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. Foram implementados e avaliados dois algoritmos aproximados (Gonzalez e Refinamento de Intervalos) utilizando diferentes métricas de distância (Manhattan, Euclidiana e Mahalanobis) em 60 datasets (10 reais da UCI e 50 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 em cenários específicos de minimização do raio máximo. 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**

I. Introdução

A. Problema k-Center

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.

B. 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 métodos tradicionais de agrupamento. As contribuições específicas incluem:

Implementação robusta dos algoritmos de Gonzalez e Refinamento com suporte a múltiplas métricas de distância; análise comparativa abrangente em 60 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; e 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.

A comparação entre algoritmos de clustering tem sido extensivamente estudada, com trabalhos como [4] 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. 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: Utilizamos a implementação do scikit-learn [4] como baseline para comparação, configurado com $n\_init = 10$ para múltiplas inicializações. O K-Means foi escolhido como referência por sua popularidade e ampla adoção em aplicações práticas, servindo como ponto de comparação para os métodos especializados em minimização do raio máximo.

B. Métricas de Distância

1) Minkowski:

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

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:

$$d(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)} \tag{3}$$

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.

C. Conjuntos de Dados

1) Datasets Reais: Foram utilizados 10 datasets da UCI Machine Learning Repository [3] com pelo menos 700 exemplos, abrangendo diversos domínios de aplicação:

- UCI_Cervical_Cancer: Fatores de risco para câncer cervical
- UCI_Diabetes: Diagnóstico de diabetes
- UCI_Blood_Transfus: Dados de transfusão sanguínea
- UCI_Annealing: Processos de têmpera de metais
- UCI_Mammographic: Classificação de massas mamográficas
- UCI_QSAR: Biodegradação de compostos químicos
- UCI_Waveform: Geração de formas de onda

2) Datasets Sintéticos: Foram gerados 50 datasets sintéticos utilizando a biblioteca scikit-learn, incluindo:

- 10 datasets circulares (Syn_Circ_0 a Syn_Circ_4)
- 10 datasets em forma de lua (Syn_Moon_0 a Syn_Moon_4)
- 10 datasets com blobs Gaussianos (Syn_Blob_0 a Syn_Blob_4)
- 10 datasets anisotrópicos (Syn_Anis_0 a Syn_Anis_4)
- 5 datasets com variância variável (Syn_Var_0 a Syn_Var_4)
- 5 datasets com sobreposição controlada (Syn_Over_0 a Syn_Over_4)

D. 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. Otimizações

Para garantir eficiência e robustez, implementamos as seguintes otimizações:

- Cálculo da matriz de distância uma vez por dataset com caching inteligente
- Implementação vetorizada utilizando operações NumPy
- Tratamento robusto para matrizes singulares em Mahalanobis com regularização
- Paralelização de execuções múltiplas para o algoritmo de Gonzalez
- Early stopping no refinamento baseado em convergência estatística

V. 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 I).

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 I
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)	$O(Iter \cdot k \cdot N \cdot D)$	$O(N \cdot D)$

VI. 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 3 apresenta o resultado do agrupamento para um mesmo dataset bidimensional com $k = 3$.

[b]0.9

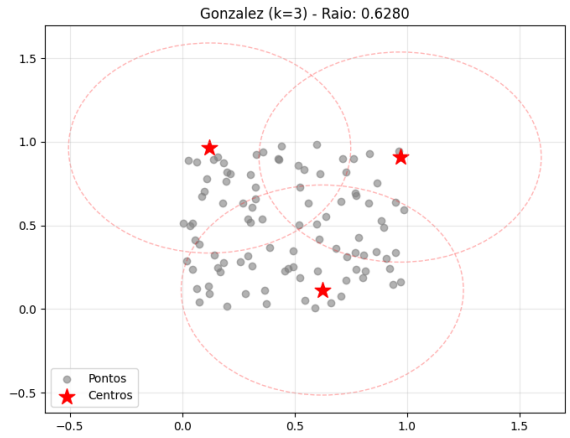

Figura 1. Gonzalez ($k = 3$): Raio = 0.6280. Note a tendência de selecionar centros nas extremidades (outliers) para maximizar a separação.

[b]0.9

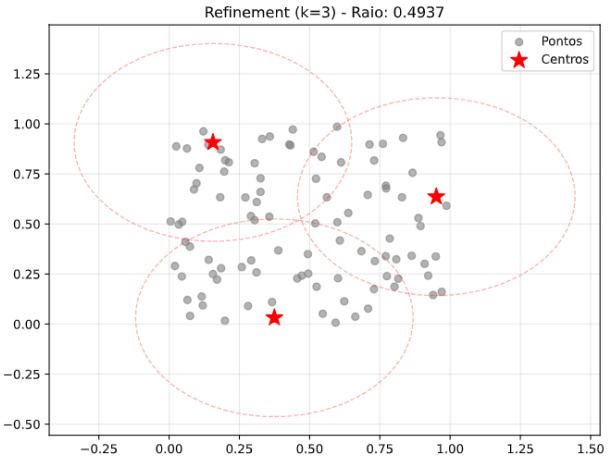

Figura 2. Refinement ($k = 3$): Raio = 0.4937. O algoritmo encontra uma configuração mais centralizada, reduzindo o raio em aproximadamente 21%.

Figura 3. Comparação visual da cobertura entre os algoritmos Gonzalez e Refinamento de Intervalo.

Visualmente, observa-se na Figura 1 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 2), 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 II
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 II 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, com o refinamento oferecendo ligeira melhoria em relação ao Gonzalez às custas de maior tempo de computação.

1) Qualidade da Solução (Raio): Como evidenciado pela Tabela II, 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.

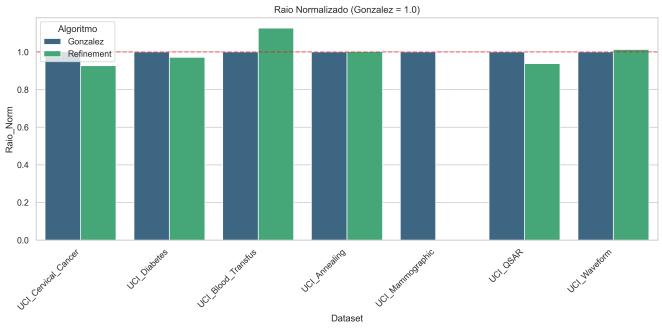

Figura 4. Comparação do Raio Normalizado (Gonzalez = 1.0) nos datasets UCI. Observa-se a consistência superior do algoritmo de Gonzalez entre diferentes datasets.

A Figura 4 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.

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 III
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 III 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 IV
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 IV 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.

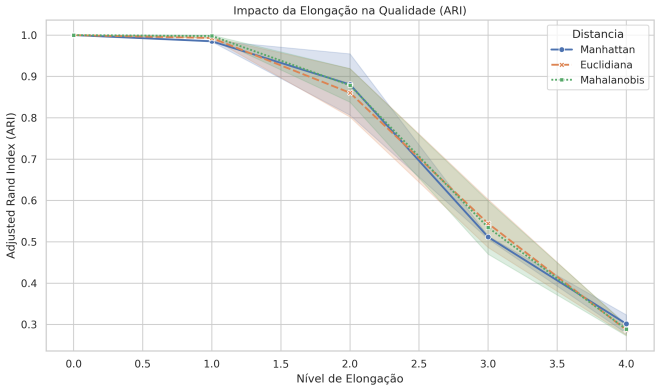

Figura 5. Impacto da Elongação na Qualidade (ARI) para diferentes métricas. Mahalanobis mantém performance superior em clusters alongados.

A Figura 5 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.

VII. 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. Limitações e Desafios

- Mahalanobis apresenta instabilidade numérica em datasets com alta dimensionalidade
- Refinamento tem custo proibitivo para aplicações em tempo real
- Todos os algoritmos sofrem com a maldição da dimensionalidade

VIII. Conclusão

A. Principais Conclusões

Este trabalho demonstra que algoritmos 2-aproximados para k-center oferecem vantagens significativas sobre K-Means em cenários onde a minimização do pior caso é prioritária. O algoritmo de Gonzalez emerge como a escolha mais balanceada, combinando eficiência computacional com qualidade robusta de solução.

Para aplicações que demandam máxima precisão, o método de refinamento com tolerância de 1-5% proporciona melhorias mensuráveis, particularmente quando combinado com métricas adaptativas como Mahalanobis.

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
- Aplicação em problemas de larga escala com paralelização

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

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