

Uma meta-heurística híbrida para o Problema da Cobertura Contínua de Disco de Raio Fixo

Mateus S. Batista

Universidade Federal de Alagoas
Av. Lourival Melo Mota, S/n - Tabuleiro do Martins, Maceió - AL
msb@ic.ufal.br

Rian Gabriel S. Pinheiro

Universidade Federal de Alagoas
Av. Lourival Melo Mota, S/n - Tabuleiro do Martins, Maceió - AL
rian@ic.ufal.br

RESUMO

O Problema de Cobertura Contínua por Discos de Raio Fixo (PCCD) é um problema $\mathcal{NP}$ -difícil da área de localização de instalações. Consiste em posicionar o menor número possível de discos de raio fixo para cobrir um conjunto de pontos no plano euclidiano. Este trabalho propõe uma matheurística, denominada *Construct, Merge, Solve & Adapt (CMSA)*, para resolver o PCCD. A abordagem híbrida combina heurísticas construtivas, programação matemática e mecanismos adaptativos para melhorar a qualidade das soluções. Experimentos computacionais com instâncias do *TSP Challenge* comparam o desempenho do *CMSA* à heurística *Enhanced DR*. Os resultados mostraram a robustez e a eficácia da matheurística proposta — obtendo soluções superiores ou equivalentes em mais de 86% dos casos.

PALAVRAS CHAVE. CMSA, Meta-heurística híbrida, Cobertura de Discos

Tópicos: MH – Meta-heurísticas

ABSTRACT

The Fixed-Radius Disk Covering Problem (PCCD) is an $\mathcal{NP}$ -hard problem in the field of facility location. It consists of positioning the minimum possible number of fixed-radius disks to cover a set of points in the Euclidean plane. This work proposes a matheuristic, called *Construct, Merge, Solve & Adapt (CMSA)*, to solve the PCCD. The hybrid approach combines constructive heuristics, mathematical programming, and adaptive mechanisms to improve solution quality. Computational experiments using instances from the *TSP Challenge* compare the performance of *CMSA* with the *Enhanced DR* heuristic. The results showed the robustness and effectiveness of the proposed matheuristic — obtaining superior or equivalent solutions in more than 88% of cases.

KEYWORDS. CMSA, Hybrid metaheuristic, Disk Covering

Paper topics: MH – Metaheuristics

1. Introdução

O Problema de Cobertura Contínua por Discos de Raio Fixo (PCCD) enquadra-se na categoria de *facility location problem* (problemas de localização de instalações). De fato, tais problemas visam determinar a localização ideal de um conjunto de instalações, com o objetivo de minimizar custos, satisfazer demandas específicas e atender a um conjunto de restrições [Hale e Moberg, 2003]. Dessa forma, as instalações correspondem aos discos, os pontos, às demandas e o conjunto de restrições consiste na cobertura de todos os pontos.

Além disso, esse problema possui aplicações relevantes em áreas como redes sem fio, robótica, processamento de imagens e aprendizado de máquina [Friederich et al., 2023]. Por exemplo, em redes sem fio, os clientes podem ser modelados como pontos, enquanto os transmissores são representados por discos [Hu, 2013]. Nesse contexto, o problema consiste em determinar o número mínimo de transmissores necessários para garantir a cobertura total dos clientes.

Entretanto, a natureza combinatória do problema — determinar a posição, num espaço contínuo, e a menor quantidade possível de discos que assegure a cobertura de todos os pontos — torna-o computacionalmente inviável à medida que o tamanho da instância aumenta. Conforme demonstrado por Fowler et al. [1981], trata-se de um problema $\mathcal{NP}$ -difícil, o que implica que não se conhece nenhum algoritmo em tempo polinomial capaz de resolvê-lo.

Neste trabalho, é proposta uma heurística *Construct, Merge, Solve & Adapt (CMSA)*, para resolver o PCCD. O restante do artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados ao problema. A Seção 3 descreve o problema. Na Seção 4, apresenta-se o $CMSA_{PCCD}$ e seus componentes. Relata-se os resultados experimentais na Seção 5. Finalmente, a Seção 6 apresenta as considerações finais.

2. Trabalhos relacionados

Muitas abordagens foram propostas ao longo dos anos, quase sempre, algoritmos aproximativos, como, por exemplo, Hochbaum e Maass [1985]; Gonzalez [1991]; Brönnimann e Goodrich [1995]; Franceschetti et al. [2001]; Charikar et al. [2004]; Liu e Lu [2014]; Biniaz et al. [2017]; Liaw et al. [2018] e Friederich et al. [2023].

Entre essas abordagens, uma das primeiras estratégias é atribuída a Hochbaum e Maass [1985], que utilizaram a *shifting strategy*. Nesse procedimento, divide-se o plano em faixas verticais de largura $l \cdot D$, em que l é o número de deslocamento e D é o diâmetro do disco, subsequentemente, cada faixa é particionada em quadrados de lado $l \cdot D$, nos quais se aplica busca exaustiva para determinar coberturas ótimas locais. Esse método é um *Polynomial-Time Approximation Scheme* (PTAS), que, para um conjunto de n pontos no plano, opera em tempo $O\left(\ell^4(2n)^{4\ell^2+1}\right)$, com um fator de aproximação de $\left(1 + \frac{1}{\ell}\right)^2$.

Mais recentemente Friederich et al. [2023] propuseram uma 7-aproximação denominada *FastCover* para o CDU. A abordagem proposta baseia-se na discretização dos pontos do plano, associando cada ponto (x_i, y_i) a inteiros a e b , formando uma grade quadrada de lado $\sqrt{2}$, definida por: $a = \left\lfloor \frac{x_i}{\sqrt{2}} \right\rfloor$, $b = \left\lfloor \frac{y_i}{\sqrt{2}} \right\rfloor$. A partir dessa discretização, define-se um disco unitário $D(a, b)$ que circunscreve a célula $\sigma(a, b)$ correspondente da grade. Além disso, os autores propõem duas variantes otimizadas do *FastCover*, denominadas *FastCover+* e *FastCover++*, que apresentam melhorias adicionais de desempenho.

Em contraste com os trabalhos anteriores, que se concentram em algoritmos aproximativos, Chen et al. [2024] propuseram a heurística denominada Redução Dinâmica (*DR*) para o PCCD — descrita pelos autores como a primeira heurística específica para esse problema. Essa abordagem opera em duas fases principais: inicialização e redução. Na primeira etapa, é estabelecido um limite superior para a quantidade de discos, para isso um ponto aleatório é selecionando

como centro, posteriormente, escolhe-se, entre os pontos não cobertos, os pontos mais distantes, até garantir que todos os pontos estejam cobertos. Subsequentemente, busca-se eliminar discos redundantes através da fase de redução: os pontos são alocados ao centro mais próximo (formando *clusters*) e o centro é recalculado para cada *cluster*. Se o raio do maior desses círculos (r_{max}) for menor ou igual a r , o algoritmo tenta remover um disco e repete a avaliação; esse processo continua até que a remoção de um disco adicional viole a restrição de cobertura ($r_{max} > r$). Os autores também apresentam uma variante otimizada da heurística, o *Enhanced DR*, que aproveita a solução do DR na etapa de inicialização.

Ademais, salienta-se que muitos trabalhos tratam da versão particular em que raio é unitário, os quais são chamados, geralmente, de problema de cobertura de raio unitário (CDU) (*Unit-Disk Cover problem*), como em Liu e Lu [2014], Biniáz et al. [2017], Liaw et al. [2018] e Friederich et al. [2023].

3. Caracterização do problema

Nesta secção, descreve-se o PCCD detalhando sua instância e função objetivo. Descreve-se uma instância do PCCD como um conjunto finito de pontos P no plano euclidiano $\mathbb{E}^2$ e um número real positivo r . O objetivo é determinar o menor número possível de discos, todos com raio r , e suas respectivas localizações no plano, de forma que cada ponto em P seja coberto por pelo menos um desses discos.

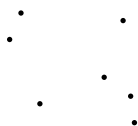


Figura 1: Exemplo de entrada

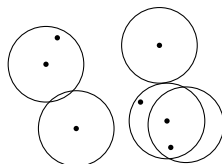


Figura 2: Solução viável

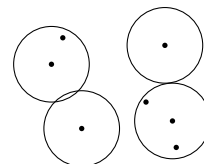


Figura 3: Solução ótima

A Figura 1 ilustra uma instância ingênua composta por sete pontos no plano, enquanto as Figuras 2 e 3 apresentam, respectivamente, uma solução viável e a solução ótima: esta utiliza quatro discos, por outro lado, aquela, cinco discos. Assim, a solução mostrada na Figura 3 é a melhor, porque utiliza menos discos.

4. Algoritmo *Construct, Merge, Solve & Adapt*

O *Construct, Merge, Solve & Adapt* (CMSA), proposto por Blum et al. [2016], trata-se de uma matheurística — um híbrido entre meta-heurística e métodos exatos —, na qual a ideia central consiste na decomposição do problema em componentes menores. Ao serem selecionados e combinados por um método exato, esses componentes constituem uma solução viável para o problema original.

Com efeito, o CMSA, a cada iteração, realiza as seguintes etapas: constrói coberturas aleatoriamente; em seguida, os componentes dessas soluções são adicionados a uma sub-instância, inicialmente, vazia, denotada por C' ; subsequentemente, um solucionador exato combina as componentes de C' e determina a melhor solução; por fim, a sub-instância atual é adaptada com base em um mecanismo de envelhecimento, componentes da solução que parecem inúteis são removidos de C' para aliviar o solucionador na próxima iteração do algoritmo. As etapas supracitadas correspondem, respectivamente, às fases *Construção*, *Combinação*, *Solução* e *Adaptação*, as quais serão detalhadas a seguir.

Com esse intuito, define-se os conjuntos C , C' , S , a função f de avaliação e o componente: C corresponde ao conjuntos de todos os componentes que podem fazer parte de uma solução para o problema; C' é o conjunto de componentes ativas, que é iterativamente expandido e adaptado; S é um subconjunto de C , que representa uma solução para o problema; f é uma função que avalia a qualidade de uma solução, $f : S \rightarrow \mathbb{R}$, em que $f(S)$ é a cardinalidade do conjunto S . Por

fim, um componente é definido como um par ordenado $c = (d, P_c) \in C$, em que d representa o centro do disco e P_c o conjunto de pontos cobertos pela a componente c .

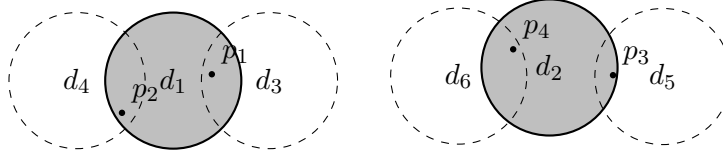


Figura 4: CMSA aplicado ao PCCD

Por exemplo, na Figura 4, observa-se $P = \{p_1, p_2, p_3, p_4\}$, um conjunto de pontos, e $C' = \{c_1, c_2, c_3, c_4, c_5, c_6\}$, um conjunto de componentes, os quais são determinados por um centro e um conjunto de pontos; como o raio é fixo, sua inclusão na representação de cada componente torna-se desnecessária. As componentes são $c_1 = (d_1, \{p_1, p_2\})$, $c_2 = (d_2, \{p_3, p_4\})$, $c_3 = (d_3, \{p_1\})$, $c_4 = (d_4, \{p_2\})$, $c_5 = (d_5, \{p_3\})$, e $c_6 = (d_6, \{p_4\})$. Note que $S = \{c_1, c_2\}$ cobre integralmente todos os pontos, sendo, portanto, considerada factível, com custo igual a $F(S) = 2$, que corresponde à quantidade de componentes.

O pseudocódigo referente ao algoritmo $CMSA_{PCCD}$ encontra-se detalhado no Algoritmo 1. A execução do algoritmo requer a definição de três hiperparâmetros de entrada: n_a , $idade_{max}$ e t_{limit} . O primeiro n_a controla o número de componentes, os quais são gerados a cada iteração. O segundo $idade_{max}$ estabelece a idade máxima permitida para cada componente e o terceiro t_{limit} define o tempo limite de execução do algoritmo.

O algoritmo inicia-se na linha 2, a qual inicializa o conjunto C' . Subsequentemente, o *loop* principal executa as quatro etapas centrais do algoritmo. A primeira delas, a etapa de *Construção*, é executada na linha 5, sendo responsável por gerar o conjunto S de componentes. Na sequência, inicia-se a fase de *Combinação*, na qual os componentes recém-criados têm suas idades inicializadas e são incorporados ao conjunto C' . Posteriormente, na linha 10 é calculada a variável t_{ILP} , que determina o tempo máximo de execução permitido para o procedimento *SolveSubinstance*. Caso esse tempo seja menor do que um valor de tolerância, o algoritmo é interrompido para respeitar o limite de tempo estabelecido. A seguir, o processo de adaptação é executado na linha 16, consistindo na eliminação dos componentes que ultrapassam a idade limite. Por último, o algoritmo retorna a melhor solução encontrada.

O restante desta seção está estruturado da seguinte forma. A Seção 4.1 fornece uma explicação detalhada da fase de *Construção*. A Seção 4.2 descreve a fase de *Combinação*. A Seção 4.3 descreve a fase de *Solução*. Finalmente, a Seção 4.4 examina a fase de *Adaptação*.

4.1. Construct

A fase *Construção* representa o ponto inicial de cada iteração do CMSA, a qual é responsável por gerar componentes, sendo acionada por meio da chamada ao procedimento *Gera-dorDeSolucoes*. O Algoritmo 2 descreve esse procedimento, no qual emprega uma estratégia baseada na subdivisão aleatória de quadrantes. Em linhas gerais, a ideia do algoritmo reside em gerar, de forma aleatória, partições, denominadas quadrantes, sucessivamente menores, de modo que cada partição seja circunscrita por um disco e assegure a cobertura de todos os pontos em seu interior.

Define-se um quadrante como a região retangular delimitada por dois pontos extremos: $P(x_{min}, y_{min})$ e $P(x_{max}, y_{max})$. No procedimento, o quadrante inicial abrange todos os pontos de entrada (linha 3). Para criar subquadrantes (linha 12), um ponto aleatório $P_{rand} = (x_{rand}, y_{rand})$ é gerado dentro dos limites do quadrante atual. Retas paralelas aos eixos x e y , passando por P_{rand} , dividem o quadrante atual em quatro novos quadrantes menores, esse processão de divisão

Algorithm 1 CMSA pseudocode

```

1: procedure CMSA( $n_a, idade_{max}, t_{limit}$ )
2:    $C' := \{\}$ 
3:    $idade[c] := 0$  para todo  $c \in C$ 
4:   while Limite de tempo de CPU não alcançado do
5:     for  $i = 1, \dots, n_a$  do
6:        $S := \text{GeradorDeSoluções}(Pontos)$  ▷ Construct (Section 4.1)
7:       for all  $c \in S$  do
8:          $idade[c] := 0$ 
9:          $C' := C' \cup c$  ▷ Merge (Section 4.2)
10:       $t_{ILP} := t_{limit} - t_{decorrido}$ 
11:      if  $t_{ILP} < tolerancia$  then
12:        break
13:       $S_{ILP} := \text{SolveSubinstance}(C', t_{ILP})$  ▷ Solve (Section 4.3)
14:      if  $f(S_{ILP}) < f(S_{bsf})$  then
15:         $S_{bsf} := S_{ILP}$ 
16:       $\text{Adaptcao}(C', S_{ILP}, idade_{max})$  ▷ Adapt (Section 4.4)
17:   return  $S_{bsf}$ 

```

é semelhante à criação de uma *KD-Tree* [Bentley, 1975]. Além disso, o diâmetro do quadrante é calculado como a distância euclidiana entre esses dois pontos, a qual é utilizado para verificar se o quadrante está contido no disco (linha 7).

Algorithm 2 Solution construction procedure

```

1: procedure GERADORDESOLUCOES( $Pontos$ )
2:    $S := \emptyset$ 
3:    $Quadrante := \text{CriaQuadrante}(Pontos)$ 
4:    $queue.push(Quadrante)$ 
5:   while not  $queue.empty()$  do
6:      $quadrante := queue.pop()$ 
7:     if  $quadrante.diametro \leq rand_{int}(1, 2) \times 2 \times raio$  then
8:        $c := \text{CriaComponente}(quadrante, raio)$ 
9:        $S := S \cup \{c\}$ 
10:    else
11:       $P_{rand} := Pontos(rand_{uniform}(quadrante))$ 
12:       $q_1, q_2, q_3, q_4 := \text{CriaQuadrante}(P_{rand}, quadrante)$ 
13:      adicionar  $q_1, q_2, q_3, q_4$  à fila se não estiverem vazios
14:   return  $S$ 

```

Esse fase inicia com a criação de um conjunto vazio S , o qual armazenará os componentes gerados, e o quadrante inicial. Esse quadrante inicial é inserido na fila (*queue*), que será utilizada para gerenciar os quadrantes a serem processados. Enquanto a fila não estiver vazia, o algoritmo remove o quadrante do início da fila e verifica se seu diâmetro é menor ou igual ao diâmetro do disco multiplicado por um número inteiro aleatório, 1 ou 2. Quando o valor sorteado é 1, o quadrante está garantidamente contido em um disco centrado no próprio quadrante. Já quando o valor é 2, essa garantia não existe. No entanto, a multiplicação por 2 reduz a quantidade de partições necessárias, acelerando a convergência do algoritmo. Por esse motivo, na primeira iteração do CMSA, utiliza-se obrigatoriamente o fator 1, assegurando uma solução viável. Após essa verificação, um novo componente é criado por meio da função *CriaComponente* (linha 8). Essa função inicia calculando o centro do quadrante, obtido pela média das coordenadas dos seus pontos extremos. Em seguida, realiza-se uma consulta espacial em uma *KD-Tree*, com o objetivo de buscar os pontos mais próximos desse centro. O componente é então formado por todos os pontos cuja distância ao

centro seja menor ou igual a r , o raio do disco.

Por outro lado, se o quadrante tiver um diâmetro superior ao limite estabelecido, é gerado um ponto aleatório P_{rand} dentro de seus limites, utilizando coordenadas x e y aleatórias. Esse ponto serve como referência para subdividir o quadrante em quatro subquadrantes, operação realizada na linha 12. O processo iterativo continua até que a fila esteja vazia. Ao final, o conjunto S , contendo todos os componentes gerados, é retornado como resultado. Esse algoritmo combina uma abordagem probabilística com subdivisão espacial, permitindo a exploração eficiente e diversificada do espaço de soluções.

4.2. Merge

O estágio de *Combinação*, segunda etapa da matheurística, é iniciado após a conclusão da fase de *Construção*, conforme visto na linha 9 do Algoritmo 1. Nessa etapa, os novos componentes gerados pelo procedimento *GeradorDeSolucoes* são incorporados ao conjunto de componentes C' , que contém elementos produzidos em iterações anteriores, e suas idades inicializadas com valor igual a zero.

4.3. Solve

A etapa *Solução*, correspondente à terceira fase, inicia-se no procedimento *SolveSubinstance* (C', t_{ILP}), localizado na linha 13. Nessa fase, em um intervalo de tempo máximo determinado t_{ILP} , um resolvidor ILP (*Integer Linear Programming*) é aplicado com o fito de selecionar os melhores componentes de modo a determinar a melhor solução possível na subinstância C' .

O modelo ILP adotado para esse processo é a formulação do problema de *set cover* [Vazirani, 2003], no qual consiste em determinar a menor quantidade de subconjuntos, cuja união é o conjunto universo. Dessa forma, os subconjuntos do *set cover* representam os discos e os elementos, os pontos do PCCD.

A formulação ILP é a seguinte:

$$\text{Minimizar} \quad \sum_{i \in I} x_i \quad (1)$$

$$\text{Sujeito a} \quad \sum_{i \in I_j} x_i \geq 1, \quad \forall j \in P \quad (2)$$

$$x_i \in \{0, 1\}, \quad \forall i \in I \quad (3)$$

- x_i — indica que o componente é usada
- P — é o conjunto de pontos
- I — é o conjunto de índices dos componentes presentes em C'
- I_j — é o conjunto de índices dos componentes que cobrem o ponto $j \in P$

A variável binária x_i é definida para cada componente $c \in C'$ (identificada pelo seu índice $i \in I$). A equação (1) representa a função objetivo, que minimiza o número total de componentes selecionados. As restrições (2) asseguram que cada ponto $j \in P$ seja coberto por pelo menos uma componente selecionada. Por fim, a restrição (3) estabelece o domínio binário da variável de decisão x_i .

4.4. Adapt

A etapa denominada de *Adaptação* (linha 16), correspondente à quarta e última fase do CMSA, é responsável pela implementação do mecanismo de envelhecimento dos componentes. Nessa fase, o primeiro passo é reiniciar as idades dos componentes selecionados pelo procedimento *SolverSubinstance* (C', t_{ILP}), em seguida, incrementar as idades dos demais componentes e eliminar os componentes do conjunto C' que ultrapassaram o limite máximo definido, $idade_{\text{max}}$.

5. Experimentos computacionais

Todos os experimentos foram conduzidos em um ambiente computacional equipado com um processador AMD Ryzen 7 6800H, 16GB de memória RAM e o sistema operacional *Arch Linux* na versão 6.14.2-arch1-1. Os algoritmos desenvolvidos foram implementados na linguagem de programação C++ e compilados utilizando o compilador GNU G++ versão 17.0, com a flag de otimização `-O3`, e usando a biblioteca CGAL 6.0.1 [The CGAL Project, 2024]. Para a resolução da formulação matemática, foi empregado o CPLEX 12.9 no modo *single-threaded*.

As instâncias utilizadas nos experimentos foram obtidas do repositório da Rutgers University¹, do *TSP Challenge*. Foram utilizadas 23 instâncias distintas e para cada instância consideraram-se três valores de raio distintos (500, 1000 e 2000), resultando em um total de 69 configurações experimentais. O identificador de cada instância corresponde ao número de pontos que ela contém. A definição dos valores de raio seguiu critérios semelhantes aos adotados por Chen et al. [2024].

5.1. Instâncias de teste e parâmetros

Os parâmetros do *CMSA_{PCCD}* foram calibrados usando a ferramenta *irace* [López-Ibáñez et al., 2016]. Trata-se de um pacote de código aberto implementado em R, que auxilia na configuração automática de hiperparâmetros de algoritmos. Para garantir uma calibração rápida e imparcial, foram selecionadas aleatoriamente 10 instâncias com raios diferentes (750 e 1500) do conjunto de testes, de modo a evitar viés no processo de avaliação.

Além dos hiperparâmetros intrínsecos ao *CMSA*, a calibração também considerou parâmetros específicos do CPLEX, uma vez que as configurações padrão nem sempre representam a melhor escolha. Foram ajustados: n_a , define o número de execuções da etapa de *Construção*; $idade_{max}$, idade máxima que um componente pode atingir; $cplex_{emphasis}$, controla o foco do CPLEX balanceando entre velocidade, viabilidade, otimalidade e melhoria dos limites; $cplex_{warmstart}$, uso de solução inicial; $cplex_{abort}$, permite configurar o CPLEX para interromper a execução assim que a melhor solução encontrada for aprimorada. Os domínios explorados para cada um desses parâmetros e a configuração final selecionada pelo *irace* são detalhados na Tabela 1.

Tabela 1: Hiperparâmetros ajustados com o *irace*.

Parameter	Domain	Chosen value
n_a	$\{1, \dots, 5\}$	1
max_{age}	$\{0, \dots, 5\}$	0
$cplex_{emphs}$	$\{0, \dots, 4\}$	2
$cplex_{abort}$	$\{true, false\}$	true
$cplex_{warmstart}$	$\{true, false\}$	false

A configuração final selecionada pelo *irace* equilibra diversidade e custo computacional com $n_a = 1$. A escolha de $idade_{max} = 0$ adota uma política agressiva de adaptação, removendo de imediato componentes inativos. Isso indica que é preferível manter subinstâncias de tamanho reduzido, porque possivelmente o CPLEX encontra dificuldades ao processar subinstâncias de maior porte. Já $cplex_{emphasis} = 2$ foca na comprovação da otimalidade, enquanto $cplex_{abort} = true$ acelera o processo ao interromper a busca ao encontrar uma solução melhor do que S_{bsf} . Por fim, $cplex_{warmstart} = false$ indica que o CPLEX não utiliza uma solução inicial.

5.2. Resultados

Para avaliar os resultado computacionais do *CMSA_{PCCD}*, foi escolhido o algoritmo *Enhanced DR* [Chen et al., 2024] como comparativo. Segundo os autores, o *Enhanced DR* possui

¹<http://archive.dimacs.rutgers.edu/Challenges/TSP/>

um desempenho bastante superior às abordagens aproximativas encontradas na literatura. Incluindo os algoritmos mais recentes como o FastCover++ [Friederich et al., 2023].

Neste trabalho, a comparação com *Enhanced DR* foi feita através de uma implementação em C++, já que o código do mesmo não foi disponibilizado pelos autores. A Tabela 2 compara os resultados reportados na literatura para a heurística *Enhanced DR* com aqueles obtidos pela implementação do mesmo algoritmo. Para verificar a corretude da implementação, realizou-se o teste estático de *Friedman* [Friedman, 1937] comparando as médias dos resultados obtidos por ambas as versões (original e implementada). O teste resultou em um p -valor de 0.2113. Como esse valor é maior que o nível de significância usual de 0.05, conclui-se que não há diferença estatisticamente significativa entre os resultados da implementação original e da realizada, o que valida a versão do algoritmo *Enhanced DR* usada para comparar com o $CMSA_{PCCD}$.

A Tabela 3 apresenta a comparação entre a heurística *Enhanced DR* e a proposta $CMSA_{PCCD}$ considerando diferentes instâncias. As colunas *best* e *mean* indicam, respectivamente, o melhor resultado e a média dos resultados obtidos em 30 execuções para cada instância. A coluna *time(ms)* mostra o tempo médio de execução em milissegundos. A coluna *time to best(ms)* para o $CMSA_{PCCD}$ representa o tempo médio, em milissegundos, necessário para encontrar a melhor solução conhecida para aquela instância, calculado sobre as 30 execuções. Especificamente, para cada execução, se a melhor solução conhecida foi encontrada, utiliza-se o tempo registrado até encontrá-la *Best solution time*, caso contrário, utiliza-se o tempo total daquela execução. Esse valor médio reflete o esforço computacional para atingir a solução de melhor qualidade conhecida. A coluna *gap(%)* representa o gap percentual entre a solução encontrada pela heurística e pelo $CMSA$, calculado como:

$$100 \times \frac{best_{CMSA} - best_{DR}}{best_{DR}}$$

A Tabela 4 oferece um resumo conciso do desempenho comparativo entre o $CMSA_{PCCD}$ e o *Enhanced DR*. Ela quantifica o número de vezes que cada algoritmo obteve um resultado estritamente superior em comparação com o outro, bem como o número de empates em relação às métricas *best* e *mean* Tabela 3.

Tabela 2: Comparação entre o *Enhanced DR* [Chen et al., 2024] e a implementação utilizada neste trabalho

Instância	<i>Enhanced DR</i>			<i>Enhanced DR</i> Implementado			
	Raio	best	mean time(ms)	best	mean	time(ms)	
d1291	1500	3	3,00	148,20	3	3,00	0,86
d1291	1000	5	5,08	136,80	5	5,13	1,43
d1291	750	7	7,47	165,80	7	7,60	2,48
d1291	500	16	17,36	246,00	14	16,81	4,45
rl1889	4000	6	6,16	236,70	6	6,07	2,88
rl1889	3500	8	8,55	192,40	8	8,69	3,18
rl1889	3000	10	11,59	213,60	10	11,62	3,95
rl1889	2500	14	15,91	237,80	14	15,88	5,02
u2319	2000	4	4,03	409,10	4	4,12	2,54
u2319	1700	5	5,02	305,00	5	5,04	2,06
u2319	1400	7	7,32	307,30	7	7,38	3,11
u2319	1000	12	13,19	264,50	12	13,34	4,65
pcb3038	1000	6	6,21	562,20	6	6,35	4,01
pcb3038	700	11	12,27	428,90	11	12,28	6,97
pcb3038	600	14	16,09	416,90	14	16,04	7,91
pcb3038	500	20	22,58	451,30	20	22,46	10,56
melhores valores	15	10	0	16	7		16

Tabela 3: Comparação de desempenho entre $CMSA_{PCCD}$ e a heurística *Enhanced DR*

Instância	<i>Enhanced DR</i>				$CMSA_{PCCD}$				
	Raio	best	mean	time(ms)	best	mean	time(ms)	time to best(ms)	gap(%)
d1291	500	15	16,90	4,71	13	13,03	300,4	127,3	-13,33
d1655	500	13	16,63	4,01	12	12,37	300,7	201,8	-7,69
d2103	500	18	20,73	11,51	17	17,37	300,7	236,6	-5,56
dsj1000	500	1000	1000,00	88,01	994	994,00	300,3	5,8	-0,60
fl1400	500	6	6,93	2,06	6	6,10	300,2	160,6	0,00
fl1577	500	6	6,10	1,96	6	6,00	300,5	32,9	0,00
fl3795	500	6	6,63	5,95	7	7,00	301,0	301,0	16,67
nrw1379	500	8	8,17	2,12	8	8,00	300,4	85,2	0,00
pcb1173	500	10	11,23	2,39	10	10,83	300,7	278,4	0,00
pcb3038	500	20	22,43	10,57	21	21,93	301,6	301,6	5,00
pr1002	500	215	223,30	12,85	126	128,67	300,4	299,5	-41,40
pr2392	500	207	238,07	81,72	170	173,67	301,2	301,2	-17,87
rl1304	500	197	225,50	35,06	153	155,83	300,5	299,5	-22,34
rl1323	500	266	274,77	32,54	163	166,20	300,5	299,8	-38,72
rl1889	500	230	252,73	71,82	185	187,53	300,6	292,2	-19,57
rl5915	500	297	358,17	364,56	274	282,57	303,3	301,7	-7,74
u1060	500	173	178,50	12,97	107	108,30	300,3	286,6	-38,15
u1432	500	43	46,40	7,06	39	40,40	300,8	295,7	-9,30
u1817	500	8	9,33	3,55	8	8,97	300,9	295,7	0,00
u2152	500	8	9,37	3,89	9	9,13	301,1	301,1	12,50
u2319	500	45	49,50	11,32	42	44,03	301,3	298,2	-6,67
vm1084	500	174	215,03	20,14	134	136,60	300,3	300,3	-22,99
vm1748	500	275	294,27	57,74	179	185,83	300,7	300,2	-34,91
d1291	1000	5	5,20	1,48	5	5,00	300,3	23,9	0,00
d1655	1000	5	5,00	1,73	5	5,00	300,3	6,3	0,00
d2103	1000	5	5,90	2,61	6	6,00	300,7	300,7	20,00
dsj1000	1000	993	993,97	89,85	987	987,00	300,4	7,0	-0,60
fl1400	1000	3	3,23	0,72	3	3,00	300,3	14,1	0,00
fl1577	1000	2	2,00	0,50	2	2,00	300,3	22,0	0,00
fl3795	1000	3	3,27	1,84	3	3,00	301,0	36,0	0,00
nrw1379	1000	2	2,57	0,75	3	3,00	300,2	300,2	50,00
pcb1173	1000	3	3,63	1,09	4	4,00	300,3	300,3	33,33
pcb3038	1000	6	6,33	4,16	6	6,10	301,4	112,5	0,00
pr1002	1000	73	79,23	3,18	48	48,73	300,3	273,0	-34,25
pr2392	1000	56	66,23	18,06	57	60,00	301,6	301,6	1,79
rl1304	1000	61	72,70	10,90	57	59,20	300,5	299,1	-6,56
rl1323	1000	86	92,80	10,75	62	63,33	300,5	291,7	-27,91
rl1889	1000	76	89,40	16,71	67	69,07	300,8	299,2	-11,84
rl5915	1000	82	94,77	77,34	87	94,30	306,6	306,6	6,10
u1060	1000	43	57,13	4,25	39	40,87	300,5	299,4	-9,30
u1432	1000	11	12,57	2,94	12	12,03	300,6	300,6	9,09
u1817	1000	3	3,10	1,12	3	3,00	300,5	27,8	0,00
u2152	1000	3	3,03	1,42	3	3,00	300,8	20,6	0,00
u2319	1000	12	13,43	4,57	12	12,70	301,1	278,8	0,00
vm1084	1000	53	63,87	7,56	50	50,80	300,4	278,6	-5,66
vm1748	1000	72	86,50	16,10	67	68,80	300,7	296,6	-6,94
d1291	2000	2	2,00	0,37	2	2,00	300,3	3,1	0,00
d1655	2000	2	2,00	0,49	2	2,00	300,6	3,1	0,00
d2103	2000	2	2,00	0,75	2	2,00	300,4	30,7	0,00
dsj1000	2000	985	986,57	90,62	940	940,00	300,4	20,5	-4,57
fl1400	2000	1	1,00	0,29	1	1,00	300,3	2,2	0,00
fl1577	2000	1	1,00	0,27	1	1,00	300,2	2,8	0,00
fl3795	2000	1	1,00	0,75	1	1,00	301,2	5,9	0,00
nrw1379	2000	1	1,00	0,26	1	1,00	300,3	2,3	0,00
pcb1173	2000	1	1,00	0,25	1	1,00	300,2	5,1	0,00
pcb3038	2000	2	2,07	1,08	2	2,00	301,0	34,2	0,00
pr1002	2000	16	18,20	2,34	15	15,67	300,5	285,3	-6,25
pr2392	2000	16	17,63	6,94	16	17,33	301,4	294,7	0,00

Continua na próxima página

Tabela 3 – continuação da página anterior

Instância	Raio	Enhanced DR			CMSA _{PCCD}				
		best	mean	time(ms)	best	mean	time(ms)	time to best(ms)	gap(%)
rl1304	2000	20	22,63	3,78	19	20,10	300,6	297,1	-5,00
rl1323	2000	20	22,40	4,86	20	20,77	300,6	292,0	0,00
rl1889	2000	21	23,63	6,86	21	22,10	301,0	296,0	0,00
rl5915	2000	23	24,77	28,95	23	24,50	304,7	301,1	0,00
u1060	2000	12	14,43	2,13	12	12,80	300,5	283,6	0,00
u1432	2000	4	4,13	1,40	4	4,00	300,3	8,4	0,00
u1817	2000	1	1,00	0,39	1	1,00	300,4	4,6	0,00
u2152	2000	1	1,00	0,47	1	1,00	300,5	5,1	0,00
u2319	2000	4	4,13	2,46	4	4,00	300,9	13,1	0,00
vm1084	2000	17	19,17	3,56	17	17,10	300,5	140,8	0,00
vm1748	2000	22	24,03	6,57	22	22,67	300,9	270,1	0,00
melhores valores		43	16		60	65			
média				17,97				182,62	-3,64

Tabela 4: Resumo de desempenho Comparação entre CMSA_{PCCD} e Enhanced DR

	best	mean
CMSA _{PCCD} wins	26	53
DR wins	9	4
Ties	34	12

Uma análise dos resultados apresentados na Tabela 3 permite uma comparação objetiva entre os algoritmos avaliados. No que diz respeito à qualidade das soluções (coluna *best*), o CMSA_{PCCD} demonstrou desempenho igual ou superior ao da heurística Enhanced DR na maioria das instâncias. Especificamente, o CMSA_{PCCD} encontrou soluções estritamente melhores em 26 das 69 instâncias (aproximadamente 37,68%), enquanto foi superado pela heurística Enhanced DR em apenas 9 casos (cerca de 13,04%). Em 31 instâncias (49,27%), ambos os algoritmos apresentaram o mesmo resultado. Esse comportamento também se reflete na coluna *gap*, que apresenta valores negativos ou nulos em 86,95% das instâncias. Valores negativos indicam casos favoráveis ao CMSA_{PCCD}, valores nulos representam empates e valores positivos indicam vantagem para a heurística Enhanced DR.

Ao considerar a média dos resultados ao longo das 30 execuções, a superioridade do CMSA_{PCCD} torna-se ainda mais evidente. Em 53 das 69 instâncias (cerca de 76,81%), o CMSA_{PCCD} apresentou desempenho médio estritamente superior, enquanto o Enhanced DR obteve melhores médias em apenas 5 instâncias (aproximadamente 5,79%). Esses resultados destacam não apenas a capacidade do CMSA_{PCCD} de gerar soluções de alta qualidade, mas também sua robustez e consistência ao longo das execuções.

Em relação ao tempo de execução, o Enhanced DR demonstra uma vantagem significativa, sendo consideravelmente mais rápido. No entanto, por se tratar de uma heurística, sua limitação em escapar de ótimos locais impede que alcance soluções superiores mesmo se executasse por mais tempo. Ainda assim, o CMSA_{PCCD} mostrou-se competitivo em algumas instâncias, como, por exemplo, a dsj1000, em que os tempos de execução (*time to best*) foram superiores.

Além disso, os resultados revelam uma relação entre o desempenho dos algoritmos e o parâmetro de raio: raios menores requerem mais discos, tornando o problema mais difícil. Para valores de raio maiores ($r = 2000$), ambos os algoritmos alcançam resultados comparáveis, com empates em 20 das 23 instâncias, considerando a métrica *best*. Para valores de raio intermediários ($r = 1000$), a competição torna-se mais equilibrada: o CMSA_{PCCD} demonstra vantagens em instâncias que requerem mais discos (dsj1000, pr1002, rl1323), enquanto o Enhanced DR ainda prevalece em

outras (nrw1379, pcb1173). Para valores de raio pequenos ($r = 500$), o $CMSA_{PCCD}$ exibe superioridade significativa, obtendo os melhores resultados na maioria das instâncias.

6. Considerações finais

Este trabalho propõe a matheurística $CMSA_{PCCD}$ para o Problema de Cobertura Contínua de Discos de Raio Fixo (PCCD). O algoritmo híbrido combina a geração de componentes baseada em subdivisão aleatória com um resolvidor exato para encontrar soluções de alta qualidade. Destaca-se, ainda, que, segundo o conhecimento dos autores, é primeira vez aplica-se uma KD-Tree para esse problema. Foram realizados experimentos computacionais em um conjunto amplo de instâncias adaptadas do *TSP Challenge*, comparando-se o desempenho do $CMSA_{PCCD}$ com a heurística *Enhanced DR* [Chen et al., 2024]. Conforme apresentado nas Tabelas 3 e 4, o $CMSA_{PCCD}$ superou ou igualou a *Enhanced DR* em 60 das 69 instâncias avaliadas — um desempenho equivalente a 86,95% dos casos. Apesar dos resultados promissores, a abordagem apresenta algumas limitações. A eficiência do algoritmo depende do desempenho do resolvidor ILP, que pode se tornar um gargalo em grandes instâncias. Como trabalhos futuros, propõe-se a investigação de heurísticas construtivas alternativas na fase Construção, a incorporação de técnicas de busca local para refinamento das soluções e a avaliação de métodos exatos ou heurísticos distintos na fase de Adaptação.

Referências

- Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517. ISSN 0001-0782. URL <https://doi.org/10.1145/361002.361007>.
- Biniaz, A., Liu, P., Maheshwari, A., e Smid, M. (2017). Approximation algorithms for the unit disk cover problem in 2d and 3d. *Computational Geometry*, 60:8–18. ISSN 0925-7721. URL <https://www.sciencedirect.com/science/article/pii/S092577211630030X>. The Twenty-Seventh Canadian Conference on Computational Geometry August 2015.
- Blum, C., Pinacho, P., López-Ibáñez, M., e Lozano, J. A. (2016). Construct, merge, solve & adapt: A new general algorithm for combinatorial optimization. *Computers & Operations Research*, 68: 75–88. ISSN 0305-0548.
- Brönnimann, H. e Goodrich, M. T. (1995). Almost optimal set covers in finite vc-dimension. *Discrete & Computational Geometry*, 14(4):463–479. ISSN 1432-0444. URL <https://doi.org/10.1007/BF02570718>.
- Charikar, M., Chekuri, C., Feder, T., e Motwani, R. (2004). Incremental clustering and dynamic information retrieval. *SIAM Journal on Computing*, 33(6):1417–1440. URL <https://doi.org/10.1137/S0097539702418498>.
- Chen, X., Gao, X., Li, C., e and, C. P. (2024). A new heuristic for the continuous fixed-radius disc cover problem. *Engineering Optimization*, 0(0):1–14. URL <https://doi.org/10.1080/0305215X.2024.2368609>.
- Fowler, R. J., Paterson, M., e Tanimoto, S. L. (1981). Optimal packing and covering in the plane are np-complete. *Information Processing Letters*, 12(3):133–137. URL <https://lens.org/129-301-324-209-992>.

- Franceschetti, M., Cook, M., e Bruck, J. (2001). A geometric theorem for approximate disk covering algorithms. Technical report, California Institute of Technology, Pasadena, CA. URL https://www.researchgate.net/publication/2328060_A_Geometric_Theorem_for_Approximate_Disk_Covering_Algorithms.
- Friederich, R., Ghosh, A., Graham, M., Hicks, B., e Shevchenko, R. (2023). Experiments with unit disk cover algorithms for covering massive pointsets. *Computational Geometry*, 109:101925–101925. URL <https://arxiv.org/abs/2205.01716>.
- Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701. ISSN 01621459, 1537274X. URL <http://www.jstor.org/stable/2279372>.
- Gonzalez, T. F. (1991). Covering a set of points in multidimensional space. *Information Processing Letters*, 40(4):181–188. URL <http://www.sciencedirect.com/science/article/pii/002001909190075S>.
- Hale, T. S. e Moberg, C. R. (2003). Location science research: A review. *Annals of Operations Research*, 123(1):21–35. ISSN 1572-9338. URL <https://doi.org/10.1023/A:1026110926707>.
- Hochbaum, D. S. e Maass, W. (1985). Approximation schemes for covering and packing problems in image processing and vlsi. *Journal of the ACM*, 32(1):130–136. URL <https://dl.acm.org/doi/10.1145/2455.214106>.
- Hu, N. (2013). Approximation algorithms for geometric covering problems for disks and squares. Master of mathematics thesis, University of Waterloo, Waterloo, Ontario, Canada.
- Liaw, C., Liu, P., e Reiss, R. (2018). Approximation schemes for covering and packing in the streaming model. In Durocher, S. e Kamali, S., editors, *Proceedings of the 30th Canadian Conference on Computational Geometry, CCCG 2018, August 8-10, 2018, University of Manitoba, Winnipeg, Manitoba, Canada*, p. 172–179. URL <http://www.cs.umanitoba.ca/%7Ecccg2018/papers/session4A-p3.pdf>.
- Liu, P. e Lu, D. (2014). A fast 25/6-approximation for the minimum unit disk cover problem. URL <https://arxiv.org/abs/1406.3838>.
- López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., e Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Oper. Res. Perspect.*, 3:43–58. ISSN 2214-7160. URL <https://www.sciencedirect.com/science/article/pii/S2214716015300270>.
- The CGAL Project (2024). *CGAL User and Reference Manual*. CGAL Editorial Board, 6.0.1 edition. URL <https://doc.cgal.org/6.0.1/Manual/packages.html>.
- Vazirani, V. V. (2003). *Set Cover via Dual Fitting*, p. 108–117. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-662-04565-7. URL https://doi.org/10.1007/978-3-662-04565-7_13.