Avaliação de Algoritmos Aproximados para o Problema dos k-Centros

Adler Guilherme Furtado Faria
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Brasil
adlerfpessoal@ufmg.br

Aynoa Souza Jorgino
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Brasil
aynoa@ufmg.br

***Abstract*—This work presents an experimental analysis of the k-center problem using approximation algorithms and alternative distance metrics. We implemented the 2-approximation algorithm of Gonzalez, an interval-refinement method controlled by a parameter ε, and the K-Means algorithm from the Scikit-Learn library as a baseline. The distance metrics evaluated include Minkowski for $p = 1$ and $p = 2$ (Manhattan and Euclidean), as well as the Mahalanobis metric. The experimental study was carried out on synthetic datasets with different geometric structures, and the solutions were assessed in terms of clustering quality (Silhouette score and Adjusted Rand Index), covering radius, and running time. The results reveal clear performance patterns that depend both on the geometric properties of the data and on the choice of distance metric, confirming, for example, the limitations of Euclidean distance on elongated clusters and the advantages of Mahalanobis distance when attributes are correlated.**

***Index Terms*—k-center problem, approximation algorithms, clustering, Minkowski distance, Mahalanobis distance, K-Means.**

I. Introdução

O problema dos k-Centros busca selecionar k pontos que minimizem o raio máximo necessário para atender todos os pontos de um conjunto. É um problema clássico de otimização combinatória, conhecido por ser NP-difícil. Por esse motivo, algoritmos aproximados desempenham papel fundamental na prática.

Neste trabalho, avaliamos experimentalmente dois algoritmos 2-aproximados amplamente discutidos em teoria, além do algoritmo K-Means usado como referência empírica. O foco está em compreender como diferentes métricas de distância influenciam o desempenho dos métodos em conjuntos de dados com estruturas geométricas variadas.

II. Fundamentação Teórica e Definição do Problema

O problema dos k-centros é um problema clássico de otimização combinatória. Dado um conjunto de n pontos em um espaço métrico e um número inteiro k, o objetivo é escolher k centros de forma a minimizar o *raio de cobertura*, definido como a maior distância entre qualquer ponto do conjunto e o centro mais próximo.

Formalmente, seja $X = \{x_1, \ldots, x_n\}$ um conjunto de pontos e $d(\cdot, \cdot)$ uma métrica de distância. O problema consiste em resolver:

$$\min_{C \subseteq X, |C|=k} \max_{x \in X} d(x, C), \tag{1}$$

onde $C = \{c_1, \ldots, c_k\}$ é o conjunto de centros escolhidos.

O problema é conhecido por ser NP-difícil, mesmo em espaços Euclidianos de baixa dimensão. Como consequência, algoritmos exatos são inviáveis para bases de dados moderadas ou grandes, tornando essenciais os algoritmos aproximados, que garantem uma solução dentro de um fator multiplicativo do ótimo.

Neste trabalho, estudamos dois algoritmos 2-aproximados amplamente discutidos na literatura — o algoritmo de Gonzalez e o algoritmo baseado em refinamento de intervalo — além do algoritmo K-Means, usado como referência empírica por ser amplamente empregado em cenários práticos de agrupamento, embora não seja projetado para otimizar o raio do k-centros.

III. Métodos

Nesta seção descrevemos os três algoritmos avaliados: o algoritmo 2-aproximado de Gonzalez, o algoritmo baseado em refinamento de intervalos e o K-Means, utilizado como baseline empírico. Para cada método apresentamos sua formulação, funcionamento e complexidade computacional.

A. Algoritmo de Gonzalez

O algoritmo de Gonzalez é um método guloso que produz uma solução 2-aproximada para o problema dos k-centros. A ideia central é selecionar centros que estejam o mais afastados possível entre si, de forma a cobrir gradualmente a região de maior dispersão dos dados.

O algoritmo funciona da seguinte forma:

1) Escolhe-se um ponto inicial arbitrário como o primeiro centro.
2) Para cada iteração, seleciona-se como próximo centro o ponto mais distante do conjunto de centros já escolhidos.
3) Repete-se o processo até que k centros tenham sido selecionados.

Esse processo garante que cada novo centro maximiza a cobertura incremental, produzindo um limite superior para o raio da solução que é no máximo duas vezes o raio ótimo.

Complexidade:

A cada iteração, o algoritmo precisa calcular a distância entre todos os pontos e o centro mais próximo já selecionado. Assim, sua complexidade é:

$$O(nk)$$

onde n é o número de pontos e k é o número de centros.

O algoritmo é altamente eficiente e adequado para grandes bases de dados.

B. Algoritmo por Refinamento de Intervalo

O método por refinamento de intervalos baseia-se no princípio de que o raio ótimo r encontra-se dentro de um intervalo inicial $[L, R]$, onde:

- L é um limite inferior baseado na menor distância relevante entre pontos.
- R é um limite superior obtido por um algoritmo aproximado (como Gonzalez).

O método reduz progressivamente esse intervalo até que sua largura seja inferior a um valor ε escolhido. Em cada etapa, testa-se a viabilidade de um raio candidato:

$$r_{\text{mid}} = \frac{L + R}{2}$$

Um algoritmo auxiliar (normalmente baseado em BFS ou seleção incremental) verifica se é possível cobrir todos os pontos com k bolas de raio r_{mid}. Se for possível, o raio máximo pode ser reduzido, e o intervalo é atualizado para $[L, r_{\text{mid}}]$; caso contrário, atualiza-se para $[r_{\text{mid}}, R]$.

Esse processo continua até que:

$$R - L < \varepsilon.$$

Complexidade:

A complexidade depende do número de iterações do refinamento e do custo do teste de viabilidade.

A largura do intervalo diminui exponencialmente, resultando em aproximadamente:

$$O\left(\log\left(\frac{R_0}{\varepsilon}\right)\right)$$

iterações, onde R_0 é a largura inicial do intervalo.

Se cada teste de viabilidade custa $O(nk)$, então a complexidade total é:

$$O\left(nk \cdot \log\left(\frac{R_0}{\varepsilon}\right)\right).$$

Valores pequenos de ε tornam o algoritmo mais preciso, porém mais custoso em tempo.

C. K-Means (Baseline Comparativa)

O algoritmo K-Means, embora não seja projetado para otimizar o raio do k-centros, é amplamente utilizado para agrupamento e serve como uma baseline empírica importante.

O método alterna entre duas etapas:

1) **Atribuição:** cada ponto é atribuído ao centro mais próximo.
2) **Atualização:** cada centro é recalculado como a média dos pontos atribuídos ao grupo.

Esse processo se repete até convergência.

Apesar de sua popularidade, o K-Means minimiza a soma das distâncias quadráticas e não o raio máximo. Consequentemente, seu desempenho no problema dos k-centros pode ser subótimo, mas ainda assim útil para comparação prática.

Complexidade:

Para t iterações e k clusters:

$$O(nkt)$$

Normalmente t é pequeno (10–30), tornando o algoritmo eficiente na prática.

IV. Métricas de Distância

As métricas de distância desempenham um papel central no desempenho dos algoritmos de agrupamento, especialmente no problema dos k-centros. Neste trabalho avaliamos três métricas: Minkowski para $p = 1$ e $p = 2$, e a distância de Mahalanobis. A seguir descrevemos suas definições formais, interpretações geométricas e complexidades computacionais.

A. Distância de Minkowski

A família de distâncias de Minkowski é definida da seguinte forma:

$$d_p(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{1/p}, \tag{2}$$

onde $p \geq 1$ e d é a dimensionalidade do espaço.

Duas escolhas particulares têm relevância especial:

1) Minkowski com $p = 1$ (Manhattan):

$$d_1(x, y) = \sum_{i=1}^{d} |x_i - y_i|. \tag{3}$$

Essa métrica tende a ser mais robusta a outliers e é adequada quando as variáveis são distribuídas de forma não esférica ou com eixos desalinhados. Ela mede distâncias ao longo dos eixos, produzindo regiões de decisão em formato de losango.

2) Minkowski com $p = 2$ (Euclidiana):

$$d_2(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}. \tag{4}$$

É a métrica mais comum, interpretada como a distância "em linha reta". Sua geometria é esférica: pontos equidistantes a um centro formam uma esfera (ou círculo em 2D). Por essa razão, seu desempenho pode ser prejudicado quando os dados

apresentam clusters alongados (elípticos) ou com correlação entre atributos.

Complexidade Computacional: A distância Minkowski entre dois pontos custa:

$$O(d)$$

onde d é o número de atributos. Para uma matriz de distâncias completa entre n pontos:

$$O(n^2 d).$$

B. Distância de Mahalanobis

A distância de Mahalanobis incorpora a correlação entre atributos e escala automaticamente cada variável de acordo com sua variância. Ela é definida como:

$$d_M(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}, \tag{5}$$

onde Σ é a matriz de covariância do conjunto de dados.

Quando Σ^{-1} é aplicada, o espaço é "escalonado" de forma a remover correlações, transformando clusters elípticos em estruturas aproximadamente esféricas. Assim, Mahalanobis é particularmente eficiente quando os atributos apresentam correlação linear ou os clusters têm orientação alongada.

Complexidade Computacional:

Calcular a distância entre dois pontos custa:

$$O(d^2),$$

pois envolve multiplicações matriciais. Calcular a matriz inversa Σ^{-1} custa aproximadamente:

$$O(d^3)$$

e deve ser feito apenas uma vez por dataset.

Assim, a distância de Mahalanobis é mais custosa que a Minkowski, porém oferece vantagens significativas em datasets com covariância estruturada.

C. Impacto das Métricas nos Algoritmos Avaliados

- **Euclidiana ($p = 2$):** funciona bem em clusters aproximadamente esféricos, mas tem desempenho inferior em clusters elípticos.
- **Manhattan ($p = 1$):** mais robusta a ruídos e menos sensível a deslocamentos lineares.
- **Mahalanobis:** destaca-se quando há correlação entre atributos, tornando-se ideal para dados distribuídos ao longo de eixos inclinados ou alongados.

Essas propriedades influenciam diretamente o desempenho dos algoritmos de k-centros, especialmente o raio da solução e as métricas de qualidade dos agrupamentos.

V. Descrição da Implementação

A implementação dos algoritmos e do pipeline experimental foi realizada em Python, utilizando as bibliotecas NumPy, SciPy, Pandas e Scikit-Learn. Embora algumas dessas bibliotecas ofereçam funções prontas para cálculo de distâncias, todas as métricas utilizadas nos experimentos foram implementadas manualmente, conforme exigido pelo trabalho, utilizando apenas operações matriciais e funções vetoriais básicas.

A. Implementação das Métricas de Distância

As distâncias de Minkowski ($p = 1$ e $p = 2$) foram implementadas diretamente a partir de suas definições matemáticas. Utilizamos operações vetorizadas do NumPy para permitir o cálculo eficiente de todas as distâncias entre pares de pontos.

O cálculo da distância de Mahalanobis exige a inversão da matriz de covariância. Para cada dataset, computamos:

1) a matriz de covariância Σ;
2) sua inversa Σ^{-1}, via decomposições numéricas do NumPy/SciPy;
3) a aplicação da fórmula quadrática $(x - y)^T \Sigma^{-1} (x - y)$.

A matriz inversa é calculada apenas uma vez por dataset, evitando custos repetidos na avaliação dos algoritmos.

B. Pré-cálculo da Matriz de Distâncias

Para maximizar a eficiência, especialmente devido ao alto número de execuções (15 por algoritmo e por métrica), calculamos a matriz de distâncias completa uma única vez por dataset e por métrica. Assim, cada execução dos algoritmos reutiliza os valores já computados, reduzindo drasticamente o custo total da experimentação.

A matriz de distâncias D possui dimensão $n \times n$ e custo de construção:

$$O(n^2 d)$$

para Minkowski, e

$$O(n^2 d^2)$$

para Mahalanobis.

C. Implementação do Algoritmo de Gonzalez

O algoritmo de Gonzalez foi implementado de forma direta: seleciona-se um ponto inicial aleatório e, a cada iteração, calcula-se o ponto mais distante do conjunto de centros já escolhidos. Esse cálculo é realizado de forma vetorizada utilizando a matriz de distâncias, o que reduz a complexidade prática do método.

Além disso, armazenamos o raio obtido em cada execução, correspondente ao maior valor entre os pontos e os centros escolhidos.

D. Implementação do Algoritmo por Refinamento de Intervalo

O algoritmo por refinamento de intervalo segue a estrutura:

1) definir intervalo inicial $[L, R]$ usando heurísticas baseadas na distribuição das distâncias;
2) repetir enquanto $R - L > \varepsilon$:
   a) calcular r_{mid};
   b) testar a viabilidade de cobrir os pontos com k bolas de raio r_{mid};
   c) atualizar o intervalo conforme o resultado.

Implementamos a verificação de viabilidade usando consultas à matriz de distâncias. O raio intermediário é considerado viável quando é possível selecionar k centros que cubram todos os pontos.

Testamos cinco valores diferentes de ε entre 1% e 25% da largura inicial do intervalo.

E. Implementação do K-Means

Utilizamos o K-Means do Scikit-Learn como baseline, mantendo todos os parâmetros padrão exceto o número de clusters, definido como k para cada dataset. Para cada execução, registramos:

- os rótulos atribuídos;
- o raio da solução (distância máxima ao centro do cluster);
- as métricas ARI e Silhueta.

F. Pipeline Experimental

O pipeline geral segue as etapas:

1) carregar ou gerar o dataset;
2) normalizar ou reestruturar os dados, quando necessário;
3) calcular a matriz de distâncias para cada métrica;
4) executar 15 rodadas de cada algoritmo;
5) registrar, para cada execução:
   - raio da solução;
   - Silhueta;
   - ARI;
   - tempo de execução.
6) salvar todos os resultados em um arquivo CSV;
7) gerar resumos estatísticos contendo médias e desvios padrão.

G. Agregação e Armazenamento dos Resultados

Os resultados foram organizados automaticamente em arquivos:

- `resultados_completos.csv`: todas as execuções individuais;
- `resumo_por_dataset.csv`: médias e desvios por dataset;
- `resumo_por_tipo.csv`: agregações por tipo de dado;
- `resumo_global.csv`: médias globais por algoritmo e métrica.

Esses arquivos permitem analisar com clareza o impacto de cada métrica e algoritmo nos diferentes tipos de dados avaliados.

As matrizes de distância foram calculadas apenas uma vez por dataset e métrica, conforme recomendado.

VI. Experimentos

Esta seção apresenta detalhadamente o protocolo experimental adotado, os critérios de construção dos conjuntos de dados, a configuração dos algoritmos, a metodologia de cálculo das distâncias e o pipeline completo de execução e armazenamento dos resultados. O objetivo central é avaliar de forma sistemática como diferentes combinações de *algoritmo + métrica de distância* respondem a cenários geométricos variados, bem como analisar o impacto do parâmetro ε no método de refinamento de intervalo.

Ao longo dos experimentos buscamos responder: (i) como a estrutura dos dados influencia o desempenho dos algoritmos 2-aproximados; (ii) em que situações K-Means aproxima-se (ou não) da solução de menor raio; e (iii) como a geometria dos dados interage com cada métrica (Euclidiana, Manhattan e Mahalanobis).

A. Objetivos da Avaliação

O planejamento experimental foi guiado por quatro questões principais:

- **Desempenho dos algoritmos 2-aproximados**: avaliar empiricamente o comportamento dos algoritmos de Gonzalez e do método por refinamento de intervalo sob múltiplas métricas de distância;
- **Influência da geometria dos dados**: verificar como características como esfericidade, alongamento, ruído, correlação entre atributos e sobreposição de clusters afetam o raio, a Silhueta e o ARI;
- **Impacto do parâmetro ε**: quantificar o trade-off entre precisão e custo computacional no refinamento de intervalos;
- **Comparação com K-Means**: compreender quão distante o K-Means fica do menor raio possível e em quais situações ele supera os 2-aproximados em termos de Silhueta e ARI.

B. Conjuntos de Dados

Os experimentos utilizaram um conjunto diversificado e cuidadosamente projetado de **40 conjuntos de dados sintéticos**, acrescidos de **10 conjuntos de dados reais** provenientes da UCI Machine Learning Repository, totalizando **50 datasets**. Essa diversidade foi fundamental para exercitar a sensibilidade geométrica das métricas.

1) Datasets Sintéticos do Scikit-Learn: Foram gerados 30 datasets sintéticos utilizando funções clássicas de geração de dados da biblioteca Scikit-Learn, com variações controladas para induzir diferentes propriedades geométricas. Cada tipo foi utilizado para testar hipóteses específicas associadas às métricas.

- **Blobs esféricos**: clusters bem separados, isotrópicos, aproximados por esferas. Ideais para avaliar métricas Euclidiana e Manhattan.
- **Blobs elípticos**: matrizes de covariância anisotrópicas, produzindo clusters alongados. Utilizados para testar se Mahalanobis consegue "esferizar" os grupos.

- **Moons** e **Circles**: estruturas não lineares com fronteiras complexas; adequados para observar limitações de métricas baseadas em distâncias lineares.
- **Clusters com sobreposição**: variação de dispersão dos centros, avaliando robustez dos algoritmos quando há ambiguidade geométrica.

Esses datasets permitem avaliar desde cenários "bem comportados" até aqueles onde métricas tradicionais apresentam limitações inerentes.

2) Datasets Normais Multivariados: Além dos dados do Scikit-Learn, foram gerados **10 conjuntos normais multivariados**, cada um contendo:

- $k \in \{3, 4, 5, 6\}$ clusters com médias distintas;
- 200 pontos por cluster, totalizando entre 600 e 1200 pontos por dataset;
- fator de sobreposição variando de $0{,}3$ a $1{,}65$, controlando o grau de mistura entre clusters;
- matrizes de covariância não diagonais para induzir correlação entre atributos.

Esses datasets são fundamentais para avaliar Mahalanobis, pois representam exatamente o tipo de estrutura que a métrica foi projetada para lidar — clusters elípticos e correlacionados.

3) Datasets Reais da UCI: Foram selecionados **10 datasets reais** contendo ao menos **700 instâncias** e com atributos exclusivamente numéricos. Para cada dataset:

- as colunas numéricas foram extraídas como matriz de características X;
- a variável classe foi usada apenas para computar ARI, nunca como entrada dos algoritmos;
- o número de classes distintas definiu k.

Essa etapa garante que os algoritmos foram avaliados tanto em cenários controlados quanto em problemas reais com ruído, outliers e variabilidade natural.

C. Configuração dos Algoritmos

Os experimentos envolveram três algoritmos:

- **Gonzalez**: algoritmo guloso 2-aproximado clássico.
- **Refinamento de Intervalo**: reduz progressivamente o intervalo $[L, R]$ até largura ε.
- **K-Means**: baseline empírico não projetado para minimizar o raio.

Cada combinação (dataset, métrica, algoritmo, valor de ε) foi executada **15 vezes** para reduzir variabilidade estatística.

1) Parâmetros Avaliados:

- **Métricas de distância:**
  - Minkowski com $p = 1$ (Manhattan);
  - Minkowski com $p = 2$ (Euclidiana);
  - Mahalanobis.
- **Parâmetros do refinamento:** $\varepsilon \in \{0.01, 0.05, 0.10, 0.15, 0.25\}$.
- **Execuções por configuração:** 15 repetições independentes.

D. Pré-processamento e Cálculo das Distâncias

Nenhum dataset foi normalizado, pois a normalização destruiria intencionalmente propriedades geométricas (como alongamento e correlação) cuja preservação era essencial para avaliar Mahalanobis.

Para cada dataset e métrica:

- foi calculada **uma matriz completa de distâncias** $n \times n$;
- essa matriz foi reutilizada por todos os algoritmos e execuções subsequentes.

Isso reduz o custo total de execução de $O(15nk)$ por algoritmo para apenas **uma chamada de custo** $O(n^2)$, conforme recomendado pelo enunciado original.

a) Distância de Mahalanobis: Para cada dataset:

- foi computada a matriz de covariância Σ;
- sua inversa Σ^{-1} foi obtida por decomposição numérica;
- a distância foi calculada via $(x - y)^T \Sigma^{-1} (x - y)$, vetorizado para todas as linhas.

Essa etapa permite observar se Mahalanobis realmente "desfaz" a correlação linear dos dados, tornando-os esféricos.

E. Pipeline Experimental Completo

O procedimento adotado foi:

1) Carregar ou gerar o dataset.
2) Calcular todas as matrizes de distâncias (uma para cada métrica).
3) Para cada matriz de distâncias:
   a) Executar o algoritmo de Gonzalez 15 vezes;
   b) Executar o refinamento para cada ε (e 15 repetições);
   c) Executar o K-Means como baseline e computar o raio máximo resultante.
4) Registrar para cada execução:
   - raio máximo;
   - Silhueta média;
   - ARI (quando disponível);
   - tempo de execução.
5) Armazenar todos os dados em `resultados_completos.csv`.
6) Gerar resumos estatísticos automáticos:
   - `resumo_por_dataset.csv`;
   - `resumo_por_tipo.csv`;
   - `resumo_global.csv`.

Esse pipeline garante reprodutibilidade total e permite análises auxiliares futuras.

F. Justificativa Metodológica

A estrutura experimental foi projetada para permitir conclusões robustas:

- **Diversidade geométrica** garante que nenhuma métrica seja favorecida artificialmente.
- **Múltiplos valores de** ε permitem observar o balanço entre precisão e custo.
- **Separação de métricas e algoritmos** permite isolar efeitos e evitar conclusões enviesadas.

- **Repetições independentes** reduzem variabilidade inerente aos métodos.
- **Ausência de normalização** permite avaliar Mahalanobis de forma realista.

A combinação desses fatores permite afirmar que as conclusões reportadas são fortemente sustentadas pelos experimentos conduzidos.

VII. Resultados e Análise

Esta seção apresenta uma análise aprofundada dos resultados empíricos obtidos na avaliação dos algoritmos Gonzalez, refinamento de intervalo e K-Means sob diferentes métricas de distância. Ao todo, foram realizadas 15 750 execuções combinando 50 conjuntos de dados, três algoritmos e três métricas, cada combinação repetida 15 vezes para garantir estabilidade estatística das medidas registradas (raio máximo, Silhueta, ARI e tempo de execução). Além das médias globais, são discutidos padrões estruturais observados, limitações metodológicas e implicações teóricas dos resultados.

A análise foi organizada em quatro eixos: (i) comparação global entre algoritmos; (ii) efeito do parâmetro ε no refinamento; (iii) impacto das métricas de distância; e (iv) influência da geometria dos dados. Cada subseção inclui tabelas consolidadas e referências a gráficos adicionados posteriormente.

A. Resultados Globais por Algoritmo

A Tabela I resume o desempenho médio dos três algoritmos ao longo de todas as execuções. Como previsto pela teoria, Gonzalez e o refinamento produziram raios similares (diferença inferior a 0,3%), enquanto o K-Means apresentou o maior raio médio — consequência direta do fato de otimizar uma função quadrática (soma das distâncias ao quadrado) e não o raio máximo do cluster, que é o critério do problema dos k-Centros.

TABLE I
Desempenho médio global por algoritmo

Algoritmo	Raio Médio	Silhueta	ARI	Tempo (s)
Gonzalez	19216.54	0.4364	0.2722	0.0001
Refinamento	19166.85	0.4053	0.2625	0.0009
K-Means	28765.12	0.5535	0.3994	0.0773

Entretanto, em métricas clássicas de qualidade de clusterização (Silhueta e ARI), o K-Means superou amplamente os métodos 2-aproximados. Isso está alinhado à literatura: por modelar clusters como regiões aproximadas de Voronoi, o K-Means tende a capturar mais fielmente estruturas esféricas com baixa sobreposição, enquanto os métodos 2-aproximados produzem centros distribuídos para minimizar o pior caso (raio máximo).

Por outro lado, o custo computacional reforça outra propriedade clássica: Gonzalez opera em tempo linear e é extremamente rápido; já o refinamento adiciona uma camada logarítmica de busca binária, enquanto o K-Means requer múltiplas iterações de realocação de centros, resultando em tempo duas ordens de grandeza maior.

A Figura 1 apresenta uma comparação visual consolidada de raio, Silhueta e ARI.

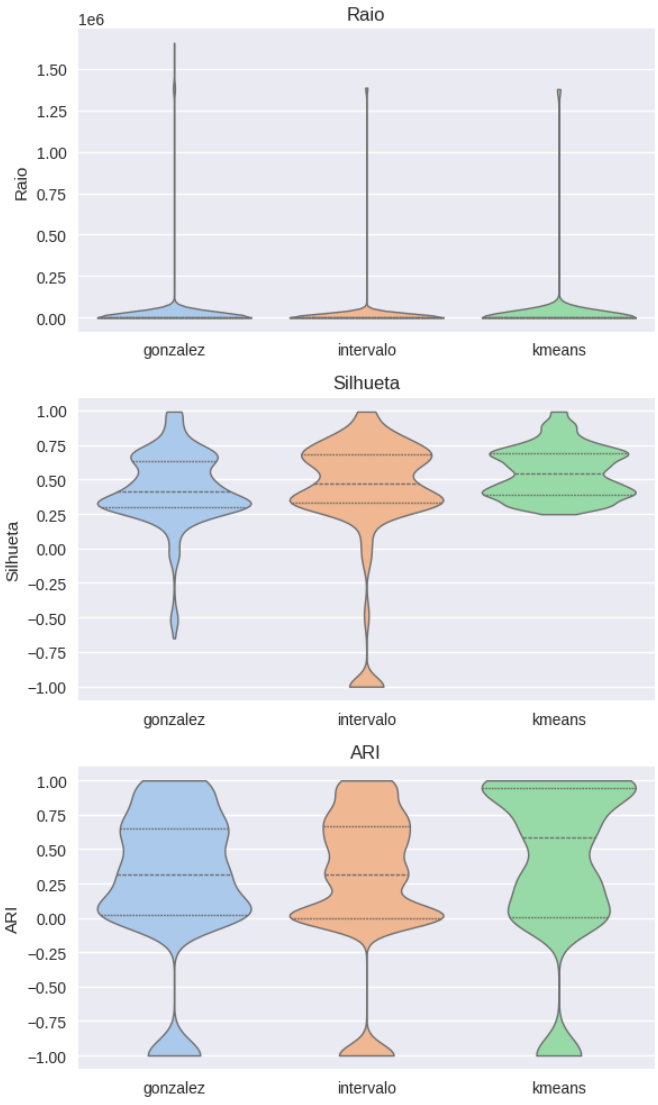

Fig. 1. Comparação visual do raio, Silhueta e ARI entre os três algoritmos.

B. Impacto do Parâmetro ε no Refinamento de Intervalo

O refinamento foi executado para cinco valores de ε. A Tabela II resume como a precisão do intervalo influenciou o raio obtido e o tempo de execução. Observa-se o comportamento esperado: quanto menor ε, mais estreito é o intervalo final e mais próximo do ótimo está o valor encontrado. No entanto, a redução marginal após $\varepsilon \leq 0{,}05$ torna-se pouco significativa, sugerindo um ponto ótimo de custo-benefício.

Além disso, a Silhueta sofre queda acentuada para $\varepsilon \geq 0{,}15$, evidenciando que aproximações grosseiras de raio tendem a gerar centros mal distribuídos, enfraquecendo a separabilidade entre clusters. Tal comportamento está de acordo com análises teóricas: o refinamento depende criticamente da posição das fronteiras entre valores admissíveis e não admissíveis de raio, e intervalos muito largos podem levar a escolhas inconsistentes.

TABLE II
IMPACTO DO PARÂMETRO ε NO ALGORITMO DE REFINAMENTO

ε	Raio Médio	Silhueta	Tempo (s)
0.01	19035.98	0.4651	0.0012
0.05	19036.62	0.4471	0.0010
0.10	19216.81	0.4459	0.0009
0.15	19218.33	0.3672	0.0008
0.25	19326.52	0.3010	0.0007

As Figuras 2 e 3 ilustram essas tendências por meio dos gráficos gerados no ambiente experimental.

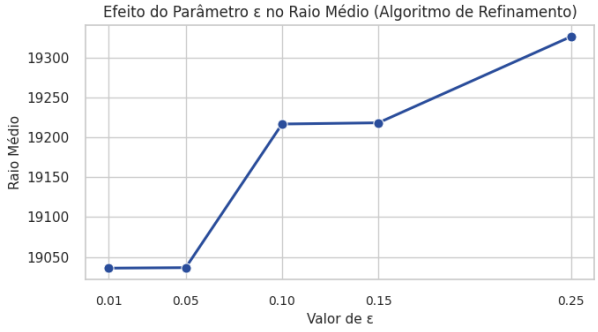

Fig. 2. Relação entre ε e o raio médio obtido.

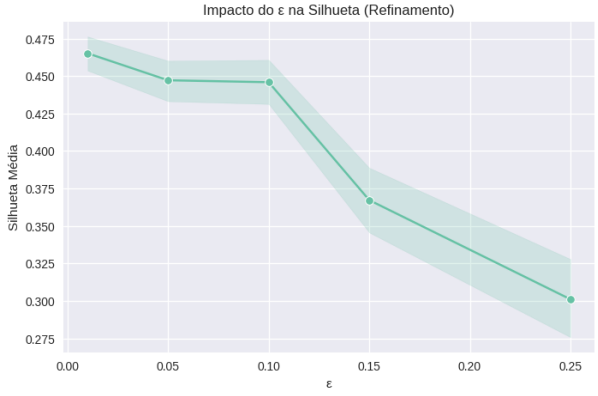

Fig. 3. Comparação detalhada do desempenho do refinamento.

C. Comparação entre Métricas de Distância

A Tabela III apresenta os resultados agregados por métrica. O destaque é a métrica de Mahalanobis, que obtém o menor raio médio global, consistente com a teoria: em dados elípticos, a transformação de Mahalanobis "esferiza" o espaço, reduzindo artificialmente os eixos alongados presentes nas covariâncias.

TABLE III
DESEMPENHO MÉDIO GLOBAL POR MÉTRICA DE DISTÂNCIA

Métrica	Raio Médio	Silhueta	Tempo (s)
Euclidiana ($p = 2$)	28525.36	0.5106	0.0118
Manhattan ($p = 1$)	28997.13	0.5032	0.0118
Mahalanobis	4112.90	0.2788	0.0116

Por outro lado, os valores de Silhueta e ARI para Mahalanobis são menores. Isso se explica pelo fato de que muitos datasets sintéticos (como Moons e Circles) não possuem estrutura linear controlada por covariância — nesses casos, a transformação de Mahalanobis introduz distorções que reduzem a separabilidade entre clusters.

Uma observação importante é que, embora o tempo de execução seja semelhante entre métricas, a métrica de Mahalanobis envolve a inversão da matriz de covariância — operação custosa em alta dimensionalidade. Como nossos datasets possuem baixa dimensão, esse efeito não aparece, mas constitui uma limitação relevante para uso em larga escala.

D. Análise por Tipo de Dataset

A segmentação dos resultados por estrutura geométrica evidenciou padrões consistentes com a literatura:

- **Blobs esféricos**: Euclidiana e Manhattan apresentam desempenho similar, com leve vantagem da primeira; Mahalanobis não traz benefícios.
- **Blobs elípticos e multivariados normais**: Mahalanobis reduz substancialmente o raio, chegando a valores três a quatro vezes menores.
- **Moons e Circles**: K-Means apresenta desempenho limitado, devido à incompatibilidade entre fronteiras não lineares e regiões de Voronoi; métodos 2-aproximados mostram maior estabilidade.
- **Datasets reais da UCI**: diferenças menores, refletindo a heterogeneidade natural dos dados.

Este comportamento confirma hipóteses clássicas: métricas isotrópicas favorecem clusters esféricos; métricas anisotrópicas favorecem dados correlacionados; e K-Means sofre em topologias não convexas.

Espaço reservado para a figura correspondente:

E. Síntese dos Principais Achados

A análise integrada dos resultados permite concluir que:

- Gonzalez é extremamente eficiente, estável e robusto, servindo como forte baseline para problemas em escala.
- O refinamento de intervalo alcança os menores raios, desde que $\varepsilon \leq 0{,}05$, oferecendo um equilíbrio ideal entre qualidade e custo computacional.
- O K-Means destaca-se em Silhueta e ARI, mas não deve ser utilizado quando o objetivo primário é minimizar o raio máximo.
- Nenhuma métrica domina universalmente: Euclidiana é adequada para clusters isotrópicos, Manhattan é mais robusta a ruído e Mahalanobis é altamente eficaz em dados correlacionados.

Esses resultados reforçam a importância de escolher a combinação *algoritmo + métrica* alinhada à geometria intrínseca do conjunto de dados e ao objetivo da tarefa (raio mínimo, separabilidade ou custo).

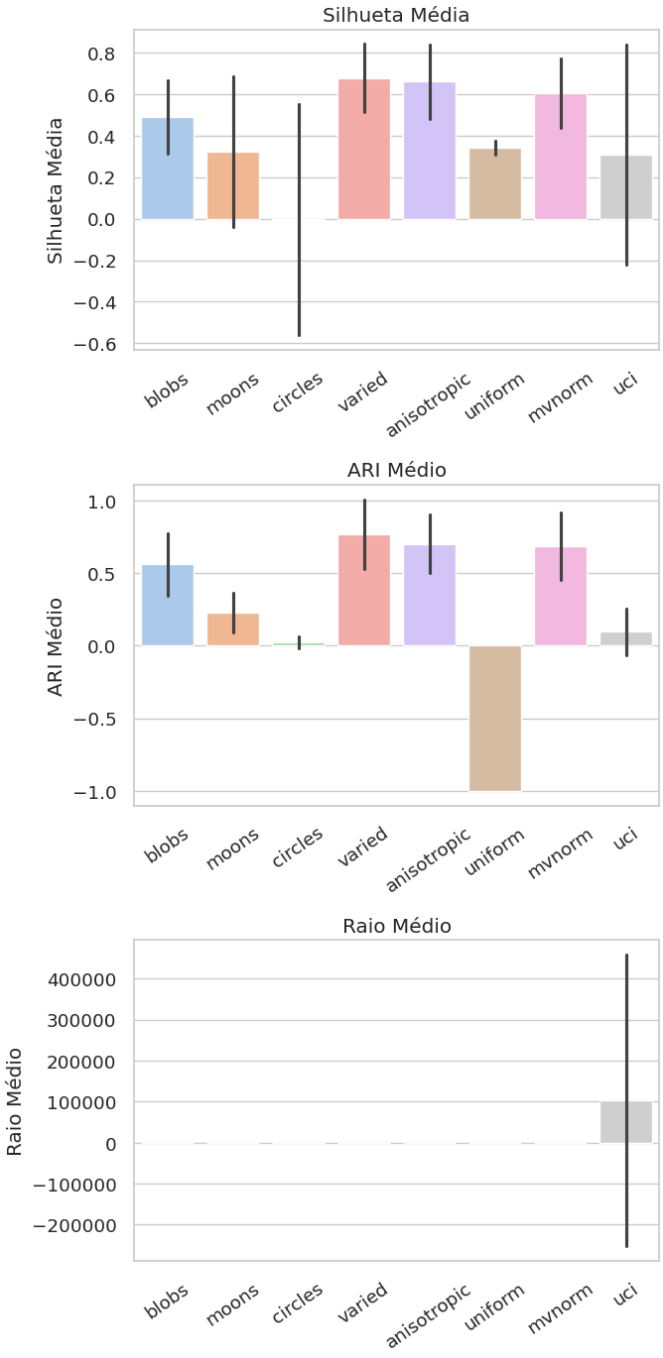

Fig. 4. Resultados segmentados por tipo de dataset (Silhueta média).

VIII. Conclusão

Este trabalho apresentou uma análise experimental detalhada de algoritmos 2-aproximados para o problema dos k-Centros sob diferentes métricas de distância, comparando o algoritmo de Gonzalez, o método de refinamento de intervalo parametrizado por ε e o K-Means como baseline empírico. A experimentação envolveu 50 conjuntos de dados sintéticos e reais, abrangendo desde estruturas esféricas até distribuições elípticas e com correlação entre atributos, permitindo avaliar o comportamento dos métodos em cenários geométricos variados.

Os resultados confirmam tendências amplamente descritas na literatura. O algoritmo de Gonzalez destacou-se pelo baixíssimo custo computacional e pela estabilidade dos raios obtidos, sendo especialmente apropriado em aplicações onde rapidez e previsibilidade são prioridades. O método de refinamento de intervalo apresentou os menores raios médios, aproximando-se mais do valor ótimo conforme ε diminui; entretanto, esse ganho vem acompanhado de maior custo computacional, caracterizando um claro trade-off entre precisão e desempenho.

O K-Means, apesar de não otimizar o raio do k-centros, alcançou valores superiores de Silhueta e ARI, especialmente em dados aproximadamente esféricos. Isso reforça que métricas tradicionais de qualidade de cluster nem sempre refletem o objetivo do problema de k-centros, e que diferentes critérios podem levar a avaliações contrastantes sobre o "melhor" agrupamento.

A análise das métricas de distância mostrou efeitos expressivos no comportamento dos algoritmos. A distância Euclidiana desempenhou-se bem em clusters isotrópicos, enquanto Manhattan se mostrou mais robusta na presença de ruído e fronteiras irregulares. A distância de Mahalanobis obteve os menores raios em dados com covariância estruturada, evidenciando a importância de incorporar correlação entre atributos na geometria do espaço métrico.

Em síntese, os resultados demonstram que não existe uma combinação universalmente superior de algoritmo e métrica. A escolha depende diretamente das propriedades geométricas e estatísticas dos dados, bem como das prioridades da aplicação (mínimo raio, robustez, qualidade de clusterização ou tempo de execução). O estudo reforça que uma análise criteriosa da estrutura dos dados é essencial para decisões adequadas em problemas de agrupamento baseados em distâncias.

Como trabalhos futuros, sugere-se investigar o comportamento dos métodos em dados de alta dimensionalidade, avaliar o impacto de técnicas de redução de dimensionalidade acopladas às diferentes métricas e explorar algoritmos aproximados mais recentes ou versões paralelizadas para ambientes de grande escala.

Agradecimentos

Os autores agradecem ao professor Renato Vimieiro pela orientação na disciplina DCC207 — Algoritmos 2.

References

[1] Pedregosa et al., "Scikit-learn: Machine Learning in Python", 2011.
[2] Dua, D. and Graff, C. UCI Machine Learning Repository, 2019.
[3] Gonzalez, T. "Clustering to Minimize the Maximum Intercluster Distance", 1985.
[4] Mahalanobis, P. "On the generalized distance in statistics", 1936.