadamente *d1291*, *rl1889*, *u2319* e *pcb3038*, foram testadas utilizando diferentes raios de discos.

2. Instâncias Sintéticas: Conjuntos de 500 e 1500 pontos gerados aleatoriamente em três configurações: distribuídos uniformemente em um disco, em um anel e em um quadrado.

3. Instâncias do mundo real:

    - *usa*[3] *:* Um conjunto de dados com 115.475 cidades e vilas dos Estados Unidos.

[2]Para mais detalhes sobre o `cplex_emphasis`, consulte https://www-eio.upc.es/lceio/manuals/cplex90/relnotescplex/relnotescplex10.html

[3]www.math.uwaterloo.ca/tsp/

- *monalisa*[4] *:* Um conjunto de dados contendo 100.000 pontos formando uma representação linear da Mona Lisa.
- *mcz*[5]*:* Um conjunto de dados com 50 bairros da cidade de Maceió.

6.4 Resultados experimentais

Para avaliar o CMSA, os resultados foram comparados com os algoritmos mais recentes na literatura: a heurística *Dynamic Reduction* (DR) proposta por Chen et al. [2024] e o algoritmo aproximativo *FastCover*, desenvolvido por Friederich et al. [2023].

6.4.1 FastCover

O código-fonte do *FastCover* foi disponibilizado publicamente pelos autores, o que possibilitou a replicação dos experimentos apresentados na Tabela 6.3 e 6.4 no mesmo ambiente computacional. Foram utilizadas tanto instâncias sintéticas quanto reais, conforme detalhado na seção anterior, para a avaliação do *FastCover*.

Na análise comparativa com o algoritmo aproximativo *FastCover* [Friederich et al., 2023] para o problema de raio unitário, foram realizadas 10 execuções independentes para cada instância. Para o CMSA, os limites de tempo foram definidos como 200 milissegundos para instâncias com 500 pontos, 300 milissegundos para aquelas com 1500 pontos e 500 milissegundos para as reais.

Os resultados apresentados na Tabela 6.3 estão organizados por algoritmo, com colunas separadas para *"CMSA"* e *"FastCover"*. Os campos nessas colunas incluem:

- *best*: a melhor solução obtida entre as 10 execuções;
- *mean*: a qualidade média da solução ao longo das 10 execuções;
- *time*: o tempo médio de execução.

Para o *FastCover*, foi executada apenas uma única execução, e o desempenho foi reportado de acordo. A coluna *gap* representa a diferença relativa na qualidade da solução entre o *FastCover* e o CMSA, calculada da seguinte forma:

$$\text{gap} = \frac{x - x^*}{x} \tag{6.1}$$

em que x representa o melhor valor de solução encontrado pelo *FastCover* e x^* corresponde ao valor da solução obtida pelo CMSA.

A coluna *Pontos* mostra a quantidade de pontos naquela instância.

[4] www.math.uwaterloo.ca/tsp/

[5] Dados fornecidos por Instituto de Tecnologia em Informática e Informação do Estado de Alagoas (ITEC) [2020]

6.4.2 DR

O algoritmo DR não foi disponibilizado publicamente por seus autores. No entanto, a publicação fornece detalhes sobre a configuração computacional utilizada, que envolveu um processador Intel i9 — comparável ao utilizado para executar o CMSA neste estudo. Para uma comparação justa com o DR, foram empregadas as mesmas instâncias descritas no artigo original, seguindo rigorosamente o protocolo experimental estabelecido pelos autores.

Para garantir comparações robustas, foram realizadas 100 execuções independentes para cada instância, seguindo o protocolo especificado no trabalho original. Os resultados, resumidos na Tabela 6.2, estão organizados por algoritmo, com colunas separadas para *"CMSA"* e *"DR"*. Cada coluna apresenta as seguintes métricas:

- *best*: a melhor solução obtida nas 100 execuções;
- *mean*: a qualidade média da solução nas 100 execuções;
- *time*: o tempo médio de execução nas 100 execuções.

Adicionalmente, a linha rotulada como *Qtd. de melhores* indica o número de vezes que cada algoritmo alcançou o melhor resultado. A coluna *Raio* especifica o tamanho do disco utilizado no problema de cobertura.

Salienta-se que para cada instância, o limite de tempo para o CMSA foi configurado para se alinhar ao tempo gasto pelo DR, garantindo uma base consistente para comparação.

6.5 Discussão

Os resultados experimentais demonstram que a meta-heurística CMSA supera significativamente os métodos mais recentes da literatura, especificamente a heurística DR e o algoritmo aproximativo FastCover, em termos de qualidade da solução. Conforme evidenciado nas Tabelas 6.2, 6.3 e 6.4, o CMSA produziu consistentemente soluções com menores custos (indicados pelo valor *best*), mantendo tempos de execução competitivos, mesmo quando aplicado a instâncias de problemas complexos.

A Tabela 6.2 destaca a eficácia do CMSA em reduzir o valor de cobertura para diversas instâncias. Por exemplo, na instância *d1291* com raio de 500, o CMSA alcançou um valor de cobertura de 13, em comparação com a heurística DR, que produziu uma solução com valor 16. Isso ilustra a superioridade do CMSA na geração de soluções mais eficientes.

Em comparação com o FastCover, o CMSA demonstrou uma melhoria substancial ao minimizar o número de discos necessários para a cobertura de pontos nas instâncias *disk* e *annulus* (ver Tabela 6.3). Por exemplo, na instância *disk-01* com 500 pontos, o CMSA alcançou um valor *best* de 80 discos, enquanto o FastCover necessitou de 114 discos. Essa redução significativa destaca a capacidade do CMSA de gerar soluções mais compactas e econômicas.

Tabela 6.2: Resultados para instâncias do TSP

		DR			CMSA		
Nome	Raio	best	mean	time(ms)	best	mean	time(ms)
d1291	1500	**3**	**3.00**	76.1	**3**	**3.00**	76.58
d1291	1000	**5**	5.18	76.7	**5**	**5.00**	76.98
d1291	750	**7**	7.89	101.2	**7**	**7.22**	102.00
d1291	500	16	17.62	171.3	**13**	**13.03**	172.25
rl1889	4000	**6**	**6.23**	138.3	**6**	6.40	140.57
rl1889	3500	**8**	8.88	109.6	**8**	**8.33**	111.58
rl1889	3000	**11**	12.00	127.0	**11**	**11.51**	129.88
rl1889	2500	**15**	16.68	147.4	**15**	**15.16**	149.59
u2319	2000	**4**	4.17	214.8	**4**	**4.00**	215.64
u2319	1700	**5**	5.16	152.1	**5**	**5.00**	153.83
u2319	1400	**7**	7.53	160.1	**7**	**7.20**	162.53
u2319	1000	**12**	13.60	158.6	**12**	**12.93**	161.17
pcb3038	1000	**6**	6.47	308.2	**6**	**6.00**	311.70
pcb3038	700	**11**	12.56	265.8	**11**	**11.96**	270.83
pcb3038	600	**15**	16.53	247.5	**15**	**16.36**	252.42
pcb3038	500	**21**	23.44	292.8	**21**	**23.04**	297.33
Qtd. de melhores		15	2		16	15	

No entanto, em instâncias específicas, como *monalisa* e *usa*, o CMSA produziu resultados equivalentes aos do FastCover, com ambos os métodos alcançando o valor máximo necessário de cobertura. Esses casos sugerem que o CMSA pode enfrentar limitações em instâncias particularmente homogêneas ou de grande escala, onde ambos os algoritmos convergem para soluções semelhantes.

A comparação visual apresentada nas Figuras 6.1 e 6.2 reforça ainda mais as vantagens do CMSA. Para a instância *mcz*, a disposição dos discos gerados pelo CMSA claramente utiliza menos discos do que a configuração gerada pelo FastCover.

De forma geral, os resultados validam o CMSA como uma abordagem robusta e eficiente para o Problema de Cobertura de Discos de Raio Fixo ao entregar soluções de alta qualidade e avançar o estado da arte. O CMSA demonstra, portanto, ser particularmente eficaz no enfrentamento de instâncias de problemas desafiadoras e complexas.

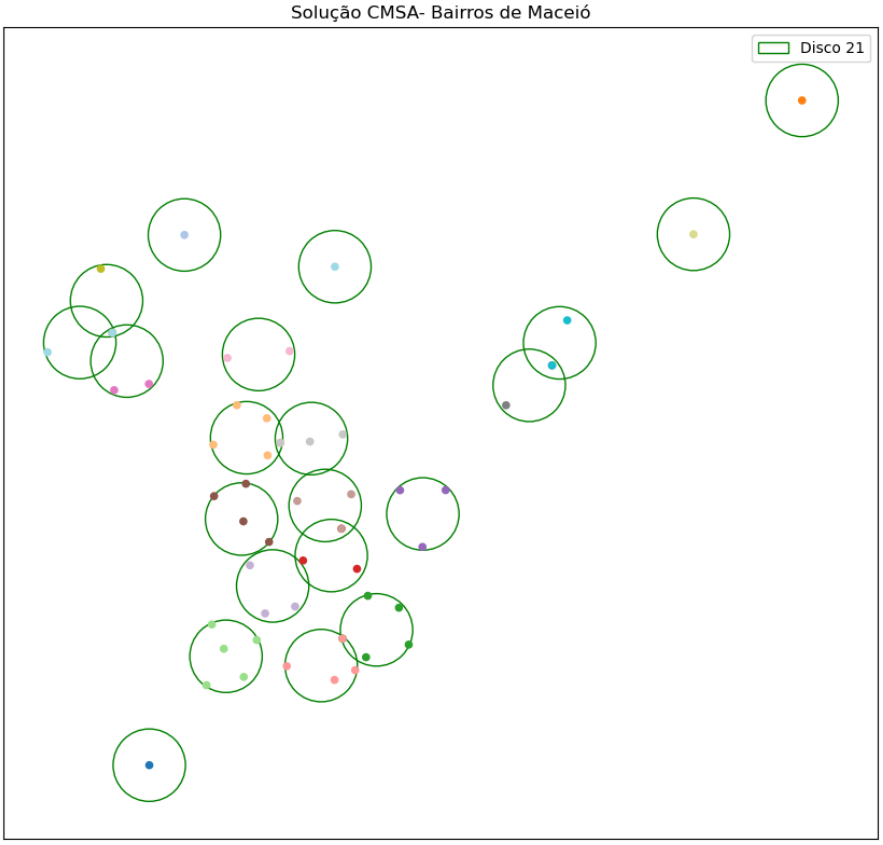

Figura 6.1: Solução do CMSA para bairros de Maceió

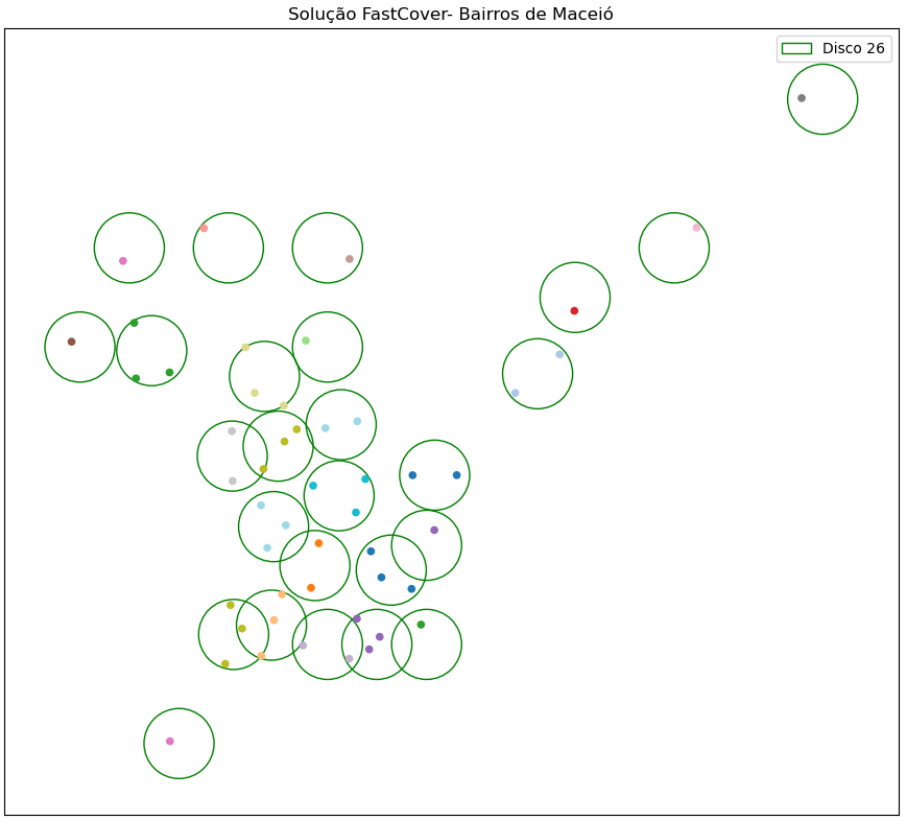

Figura 6.2: Solução do *FastCover* para bairros de Maceió

Tabela 6.3: Resultados para instâncias sintéticas

		FastCover		CMSA			
Nome	Pontos	best	time(ms)	best	mean	time(ms)	gap
disk-01	500	114	0.24	**80**	**81.9**	200	0.29
disk-02	500	115	0.24	**83**	**84.8**	200	0.27
disk-03	500	113	0.24	**80**	**81.3**	200	0.29
disk-04	500	113	0.23	**78**	**79.3**	200	0.30
disk-05	500	113	0.22	**77**	**78.6**	200	0.31
annulus-01	500	112	0.22	**76**	**77..4**	200	0.32
annulus-02	500	103	0.23	**70**	**72.1**	200	0.32
annulus-03	500	102	0.24	**66**	**67.8**	200	0.49
annulus-04	500	80	0.17	**52**	**53.1**	200	0.35
annulus-05	500	44	0.10	**29**	**29.3**	200	0.34
square-01	500	**500**	0.53	**500**	**500.0**	200	0.0
square-02	500	**496**	0.54	**496**	**496.0**	200	0.0
square-03	500	**483**	0.55	**483**	**483.0**	200	0.0
square-04	500	402	0.50	**388**	**388.0**	200	0.03
square-05	500	191	0.36	**155**	**156.2**	200	0.18
disk-01	1500	162	0.40	**124**	**131.1**	300	0.23
disk-02	1500	158	0.62	**123**	**127.3**	300	0.22
disk-03	1500	157	1.41	**116**	**122.3**	300	0.26
disk-04	1500	161	0.41	**114**	**119.1**	300	0.29
disk-05	1500	157	0.53	**116**	**122.3**	300	0.26
annulus-01	1500	160	0.42	**118**	**119.7**	300	0.26
annulus-02	1500	155	0.41	**112**	**115.1**	300	0.27
annulus-03	1500	133	0.36	**100**	**102.3**	300	0.24
annulus-04	1500	99	0.29	**65**	**71.9**	300	0.34
annulus-05	1500	51	0.20	**33**	**33.4**	300	0.35
square-01	1500	**1497**	1.60	**1497**	**1497.0**	300	0.0
square-02	1500	1483	1.55	**1480**	**1480.0**	300	0.002
square-03	1500	1367	1.52	**1360**	**1360.0**	300	0.005
square-04	1500	901	1.32	**832**	**833.4**	300	0.07
square-05	1500	340	0.74	**246**	**247.3**	300	0.11
Qntd. de melhores		4		30	30		

Tabela 6.4: Resultados para instâncias reais

		FastCover		CMSA		
Nome	Pontos	best	time(ms)	best	mean	time(ms)
mcz	50	26	0.001	21	22.6	500
monalisa	10000	10000	80	10000	10000.0	500
usa	115475	115475	100	115475	115475.0	500
Qntd. de melhores		3		3	3	

7

Considerações Finais

Neste trabalho, foi proposta uma matheurística CMSA para o Problema da Cobertura de Disco de Raio Fixo (PCDRF). Os experimentos realizados tiveram como objetivo comparar o desempenho do CMSA com as abordagens mais recentes da literatura: DR e FastCover. Os resultados computacionais mostraram que o CMSA encontrou soluções superiores em todos os testes computacionais, superando os outros dois algoritmos.

Conclui-se, portanto, que a aplicação do CMSA ao PCDRF foi satisfatória e contribuiu para o avanço do estado da arte do problema, superando as abordagens mais recentes encontradas na literatura. Esse avanço se reflete na obtenção de soluções mais eficientes e no uso otimizado de recursos computacionais, consolidando o CMSA como uma ferramenta robusta e eficaz para problemas de cobertura de discos em contextos variados.

No entanto, algumas limitações do trabalho foram observadas. Primeiramente, o CMSA apresenta desafios em termos de desempenho quando aplicado a instâncias extremamente grandes, onde os resultados se mostram equivalentes aos de outros algoritmos aproximativos, como o FastCover. Além disso, a implementação do CMSA pode exigir ajustes nos parâmetros para alcançar desempenho ideal em diferentes tipos de instâncias, o que limita sua aplicabilidade direta sem calibração prévia.

Para pesquisas futuras, sugere-se investigar formas de aprimorar a adaptabilidade do CMSA a instâncias grandes, como aquelas com mais de 100.000 pontos. Outra linha de pesquisa promissora seria a integração do CMSA com técnicas de aprendizado de máquina, visando uma calibração automática dos parâmetros do algoritmo conforme o tipo e o tamanho da instância de entrada. Além disso, explorar a combinação do CMSA com métodos paralelos pode reduzir o tempo de execução, tornando-o ainda mais eficiente em contextos onde o tempo de resposta é crítico.

Em suma, os resultados e contribuições deste trabalho oferecem uma base sólida para o desenvolvimento de novas abordagens híbridas para o PCDRF, abrindo caminhos para futuras inovações e consolidações no campo da otimização combinatória.

Referências bibliográficas

Manjanna Basappa, Rashmisnata Acharyya, and Gautam K. Das. Unit disk cover problem in 2d. *Journal of Discrete Algorithms*, 33:193–201, 2015. **DOI** 10.1016/j.jda.2015.05.002. URL https://www.sciencedirect.com/science/article/pii/S1570866715000556.

Ahmad Biniaz, Paul Liu, Anil Maheshwari, and Michiel Smid. Approximation algorithms for the unit disk cover problem in 2d and 3d. *Computational Geometry*, 60:8–18, 2017. **DOI** 10.1016/j.comgeo.2016.04.002. URL https://lens.org/029-423-852-637-51X.

Christian Blum, Pedro Pinacho, Manuel López-Ibáñez, and José A. Lozano. Construct, merge, solve & adapt a new general algorithm for combinatorial optimization. *Computers & Operations Research*, 68:75–88, 2016. ISSN 0305-0548. **DOI** https://doi.org/10.1016/j.cor.2015.10.014. URL https://www.sciencedirect.com/science/article/pii/S0305054815002452.

Marco Antonio Boschetti and Vittorio Maniezzo. Matheuristics: Using mathematics for heuristic design. *4OR*, 20(2):173–208, jun 2022. ISSN 1614-2411. **DOI** 10.1007/s10288-022-00510-8. URL https://doi.org/10.1007/s10288-022-00510-8.

Xiaopeng Chen, Xuehong Gao, Chong Li, and Chanseok Park. A new heuristic for the continuous fixed-radius disc cover problem. *Engineering Optimization*, pages 1–14, July 2024. **DOI** 10.1080/0305215x.2024.2368609. URL https://lens.org/056-399-562-050-288.

S. A. Cook. The complexity of theorem proving procedures. In *Proceedings of the Third Annual ACM Symposium*, pages 151–158, New York, 1971. ACM.

Stephen Cook. The p versus np problem. In *Clay Mathematical Institute; The Millennium Prize Problem*, 2000.

Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, second edition, 2000. URL http://www.cs.uu.nl/geobook/.

Adrian Dumitrescu, Anirban Ghosh, and Csaba D. Tóth. Online unit covering in euclidean space. *Theoretical Computer Science*, 809:218–230, 2020. ISSN 0304-3975. **DOI** https://doi.org/10.1016/j.tcs.2019.12.010. URL https://www.sciencedirect.com/science/article/pii/S0304397519307856.

Robert J. Fowler, Mike Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are np-complete⋆. *Information Processing Letters*, 12(3):133–137, 1981. **DOI** 10.1016/0020-0190(81)90111-3. URL https://lens.org/129-301-324-209-992.

Rachel Friederich, Anirban Ghosh, Matthew Graham, Brian Hicks, and Ronald Shevchenko. Experiments with unit disk cover algorithms for covering massive pointsets. *Computational Geometry*, 109:101925–101925, 2023. **DOI** 10.1016/j.comgeo.2022.101925. URL https://arxiv.org/abs/2205.01716.

Bin Fu, Zhixiang Chen, and Mahdi Abdelguerfi. An almost linear time 2.8334-approximation algorithm for the disc covering problem. In Ming-Yang Kao and Xiang-Yang Li, editors, *Algorithmic Aspects in Information and Management*, pages 317–326, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-72870-2.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, first edition edition, 1979. ISBN 0716710455. URL http://www.amazon.com/Computers-Intractability-NP-Completeness-Mathematical-Sciences/dp/0716710455.

Hossein Ghasemalizadeh and Mohammadreza Razzazi. An improved approximation algorithm for the most points covering problem. *Theory of Computing Systems*, 50(3): 545–558, 2012. ISSN 1433-0490. **DOI** 10.1007/s00224-011-9353-4. URL https://doi.org/10.1007/s00224-011-9353-4.

Teofilo F. Gonzalez. Covering a set of points in multidimensional space. *Information Processing Letters*, 40(4):181–188, 1991. **DOI** 10.1016/0020-0190(91)90075-s. URL http://www.sciencedirect.com/science/article/pii/002001909190075S.

Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and vlsi. *Journal of the ACM*, 32(1):130–136, 1985. **DOI** 10.1145/2455.214106. URL https://dl.acm.org/doi/10.1145/2455.214106.

Nan Hu. Approximation algorithms for geometric covering problems for disks and squares. Master of mathematics thesis, University of Waterloo, Waterloo, Ontario, Canada, 2013.

Instituto de Tecnologia em Informática e Informação do Estado de Alagoas (ITEC). Bairros de alagoas, 2020. URL https://dados.al.gov.br/catalogo/dataset/

bairros-de-alagoas/resource/b3a2a29a-285a-48d1-aafd-a5dee7e245b6. Acesso em: 7 jul. 2024.

R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, editors, *Complexity of Computer Computations: Proceedings of a Symposium on the Complexity of Computer Computations*, pages 85–103. Springer US, Boston, MA, 1972. Held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and Sponsored By the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department.

Paul Liu and Daniel Lu. A fast 25/6-approximation for the minimum unit disk cover problem, 2014. URL https://arxiv.org/abs/1406.3838.

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016. **DOI** 10.1016/j.orp.2016.09.002.

Vittorio Maniezzo, Thomas Sttzle, and Stefan Vo. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*. Springer Publishing Company, Incorporated, 1st edition, 2009. ISBN 144191305X.

Stefan Panov, Svetlana Panova, and Atanas Garbev. Finding the minimum number of disks of fixed radius needed to cover a set of points in the plane by maximinmax approach. In *2022 International Conference Automatics and Informatics (ICAI)*. IEEE, October 2022. **DOI** 10.1109/icai55857.2022.9960053. URL https://lens.org/108-921-295-130-85X.

G. Reinelt. Tsp-lib, 1995. URL http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/.

Franz Rothlauf. *Optimization Methods*, pages 45–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-540-72962-4. **DOI** 10.1007/978-3-540-72962-4_3. URL https://doi.org/10.1007/978-3-540-72962-4_3.

Michael Ian Shamos and Dan Hoey. Closest-point problems. *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pages 151–162, 1975. URL https://api.semanticscholar.org/CorpusID:40615455.

M. J. F. Souza. Inteligência computacional para otimização: meta-heurísticas. Technical report, Departamento de Computação, Universidade Federal de Ouro Preto, Ouro Preto, Minas Gerais, 2024. URL http://www.decom.ufop.br/prof/marcone/Disciplinas/InteligenciaComputacional/InteligenciaComputacional.pdf. Disponível em:

http://www.decom.ufop.br/prof/marcone/Disciplinas/InteligenciaComputacional/InteligenciaComputacional.pdf.

El-Ghazali Talbi. *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009. ISBN 0470278587.

Fani Tomova, Stefan Filipov, and Ana Avdzhieva. Covering a maximum number of points by a fixed number of equal disks via simulated annealing. *Journal of Chemical Technology and Metallurgy*, 59(2):457–464, 2024. **DOI** 10.59957/jctm.v59.i2.2024.26. URL https://lens.org/109-586-545-353-732.

Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936. URL http://www.cs.helsinki.fi/u/gionis/cc05/OnComputableNumbers.pdf.

L.G. Alves Viana. Uma meta-heurística híbrida para o problema de cobertura de discos ponderados. Bachelor's thesis, Universidade Federal de Alagoas, Instituto de Computação, Maceió, Brazil, 2022.

Miao Wang, Han Jiang, Yongzhe Li, Songtao Wu, and Lei Xia. Location determination of hierarchical service facilities using a multi-layered greedy heuristic approach. *Engineering Optimization*, 55(8):1422–1436, 2023. **DOI** 10.1080/0305215X.2022.2086540. URL https://doi.org/10.1080/0305215X.2022.2086540.

Ya Jun Yu and Yong Ching Lim. Design of linear phase fir filters in subexpression space using mixed integer linear programming. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(10):2330–2338, 2007. **DOI** 10.1109/TCSI.2007.904599.