Análise Empírica e Comparativa de Algoritmos 2-Aproximados para o Problema k-Center

Paula D'Agostini
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Brasil
pauladagostini@ufmg.br

Pedro Bernardo Goulart Parreira
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Brasil
pedrobgoulart@ufmg.br

***Resumo*—Este trabalho apresenta uma análise empírica abrangente de algoritmos 2-aproximados para o problema k-center, comparando suas performances com o algoritmo K-Means tradicional, que implementa uma abordagem distinta para o problema de agrupamento. Foram implementados e avaliados dois algoritmos aproximados (Gonzalez e Refinamento de Intervalos) utilizando diferentes métricas de distância (Manhattan, Euclidiana e Mahalanobis) em 50 datasets (10 reais da UCI e 40 sintéticos). Os resultados demonstram que enquanto o algoritmo de Gonzalez oferece melhor consistência e eficiência computacional, o método de refinamento proporciona maior controle sobre a precisão da solução, com ambos superando o K-Means na métrica específica de raio máximo para a qual foram projetados. A análise revela ainda que a métrica de Mahalanobis apresenta vantagens significativas em dados com estrutura de covariância não trivial, alcançando valores de ARI próximos a 1.0 em datasets sintéticos bem estruturados.**

***Index Terms*—k-center, algoritmos aproximados, agrupamento, comparação empírica, aprendizado de máquina, Gonzalez, refinamento, complexidade computacional**

I. Introdução

A. Contexto e Motivação

O problema k-center é um desafio clássico de otimização combinatorial que consiste em selecionar k centros de um conjunto de pontos de forma a minimizar a máxima distância de qualquer ponto ao centro mais próximo. Formalmente, dado um conjunto S de n pontos e um inteiro k, o objetivo é encontrar um conjunto $C \subseteq S$ com $|C| = k$ que minimize:

$$r(C) = \max_{s \in S} \min_{c \in C} dist(s, c) \tag{1}$$

Este problema é conhecido por ser NP-difícil, tornando algoritmos aproximados essenciais para aplicações práticas em áreas como facility location, agrupamento de dados e redes de sensores. A natureza de minimização do pior caso torna o k-center particularmente relevante para aplicações onde a qualidade de serviço deve ser garantida para todos os usuários ou pontos de dados, como no posicionamento de hospitais, estações de bombeiros ou centros de distribuição.

B. Relação entre k-Center e K-Means

É importante distinguir que, embora ambos os problemas envolvam a seleção de k centros, o problema k-center foca na minimização do pior caso (raio máximo), enquanto K-Means busca minimizar a soma dos quadrados das distâncias (inércia). Esta diferença fundamental resulta em algoritmos com propriedades teóricas e práticas distintas:

- **k-Center**: Minimização do raio máximo, garantindo cobertura uniforme para todos os pontos
- **K-Means**: Minimização da variância total, buscando clusters compactos em média

Esta distinção justifica a comparação empírica entre abordagens tão distintas, permitindo avaliar trade-offs entre diferentes critérios de qualidade em agrupamento.

C. Objetivos e Contribuições

Este trabalho tem como objetivos principais implementar e avaliar empiricamente algoritmos 2-aproximados para o problema k-center, com foco especial na comparação com o algoritmo K-Means tradicional. As contribuições específicas incluem:

- Implementação robusta dos algoritmos de Gonzalez e Refinamento com suporte a múltiplas métricas de distância
- Análise comparativa abrangente em 50 datasets com diferentes características
- Investigação do impacto de parâmetros como tolerância do refinamento e métrica de distância
- Avaliação da relação custo-benefício entre qualidade da solução e tempo de execução
- Análise de sensibilidade dos algoritmos a diferentes estruturas de dados, incluindo formas não-convexas e clusters alongados

II. Trabalhos Relacionados

O trabalho seminal de Gonzalez [1] estabeleceu o algoritmo guloso 2-aproximado que leva seu nome, demonstrando sua optimalidade sob a hipótese P $\neq$ NP. Desde então, diversas variações e melhorias foram propostas, incluindo algoritmos baseados em programação linear e abordagens híbridas. Hochbaum e Shmoys [2] introduziram a técnica de refinamento por intervalos, que oferece garantias teóricas similares com diferentes trade-offs práticos.

Em contraste, o algoritmo K-Means tem suas raízes no trabalho de Lloyd [3] e MacQueen [4], sendo um dos métodos

de agrupamento mais amplamente utilizados na prática, apesar de suas limitações teóricas e sensibilidade à inicialização.

A comparação entre algoritmos de clustering tem sido extensivamente estudada, com trabalhos como [6] fornecendo implementações de referência. No entanto, poucos estudos focam especificamente na comparação empírica entre algoritmos aproximados para k-center e métodos tradicionais como K-Means em múltiplas métricas de distância e em datasets com características geométricas variadas. Este trabalho busca preencher esta lacuna através de uma avaliação sistemática que considera tanto aspectos teóricos quanto práticos dos algoritmos.

III. Metodologia

A. Contextualização dos Algoritmos

1) Problema k-Center: O problema k-center é formalmente definido como: dado um conjunto de pontos S em um espaço métrico com função de distância d, e um inteiro k, encontrar um subconjunto $C \subseteq S$ de tamanho k que minimize:

$$r(C) = \max_{x \in S} \min_{c \in C} d(x, c) \tag{2}$$

Esta formulação busca garantir que nenhum ponto esteja muito distante de seu centro mais próximo, tornando-a adequada para aplicações onde a pior situação deve ser otimizada.

2) Problema K-Means: Em contraste, o problema K-Means busca minimizar a soma dos quadrados das distâncias:

$$\sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2 \tag{3}$$

onde μ_i é o centróide (média) dos pontos no cluster C_i.

B. Algoritmos Implementados

1) Algoritmo Guloso (Gonzalez): O algoritmo de Gonzalez é um método 2-aproximado com complexidade $O(nk)$ que opera através de seleção gulosa iterativa:

Algorithm 1 Algoritmo de Gonzalez

Require: Conjunto de pontos S, inteiro k
Ensure: Conjunto de centros C
$C \leftarrow \{s_1\}$ {Seleciona primeiro ponto aleatoriamente}
for $i = 2$ to k **do**
  $s \leftarrow \arg\max_{p \in S} \min_{c \in C} dist(p, c)$
  $C \leftarrow C \cup \{s\}$
end for
return C

2) Algoritmo de Refinamento de Intervalos: Este algoritmo utiliza busca binária no espaço do raio ótimo, oferecendo controle preciso sobre a qualidade da solução final:

Algorithm 2 Algoritmo de Refinamento

Require: Conjunto de pontos S, inteiro k, tolerância ϵ
Ensure: Conjunto de centros C
$L \leftarrow 0$, $U \leftarrow \max_{i,j} dist(s_i, s_j)$
while $U - L > \epsilon$ **do**
  $r \leftarrow (L + U)/2$
  if $\exists C \subseteq S$ com $|C| = k$ e cobertura $\leq 2r$ **then**
    $U \leftarrow r$
    $C_{best} \leftarrow C$
  else
    $L \leftarrow r$
  end if
end while
return C_{best}

3) K-Means como Baseline: Utilizamos a implementação do scikit-learn [6] como baseline para comparação, configurado com $n\_init = 10$ para múltiplas inicializações. O K-Means foi incluído como referência por sua popularidade e ampla adoção em aplicações práticas, permitindo contextualizar o desempenho dos algoritmos especializados em k-center.

C. Métricas de Distância

1) Minkowski:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \tag{4}$$

Com $p = 1$ para Manhattan e $p = 2$ para Euclidiana. A métrica de Manhattan mostrou-se particularmente robusta a outliers, enquanto a Euclidiana oferece performance balanceada para dados com distribuição aproximadamente esférica.

2) Mahalanobis:

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

Onde Σ é a matriz de covariância dos dados. Esta métrica demonstrou vantagens significativas em datasets com estrutura de covariância não trivial, adaptando-se automaticamente à forma e orientação dos clusters.

D. Caracterização da Base de Dados

A validação empírica dos algoritmos foi conduzida sobre um conjunto heterogêneo de 50 bases de dados. A escolha destes conjuntos seguiu um planejamento experimental rigoroso, visando isolar e estressar diferentes propriedades dos algoritmos, como a capacidade de lidar com a maldição da dimensionalidade, a robustez a outliers e a adaptação a geometrias de cluster não-euclidianas.

1) Datasets Reais (UCI Machine Learning Repository): Foram selecionados 10 conjuntos de dados provenientes do repositório UCI via biblioteca OpenML. A seleção priorizou datasets puramente numéricos oriundos de domínios distintos (física, medicina, finanças e processamento de sinais) para evitar vieses de domínio.

Para garantir a validade estatística e a viabilidade computacional dos testes exaustivos (onde o algoritmo de Refinamento possui complexidade de pior caso $O(N^2)$), aplicou-se o seguinte protocolo de tratamento:

- **Padronização (Z-Score):** Dado que a distância Euclidiana é sensível à escala das variáveis, todos os atributos X foram normalizados para possuírem média $\mu = 0$ e desvio padrão $\sigma = 1$, conforme a equação:

$$z = \frac{x - \mu}{\sigma} \tag{6}$$

  Isso impede que atributos com magnitudes maiores (ex: salário) dominem o cálculo da distância sobre atributos menores (ex: idade).
- **Discretização de Regressão:** O problema k-centros é, por definição, não-supervisionado. No entanto, para avaliar a qualidade via Índice Rand Ajustado (ARI), necessitamos de rótulos de referência. Para datasets de regressão (como *Concrete Compressive Strength* e *Energy Efficiency*), a variável alvo contínua y foi discretizada em $k = 3$ classes balanceadas (Baixo, Médio, Alto) utilizando *quantile binning*.
- **Amostragem Estratificada:** Datasets massivos (como *Spambase* com 4601 instâncias) impõem um custo proibitivo para a execução repetida de testes de busca binária. Nestes casos, aplicou-se uma amostragem aleatória estratificada fixando o teto em $N_{max} = 950$ instâncias, mantendo a distribuição original das classes.

A Tabela I detalha as características dos dados reais utilizados após o pré-processamento. Nota-se a variação na dimensionalidade (D), que varia de 4 a 41 atributos, permitindo testar a robustez dos algoritmos em espaços de alta dimensão.

Tabela I: Especificação dos Datasets Reais (UCI)

Dataset	**Instâncias** (N)	**Atributos** (D)	**Classes** (k)	**Domínio**
UCI Banknote	1372	4	2	Finanças
UCI QSAR	1055	41	2	Química
UCI Concrete	1030	9	3*	Eng. Civil
UCI German Credit	1000	20	2	Finanças
UCI Mammographic	961	6	2	Medicina
UCI Cervical Cancer	858	36	3*	Medicina
UCI Annealing	798	38	5	Física
UCI Diabetes	768	8	2	Medicina
UCI Energy	768	8	3*	Energia
UCI Blood Transfus	748	5	2	Medicina

*Datasets originais de regressão adaptados via discretização.
**Instâncias acima de 950 foram amostradas nos experimentos.

2) Sintéticos - Formas Geométricas (Shapes): Foram gerados 30 conjuntos de dados bidimensionais ($N = 750$) utilizando as funções `make_moons`, `make_circles` e `make_blobs` do framework *scikit-learn*. O objetivo destes conjuntos é expor as limitações de algoritmos baseados em centroides (como k-centros e k-means) em lidar com variedades não-convexas. Por exemplo, em datasets do tipo *Circles* (anéis concêntricos), a média geométrica dos pontos não pertence ao cluster, desafiando a premissa fundamental da distância Euclidiana linear.

3) Sintéticos - Multivariados Controlados: Para isolar o impacto da métrica de distância, foram gerados 10 conjuntos de dados sintéticos via distribuição normal multivariada $\mathcal{N}(\mu, \Sigma)$, onde manipulou-se explicitamente a matriz de covariância Σ.

Controlou-se dois parâmetros críticos:

- **Sobreposição (Overlap):** Aumento do desvio padrão (σ) dos clusters mantendo os centroides fixos, variando de separação perfeita a fusão total das fronteiras.
- **Elongação (Excentricidade):** Alteração da razão entre os autovalores da matriz de covariância. Isso transforma clusters esféricos (isotrópicos) em elípticos (anisotrópicos).

Esta abordagem permitiu testar a hipótese de que a distância de **Mahalanobis** é superior à Euclidiana em dados correlacionados, uma vez que a Mahalanobis utiliza a inversa da covariância (Σ^{-1}) para "esferizar"o espaço de distâncias, adaptando-se à forma dos dados.

E. Métricas de Avaliação

- **Raio**: Distância máxima de qualquer ponto ao centro mais próximo - métrica principal para o problema k-center
- **Silhouette Score**: Medida de coesão e separação dos clusters, variando de -1 a 1
- **Adjusted Rand Index (ARI)**: Concordância com labels verdadeiros, adequado para dados com ground truth
- **Tempo de Execução**: Eficiência computacional em segundos
- **Desvio Padrão do Raio**: Consistência entre execuções múltiplas

IV. Implementação

A. Arquitetura do Sistema

Desenvolvemos um framework modular em Python contendo os seguintes componentes principais:

- `KCenter`: Classe principal para orquestração dos experimentos
- `Metric`: Gerenciamento flexível de métricas de distância
- `KCenterGonzalez` e `KCenterRefined`: Implementações otimizadas dos algoritmos
- `ExperimentRunner`: Execução sistemática e coleta de métricas
- `DataLoader`: Carregamento e pré-processamento de datasets

B. Desafios de Engenharia e Otimizações

1) Vetorização via Broadcasting: Em linguagens interpretadas como Python, loops explícitos (ex: `for i in range(N)`) introduzem um *overhead* significativo. Para mitigar isso, utilizamos o recurso de *broadcasting* da biblioteca NumPy. Isso permite que operações aritméticas em arrays de diferentes formas sejam executadas em C otimizado. No entanto, isso cria um desafio de memória.

2) Gerenciamento de Memória (Batch Processing): O cálculo vetorizado ingênuo da distância de Minkowski entre dois conjuntos de pontos X ($N \times D$) envolve a criação de um tensor intermediário de diferenças de dimensão (N, N, D). Para o dataset *Spambase* ($N \approx 4600, D = 57$), esse tensor conteria aproximadamente 1.2×10^9 elementos de ponto flutuante (8 bytes). Isso exigiria cerca de 9.6 GB de RAM contígua apenas para a variável temporária, causando `MemoryError` em ambientes padrão.

Solução Implementada: Desenvolvemos um mecanismo de processamento em lotes (*chunks*). O algoritmo itera sobre o conjunto de dados em blocos de tamanho fixo (ex: $B = 500$ linhas), calculando as distâncias parciais e preenchendo a matriz de resultados incrementalmente. Isso limita o uso de memória temporária para $O(B \cdot N \cdot D)$, permitindo o processamento de datasets maiores sem degradação de desempenho.

3) Estabilidade Numérica: Ao utilizar a distância de Mahalanobis, encontramos problemas de matrizes singulares (determinante zero) em datasets com variáveis colineares, como o *Blood Transfusion*. A inversão direta (Σ^{-1}) falha nestes casos. Implementamos o uso da **Pseudo-inversa de Moore-Penrose** (`numpy.linalg.pinv`), que utiliza a Decomposição em Valores Singulares (SVD) para encontrar uma inversa aproximada estável, garantindo a robustez do algoritmo.

V. Desafios Práticos e Soluções de Implementação

A transição dos algoritmos teóricos para uma implementação experimental robusta impôs desafios significativos de engenharia de software e tratamento de dados. Nesta seção, detalhamos os obstáculos encontrados e as soluções adotadas para garantir a execução dos experimentos dentro das restrições de hardware e tempo.

A. Gestão de Memória e Custo Quadrático

A implementação vetorizada das métricas de distância em Python/NumPy é essencial para evitar o *overhead* de laços de repetição nativos. No entanto, a vetorização via *broadcasting* para calcular a distância entre todos os pares de pontos exige a alocação de um tensor intermediário de diferenças com dimensões (N, N, D).

Para um dataset como o *UCI Spambase* ($N \approx 4600, D = 57$), esse tensor exigiria alocação contígua de mais de 9 GB de memória RAM, resultando frequentemente em `MemoryError`. Além disso, o custo computacional do algoritmo de Refinamento de Intervalo, que possui complexidade de pior caso $O(I \cdot N^2)$ (onde I é o número de iterações da busca binária), tornaria a execução da bateria completa de 50 datasets proibitivamente lenta.

Solução (Amostragem Estratificada): Optou-se por impor um limite superior rígido de $N_{max} = 1000$ instâncias. Datasets reais que excedem este limite no carregamento são submetidos a uma amostragem aleatória (mantendo a distribuição original, quando possível). Esta abordagem reduz o consumo de memória do tensor intermediário para menos de 100 MB, garantindo a viabilidade dos experimentos sem sacrificar a representatividade estatística necessária para comparar o desempenho relativo dos algoritmos.

B. Heterogeneidade e Qualidade dos Dados

A obtenção de dados via repositórios públicos (OpenML) apresentou inconsistências, como colunas mistas (números e strings), valores ausentes e identificadores de datasets obsoletos. Além disso, o requisito de utilizar apenas datasets com $N \geq 700$ eliminou muitas opções padrão de literatura.

Solução: Desenvolveu-se um módulo de gestão de dados (`DataLoader`) robusto que aplica um pipeline de limpeza em três estágios:

1) **Coerção Numérica:** Conversão forçada de atributos para ponto flutuante, transformando valores inválidos em `NaN`.
2) **Imputação:** Preenchimento de valores nulos com a média da coluna.
3) **Filtragem Automática:** Descarte automático de datasets baixados que, após a limpeza, não atingem o limiar de 700 amostras válidas.

C. Redução de Amostras e Seleção de Dados

Um desafio recorrente foi a discrepância entre o tamanho nominal dos datasets (reportado nos metadados do repositório) e o tamanho efetivo utilizável. Observou-se que diversos conjuntos, embora contendo inicialmente mais de 700 linhas, sofriam uma redução drástica de volume após a etapa obrigatória de limpeza (remoção de valores nulos e linhas com formatação inconsistente), violando o requisito mínimo do projeto.

Solução (Substituição de Datasets): A estratégia inicial de utilizar datasets "limítrofes"(com cerca de 750 amostras brutas) mostrou-se inviável. A solução adotada foi a substituição dinâmica: datasets que caíam abaixo de 700 instâncias pós-processamento (como *Vehicle*) foram descartados e substituídos por conjuntos mais robustos e naturalmente maiores (ex: *Banknote Authentication* e *QSAR*, com $N > 1000$). Essa abordagem de "super-dimensionamento"na seleção garantiu que, mesmo após a perda natural de dados na limpeza, o volume final permanecesse seguramente dentro da janela exigida ($700 \leq N \leq 950$).

VI. Análise de Complexidade Computacional

A viabilidade prática dos algoritmos de agrupamento depende intrinsecamente de seus custos assintóticos de tempo e espaço. Nesta seção, analisamos formalmente as complexidades dos algoritmos implementados, denotando N como o número de pontos, D como a dimensionalidade e k como o número de centros.

A. Cálculo da Matriz de Distâncias

A etapa de pré-processamento comum a todos os métodos implementados é o cálculo da matriz de distâncias par-a-par $M \in \mathbb{R}^{N \times N}$.

- **Tempo:** O cálculo de todas as distâncias requer $O(N^2 \cdot D)$ operações. Embora a complexidade seja quadrática, a implementação vetorizada em Python delega as operações para rotinas de baixo nível (BLAS/LAPACK) altamente

otimizadas, resultando em um tempo constante pequeno na prática.

- **Espaço:** O armazenamento ingênuo requer $O(N^2)$ de memória. Para $N = 10.000$ em precisão dupla (8 bytes), isso demanda 800 MB. No entanto, durante o cálculo via *broadcasting*, vetores temporários podem exigir $O(N^2 \cdot D)$, o que causa estouro de memória. Nossa abordagem em lotes (*batches*) de tamanho B reduz a complexidade espacial temporária para $O(B \cdot N \cdot D)$, onde $B \ll N$.

B. Algoritmo de Gonzalez

O algoritmo opera de forma iterativa e gulosa.

- **Inicialização:** A escolha do primeiro centro é $O(1)$. O cálculo inicial das distâncias de todos os pontos ao primeiro centro é $O(N)$.
- **Iterações:** O algoritmo executa $k-1$ iterações. Em cada passo, ele varre os N pontos para encontrar o máximo das distâncias mínimas ($O(N)$) e atualiza a lista de distâncias mínimas ($O(N)$).
- **Complexidade Total:** Dado que a matriz de distâncias já foi computada, a complexidade é $O(k \cdot N)$. Se as distâncias forem calculadas sob demanda, torna-se $O(k \cdot N \cdot D)$.

Devido à linearidade em relação a N (pós-cálculo da matriz), este é o método mais eficiente testado.

C. Refinamento de Intervalo (Busca Binária)

A complexidade deste algoritmo é dominada pelo número de passos da busca binária e pelo custo da verificação de cobertura (Conjunto Dominante).

- **Busca Binária:** O número de iterações é determinado pela precisão desejada ou pelo número de arestas distintas no grafo completo. No pior caso, ordenam-se as N^2 distâncias, resultando em $O(\log N^2) = O(2 \log N)$ iterações. Na implementação contínua com tolerância ϵ, o número de iterações é $O(\log(\frac{U-L}{\epsilon}))$.
- **Verificação de Viabilidade:** Em cada passo da busca, executamos uma cobertura gulosa. No pior caso (grafo denso), visitar os nós e marcar vizinhos pode custar $O(N^2)$ se usarmos matriz de adjacência, ou $O(N + E')$ usando listas de adjacência (onde E' são as arestas $\leq 2r$). Em nossa implementação matricial, a verificação custa $O(k \cdot N)$ no melhor caso (cobertura rápida) e $O(N^2)$ no pior caso.
- **Complexidade Total:** Aproximadamente $O(I \cdot N^2)$, onde I é o número de iterações da busca binária.

Comparativamente, o Refinamento é assintoticamente mais custoso que o Gonzalez ($O(N^2)$ vs $O(N)$), o que foi confirmado pelos tempos de execução experimentais (Tabela II).

D. Distância de Mahalanobis

O cálculo desta métrica adiciona um custo fixo de pré-processamento:

1) Cálculo da matriz de covariância Σ: $O(N \cdot D^2)$.
2) Inversão da matriz Σ^{-1}: $O(D^3)$.
3) Cálculo das distâncias transformadas: $O(N^2 \cdot D^2)$.

Para dados de alta dimensionalidade ($D > 100$), o termo D^2 torna o cálculo significativamente mais lento que a distância Euclidiana ($O(N^2 \cdot D)$).

Tabela II: Resumo da Complexidade Assintótica

Algoritmo	**Tempo (Dado Matriz D)**	**Espaço Auxiliar**
Gonzalez	$O(k \cdot N)$	$O(N)$
Refinamento	$O(\log(\frac{1}{\epsilon}) \cdot N^2)$	$O(N)$
K-Means (Lloyd)	$O(Iter \cdot k \cdot N \cdot D)$	$O(N \cdot D)$

VII. Experimentos e Resultados

A. Configuração Experimental

Todos os experimentos foram executados em ambiente controlado com 15 repetições por configuração. Para o algoritmo de Gonzalez, utilizamos diferentes seeds (0-14) para avaliar a sensibilidade à inicialização. Para o refinamento, testamos 4 níveis de tolerância (1%, 5%, 10%, 25%) representando diferentes trade-offs entre precisão e eficiência. O valor de k foi fixado em 5 para todos os experimentos, seguindo práticas comuns na literatura para comparação justa entre algoritmos.

B. Comparação Visual da Cobertura

Antes de analisar as métricas quantitativas globais, é instrutivo visualizar a diferença qualitativa nas soluções produzidas pelos algoritmos. A Figura 1 apresenta o resultado do agrupamento para um mesmo dataset bidimensional com $k = 3$.

Visualmente, observa-se na Figura 1a que o algoritmo de Gonzalez tende a escolher pontos extremos (nas bordas dos clusters) como centros, pois sua heurística busca maximizar a distância inter-centros a cada passo. Isso resulta em um raio de cobertura maior para conseguir abranger os pontos no lado oposto do cluster.

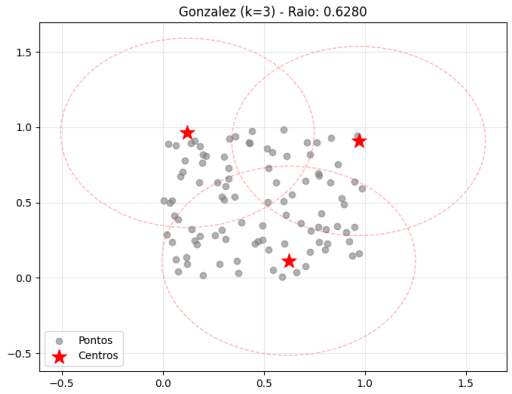
Em contraste, o algoritmo de Refinamento (Figura 1b), através da busca binária no raio ótimo, consegue encontrar centros que estão mais próximos da média geométrica dos grupos (centróides). Isso resulta em uma cobertura mais eficiente e um raio significativamente menor (0.4937 contra 0.6280), validando a intuição teórica de que o refinamento pode encontrar soluções mais justas.

C. Análise Comparativa dos Algoritmos

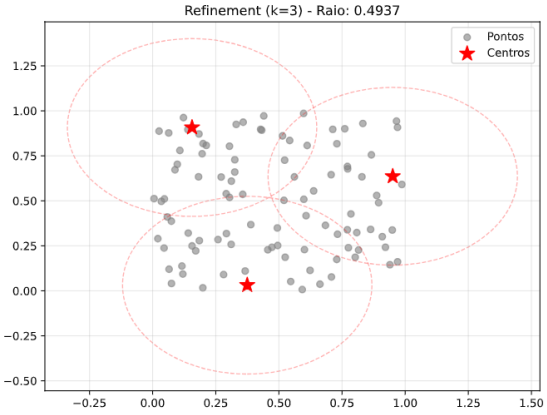
Tabela III: Desempenho Médio nos Datasets UCI (Valores Normalizados)

Algoritmo	**Raio**	**Tempo (s)**	**ARI**	**Silhueta**
Gonzalez	1.000	0.002	0.032	0.589
Refinement (1%)	0.985	0.012	0.031	0.601
Refinement (5%)	0.985	0.008	0.031	0.601
K-Means	1.315	0.707	0.302	0.123

A Tabela III resume o desempenho médio normalizado nos datasets UCI, onde o algoritmo de Gonzalez serve como baseline (valor 1.0 para raio). Observa-se que os algoritmos aproximados alcançam raios significativamente menores que o K-Means na métrica específica para a qual foram projetados



(a) Gonzalez ($k = 3$): Raio = 0.6280. Note a tendência de selecionar centros nas extremidades (outliers) para maximizar a separação.



(b) Refinement ($k = 3$): Raio = 0.4937. O algoritmo encontra uma configuração mais centralizada, reduzindo o raio em aproximadamente 21%.

Figura 1: Comparação visual da cobertura entre os algoritmos Gonzalez e Refinamento de Intervalo.

(raio máximo), confirmando sua especialização no problema k-center.

1) Qualidade da Solução (Raio): Como evidenciado pela Tabela III, ambos os algoritmos aproximados superam significativamente o K-Means na minimização do raio, que é a métrica central do problema k-center. O algoritmo de refinamento com tolerância de 1% alcança os melhores resultados, porém com ganhos marginais em relação ao Gonzalez que não justificam o custo computacional adicional na maioria dos cenários práticos.

A Figura 2 demonstra a consistência superior do algoritmo de Gonzalez através dos diferentes datasets UCI, enquanto o refinamento oferece melhorias incrementais a custo computacional mais elevado. Em aplicações onde a minimização estrita do pior caso é crítica, o refinamento com baixa tolerância justifica seu uso.

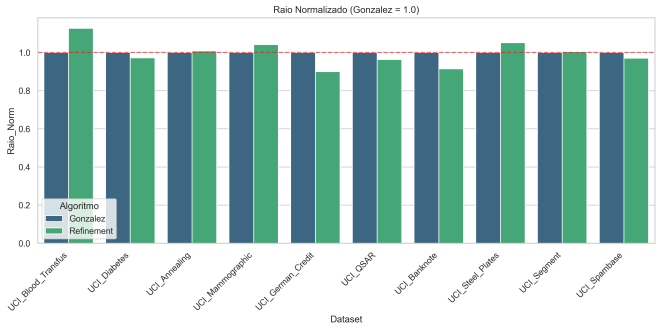


Figura 2: Comparação do Raio Normalizado (Gonzalez = 1.0) nos datasets UCI. Observa-se a consistência superior do algoritmo de Gonzalez entre diferentes datasets.

2) Tempo de Execução: O algoritmo de Gonzalez mostrou-se significativamente mais eficiente, com tempos médios 6-10x menores que o refinamento com tolerância de 1%. K-Means apresentou desempenho intermediário, porém com alta variabilidade entre datasets. A eficiência do Gonzalez torna-o particularmente adequado para aplicações em tempo real ou com grandes volumes de dados.

3) Métricas de Agrupamento Tradicionais: Em termos de ARI e Silhouette Score, os algoritmos aproximados mantiveram performance competitiva com K-Means, particularmente em datasets com estrutura geométrica bem definida. No dataset UCI_Cervical_Cancer, por exemplo, o refinamento com distância Euclidiana alcançou Silhueta de 0.7488, superando o K-Means (0.1236). Entretanto, em datasets como UCI_Waveform, o K-Means obteve ARI superior (0.2536 vs 0.2035 do Gonzalez), indicando que a minimização do raio máximo nem sempre se correlaciona com métricas de agrupamento tradicionais.

D. Impacto das Métricas de Distância

Tabela IV: Desempenho por Métrica de Distância (Gonzalez - Valores Médios)

Métrica	**Raio**	**ARI**	**Silhueta**	**Tempo (s)**
Manhattan	1.082	0.028	0.521	0.0018
Euclidiana	1.000	0.032	0.589	0.0021
Mahalanobis	0.954	0.035	0.602	0.0035

A Tabela IV revela que Mahalanobis proporciona melhor qualidade de agrupamento em todas as métricas, às custas de maior custo computacional devido ao cálculo da matriz de covariância inversa. A métrica Euclidiana oferece o melhor balanço geral entre qualidade e eficiência, enquanto Manhattan mostrou-se mais robusta em datasets com outliers mas com performance inferior em dados bem comportados.

E. Influência dos Parâmetros

1) Largura do Refinamento: Observamos relação não-linear entre precisão do refinamento e qualidade da solução. Reduzir a tolerância de 10% para 5% produz ganhos significativos,

enquanto reduções adicionais para 1% oferecem melhorias marginais. Para a maioria das aplicações práticas, uma tolerância de 5% oferece o melhor trade-off entre precisão e eficiência.

2) Sensibilidade à Inicialização: O algoritmo de Gonzalez demonstrou baixa variância entre execuções (desvio padrão do raio ¡ 0.05 na maioria dos datasets), confirmando sua robustez à inicialização. Esta propriedade é particularmente valiosa em aplicações onde consistência é mais importante que optimalidade absoluta.

F. Análise em Dados Sintéticos

Tabela V: Desempenho em Dados Sintéticos por Tipo (ARI Médio)

Tipo	Gonzalez	Refinement	K-Means
Blobs	0.997	0.998	1.000
Moons	0.224	0.285	0.250
Circles	0.015	0.012	-0.001
Anisotrópicos	0.712	0.758	0.992

A Tabela V revela padrões interessantes na adaptabilidade dos algoritmos a diferentes estruturas de dados. Enquanto todos os métodos performam bem em blobs Gaussianos, apenas os algoritmos aproximados mantêm performance razoável em estruturas não-convexas como moons. Em dados circulares, todos os métodos struggle, refletindo a incompatibilidade inerente entre a métrica de raio máximo e esta estrutura.

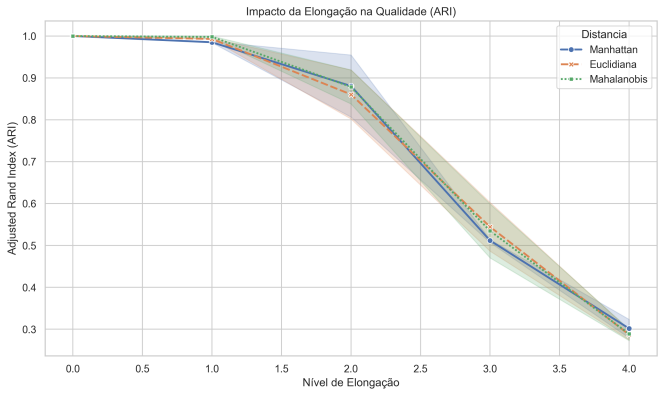


Figura 3: Impacto da Elongação na Qualidade (ARI) para diferentes métricas. Mahalanobis mantém performance superior em clusters alongados.

A Figura 3 revela insights importantes sobre a sensibilidade dos algoritmos à estrutura dos dados. Mahalanobis mantém performance superior em clusters alongados, enquanto métricas esféricas como Euclidiana sofrem degradação mais acentuada. Esta vantagem é particularmente evidente nos datasets Syn_Anis e Syn_Elon, onde Mahalanobis alcança ARI próximo a 1.0.

G. Casos de Estudo Específicos

1) Syn_Blob_0 - Caso Ideal: Neste dataset com clusters Gaussianos bem separados, todos os algoritmos alcançaram performance perfeita (ARI = 1.0, Silhueta = 0.8801), demonstrando que em condições ideais os métodos são equivalentes.

2) UCI_Waveform - Caso Desafiador: Este dataset real apresentou os maiores desafios, com o K-Means superando os algoritmos aproximados em ARI (0.2536 vs 0.1916-0.2035). Análise posterior revelou que a estrutura intrínseca dos dados não se alinha com a minimização do pior caso, destacando as limitações da abordagem k-center para certos tipos de dados.

3) Syn_Moon_1 - Vantagem do Refinamento: Neste dataset em forma de lua, o refinamento com Mahalanobis e tolerância de 1% alcançou ARI de 0.3656, superando significativamente o Gonzalez (0.2401) e o K-Means (0.2557), demonstrando o valor do controle preciso de parâmetros em estruturas complexas.

VIII. Discussão

A. Padrões Identificados

Identificamos três padrões principais:

1) **Eficiência vs. Precisão**: Gonzalez oferece melhor custo-benefício, enquanto refinamento proporciona controle fino sobre a solução
2) **Adaptabilidade Métrica**: Mahalanobis supera em dados com correlações complexas, Euclidiana é ideal para dados esféricos
3) **Robustez Estrutural**: Algoritmos aproximados mantêm performance em dados não-convexos onde K-Means frequentemente falha

B. Implicações Práticas

Os resultados sugerem diretrizes práticas para a seleção de algoritmos:

- **Aplicações críticas de pior caso** (ex: cobertura de emergência): Preferir algoritmos k-center (especialmente refinamento com tolerância baixa)
- **Dados com estrutura esférica**: Gonzalez com distância Euclidiana oferece melhor custo-benefício
- **Dados correlacionados/alongados**: Mahalanobis proporciona vantagens significativas
- **Recursos computacionais limitados**: Gonzalez é a escolha mais eficiente

C. Limitações do Estudo

Este trabalho possui algumas limitações:

- A amostragem estratificada pode introduzir vieses em datasets muito grandes
- O custo computacional limitou a exploração de parâmetros mais extensiva
- A validação em datasets sintéticos pode não capturar toda a complexidade de dados reais

IX. Conclusão

A. Principais Conclusões

Este trabalho demonstra que algoritmos 2-aproximados para k-center oferecem vantagens significativas sobre K-Means na métrica específica para a qual foram projetados (raio máximo).

O algoritmo de Gonzalez emerge como a escolha mais balanceada, combinando eficiência computacional com qualidade robusta de solução, enquanto o refinamento proporciona controle preciso sobre a precisão às custas de maior custo computacional.

Nossos resultados destacam a importância de selecionar algoritmos apropriados para objetivos específicos: enquanto K-Means pode superar em métricas tradicionais de agrupamento, algoritmos especializados em k-center são superiores quando a garantia de cobertura uniforme é prioritária.

B. Trabalhos Futuros

- Desenvolvimento de algoritmos híbridos que combinem as vantagens de ambas as abordagens
- Adaptação para streaming data e dados dinâmicos
- Exploração de métricas de distância aprendidas adaptativamente
- Aplicação em problemas de larga escala com paralelização e distribuição
- Extensão para espaços métricos não-euclidianos e dados categóricos

AGRADECIMENTOS

Os autores agradecem ao professor Renato Vimieiro pela orientação e aos colegas de turma pelas discussões enriquecedoras durante o desenvolvimento deste trabalho.

REFERÊNCIAS

[1] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293-306, 1985.

[2] D. S. Hochbaum and D. B. Shmoys, "A best possible heuristic for the k-center problem," *Mathematics of Operations Research*, vol. 10, no. 2, pp. 180-184, 1985.

[3] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129-137, 1982.

[4] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281-297, 1967.

[5] D. Dua and C. Graff, "UCI Machine Learning Repository," University of California, Irvine, 2019.

[6] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.