Relatório dos Experimentos

José Eduardo D. Massucato[1], **Lucas Almeida Amaral**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Belo Horizonte – MG – Brazil

josedmass@ufmg.br, lucasalmeidama@ufmg.br

Abstract. *This report presents the results obtained from the implementation and execution of approximation algorithms for the k-center (clustering) problem, with an emphasis on a comparative analysis of three widely used approaches: k-means, maximization of the distance between selected points, and optimal radius refinement. The report details the performance of these techniques across a variety of tests, providing a comprehensive assessment of their advantages and limitations.*

Resumo. *Este relatório apresenta os resultados obtidos na implementação e execução de algoritmos aproximativos para o problema dos k-centros (clustering), com ênfase em uma análise comparativa de três abordagens amplamente utilizadas: k-means, maximização da distância entre os pontos escolhidos e refinamento do raio ótimo. O relatório detalha o desempenho dessas técnicas em uma variedade de testes, fornecendo uma avaliação abrangente de suas vantagens e limitações.*

1. Introdução

O problema dos k-centros, também conhecido como problema de agrupamento, é uma tarefa fundamental com amplas aplicações tanto na computação quanto em diversas outras áreas. Neste problema, somos apresentados a um conjunto de pontos dispostos em um espaço (neste relatório, consideraremos que eles estão em um plano) e a um número inteiro k. O objetivo é identificar k centros que otimizem o agrupamento dos pontos, minimizando a maior distância de qualquer ponto ao centro mais próximo. Essa abordagem permite uma representação mais eficiente dos dados.

A importância do problema dos k-centros se estende a inúmeras aplicações práticas, como a otimização de redes de distribuição, onde é crucial posicionar armazéns ou centros de distribuição de forma a minimizar o tempo de entrega ao cliente mais distante. Em análise de dados, o problema é central para tarefas de agrupamento, onde a segmentação eficiente de dados é essencial para o reconhecimento de padrões, compressão de dados, e em diversos algoritmos de aprendizado supervisionado e não supervisionado. Esse problema é também muito recorrente em aprendizado de máquina e em mineração de dados, que são áreas de crescente relevância na computação atualmente.

Descrevendo formalmente o problema, temos um conjunto $S = \{s_1, s_2, ..., s_n\}$ de pontos em um plano e uma função $dist : S \times S \to R^+$ de distância entre os pontos e um inteiro k. A saída esperada é um conjunto $C = \{c_1, c_2, ..., c_k\}$ de pontos que particiona o conjunto de pontos em k grupos, de forma que um ponto s_i pertence ao centro mais

próximo. Diante disso, o objetivo é minimizar o raio máximo dos agrupamentos, isto é, queremos $r(C) = max(dist(s_i, C))$, onde $dist(s_i, C) = min\ d(s_i, c_j)$.

O problema dos k-centros é classificado como NP-difícil, o que torna inviável a busca por uma solução exata em tempo razoável para conjuntos de dados grandes. No entanto, devivo à sua ampla aplicação prática, é essencial encontrar métodos que proporcionem soluções em tempo hábil. Para isso, utilizamos algoritmos aproximativos, que permitem obter soluções que, embora não sejam ótimas, são suficientemente boas para muitas aplicações, além de funcionarem em tempo polinomial.

Durante as próximas seções, descreveremos detalhadamente os algoritmos utilizados, enfatizando que, até o momento, nenhum deles consegue oferecer uma solução melhor que a 2-aproximada, o que implica que a solução obtida pode ser, no máximo, duas vezes pior que a solução ótima. Essa limitação é significativa, pois caso existisse um algoritmo com fator de aproximação melhor que 2, isso implicaria que P = NP, desafiando os fundamentos estabelecidos na teoria da computação atual.

2. Algoritmos e Implementação

Como mencionado anteriormente, a utilização de algoritmos aproximativos é uma prática comum na resolução do problema de agrupamento. Para os experimentos realizados com as bases de dados, que serão posteriormente explicitadas, empregamos três algoritmos distintos: K-Means, Maximização da Distância entre os Pontos Selecionados e Refinamento do Raio Ótimo. Os dois últimos são baseados inteiramente na implementação contida em Kleinberg, J. and Tardos, E. Algorithm design[Kleinberg and Éva Tardos 2005].

2.1. K-Means

Um dos mais famosos algoritmos de clusterização é o K-Means [Wikipedia a], amplamente utilizado em aprendizado de máquina e ciência de dados. Originalmente desenvolvido no campo do processamento de sinais, o k-means tem como objetivo particionar um conjunto de n observações em k grupos ou clusters. Cada um desses clusters é definido por um centro, que representa a média das observações dentro do cluster. O processo de particionamento é iterativo e busca minimizar a soma das distâncias ao quadrado entre cada observação e o centro do cluster ao qual ela pertence.

Formalmente, temos um conjunto de vetores $(x_1, x_2, ..., x_n)$ e queremos particioná-los em $k \leq n$ conjuntos $S = \{S_1, S_2, ..., S_k\}$ definidos por um centro y, de forma que a variância máxima de S se torne a menor possível:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \frac{1}{|S_i|} \sum_{\mathbf{x},\mathbf{y} \in S_i} \|\mathbf{x} - \mathbf{y}\|^2$$

Explicaremos agora o funcionamento da principal implementação existente, o **naive *k*-means**. O algoritmo funciona alternando entre duas etapas:

1. **Etapa de Atribuição:** Cada ponto é atribuído ao cluster cujo centro (média) está mais próximo, usando a menor distância euclidiana ao quadrado.
2. **Etapa de Atualização:** As médias (centroides) são recalculadas para cada cluster com base nos pontos atribuídos a eles.

Essas etapas seguirão de forma iterativa até que as médias recalculadas estejam dentro de um raio de convergência, levando a um particionamento de clusters que contém todos os pontos e possui o menor raio possível.

O K-Means é popular devido à sua simplicidade e eficiência em lidar com grandes volumes de dados, mas existem cenários em que a sua perfomance fica comprometida, como o caso em que os clusters não são esféricos e não possuem tamanho semelhante, além de sua sensibilidade a outliers (pontos extremos) devido ao cálculo da variância.

2.2. Refinamento do Raio Ótimo

Dando prosseguimento à ideia de encontrar um raio de convergência ideal, os algoritmos que serão vistos a seguir buscarão entre diferentes larguras de raio aquela que minimiza a distância dos pontos aos centros pertencentes aos k-clusters.

O primeiro algoritmo a ser implementado servirá como auxiliar para determinar a existência dos k-centros tendo um raio como parâmetro. Como se trata de um algoritmo 2-aproximativo, queremos determinar, a partir de um ponto qualquer, todos os pontos que estão a uma distância de no máximo duas vezes o raio que foi passado na entrada. Esse ponto que acabamos de determinar será um novo centro. Para continuar a busca por novos centros, podemos excluir todos os pontos que estavam contidos no centro anterior e selecionar um novo ponto aleatório, dando continuidade ao processo até que não restem mais pontos a serem explorados. Caso a quantidade de pontos selecionados para serem os centros seja $\leq k$, sabemos que o raio pode ser usado na solução, mas se a quantidade de centros for $> k$, teremos que escolher um raio maior.

O segundo algoritmo a ser implementado é o principal e, nele, começaremos considerando o raio da solução ótima. Sabemos que ele é maior que zero e, no máximo, igual à distância máxima entre quaisquer dois pontos do conjunto. Podemos iniciar o processo dividindo essa distância ao meio e, em seguida, aplicar o algoritmo visto no parágrafo anterior com $r = rmax/2$. Aqui, duas situações podem ocorrer: ou encontramos um conjunto de k centros com raio de cobertura de, no máximo, $2r$, ou concluímos que não existe uma solução com raio de cobertura de, no máximo, r.

No primeiro caso, podemos diminuir nossa estimativa do raio ótimo; no segundo, precisamos aumentá-la. Isso permite realizar uma espécie de busca binária no raio. A partir daí, realizamos uma nova busca com o raio $r = (r0 + r1)/2$; se a solução ótima tiver raio maior que $r0$, ou se encontrarmos uma solução com raio de, no máximo, $2r = (r0 + r1) < 2r1$, nossas estimativas serão refinadas de um lado ou do outro, como em uma busca binária tradicional. O processo pode ser encerrado quando $r0$ e $r1$ estiverem próximos o suficiente (dado um parâmetro de convergência), resultando em uma solução cujo raio será no máximo o dobro do raio ótimo.

2.3. Maximização da Distância Entre os Pontos Selecionados

O algoritmo que será visto a seguir já não depende mais de um parâmetro de raio inicial para minimizar a distância dos pontos ao centro de cada cluster. Basta conhecer a distância entre quaisquer par de pontos e ele retornará um conjunto de clusters que possuirão um raio que será no máximo duas vezes pior que o raio ótimo.

As etapas a serem implementadas podem ser resumidas como se segue:

1. **Seleção Inicial**: Começamos escolhendo um ponto inicial como o primeiro centro, geralmente de forma arbitrária ou baseado em alguma heurística simples.
2. **Iteração**: Em cada iteração subsequente, calculamos a distância de todos os pontos restantes em relação aos centros já escolhidos. Em seguida, identificamos o ponto s que está mais distante de qualquer um dos centros existentes.
3. **Inclusão do Novo Centro**: Se a quantidade de centros selecionados até então for menor ou igual a k(parâmetro externo do algoritmo), então s é adicionado como um novo centro.
4. **Finalização**: Quando a quantidade de centros admite o mesmo valor de k, paramos a execução do programa.

3. Métricas e Métodos

Antes de mostrar os resultados obtidos por cada um dos algoritmos discutidos, precisamos definir as métricas que serão utilizadas para a avaliação dos mesmos:

1. **Raio Ótimo**
   - **Definição**: O raio ótimo do algoritmo é a distância máxima de um ponto para o centro do cluster em que ele está contido, considerando-se todos os clusters da instância.
2. **Tempo**
   - **Definição**: Indica o tempo de execução total do algoritmo em segundos.
3. **Silhueta (Silhouette Score)**
   - **Definição** [Wikipedia d]: Mede a qualidade da clusterização, avaliando quão bem os pontos estão agrupados dentro de seus clusters em comparação com outros clusters. A métrica varia de -1 a 1, onde valores próximos a 1 indicam que os pontos estão bem agrupados, valores próximos a 0 indicam que os pontos estão na borda entre dois clusters, e valores negativos indicam que os pontos podem estar agrupados incorretamente.
   - Formalmente, tomemos um ponto i pertencente a Ci, em que Ci é um cluster pré-definido e $d(i, j)$ é a distância entre os pontos i e j no cluster CI. Temos que:

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i, j)$$

   - Podemos interpretar $a(i)$ como uma medida de quão bem o ponto i está atribuído ao seu cluster (quanto menor o valor, melhor a atribuição).
4. **Índice de Rand Ajustado (Rand Index)**
   - **Definição** [Wikipedia c]: Mede a similaridade entre duas clusterizações, comparando o número de pares de pontos que são corretamente agrupados ou separados em relação às duas soluções. O índice varia de 0 a 1, onde 1 indica que as duas clusterizações são idênticas e 0 indica que são completamente diferentes.

- Formalmente, o Índice de Rand é definido a partir de quatro contagens: a (pares que estão no mesmo grupo em ambas as partições), b (pares que estão em grupos diferentes em ambas as partições), c (pares que estão no mesmo grupo em uma partição, mas em grupos diferentes na outra), e d (pares que estão em grupos diferentes em uma partição, mas no mesmo grupo na outra). O índice é então calculado como $R = \frac{a+b}{a+b+c+d}$, representando a proporção de concordâncias sobre o total de pares.

Por último, vamos listar os métodos utilizados para gerar os resultados que serão comentados na próxima seção.

1. **Parâmetro de Convergência**
   - Para o algoritmo de Refinamento do Raio Ótimo, utilizamos 5 parâmetros de convergência definidos a partir da função exponencial 2.236^i, com $i = 0, 1, 2, 3, 4$.
   - Em cada iteração do algoritmo, verificamos se a diferença entre o raio de maior largura e o raio de menor largura é maior do que 2.236^i. Caso seja menor, a execução é finalizada.
2. **Parâmetro da Função de Distância de Minkowski** [Wikipedia b]
   - A Função de Distância de Minkowski será executada com $p = 1$ e $p = 2$, as quais equivalem, respectivamente, à distância Manhattan e Euclidiana.
   - A distância Manhattan estabele que a distância entre dois pontos é a soma das diferenças absolutas de suas coordenadas. Essa métrica é útil em contextos onde as direções são restritas ou onde a estrutura dos dados possui alinhamentos claros ao longo dos eixos.
   - A distância Euclidiana estabelece que a distância entre dois pontos é a raiz quadrada da soma dos quadrados das diferenças de suas coordenadas, ou seja, a ”linha reta”mais curta entre os pontos. Esta métrica é amplamente utilizada quando se deseja uma medida direta e intuitiva de proximidade.

4. Experimentos

A experimentação foi realizada utilizando diferentes bases de dados e matrizes de distâncias, sendo uma calculada para $p = 1$ (distância de Manhattan) e outra para $p = 2$ (distância euclidiana), ambas obtidas através do algoritmo de Minkowski. Cada base de dados contém pelo menos 700 pontos e, ao todo, foram utilizados 30 conjuntos de dados, cada um testado 30 vezes para cada um dos algoritmos mencionados na Seção 2. A seguir, detalharemos as bases de dados utilizadas e apresentaremos uma análise dos resultados obtidos durante a experimentação.

4.1. Bases de dados

As bases de dados reais utilizadas neste estudo foram obtidas a partir do UCI Machine Learning Repository [Repository 2023], totalizando 10 conjuntos de dados. Além dessas, foram geradas mais 20 bases de dados sintéticas: 10 delas foram criadas utilizando as funções de geração de dados do Scikit-learn [Scikit-learn], enquanto as outras 10 foram geradas a partir de uma distribuição normal multivariada, com os pontos amostrados em torno do número de centros.

4.1.1. UCI Machine Learning Repository

O UCI Machine Learning Repository, criada e mantida pela Universidade da Califórnia, Irvine, é uma das mais amplamente utilizadas bases de dados na comunidade de aprendizado de máquina e ciência de dados. Foram selecionados 10 conjuntos de dados desse repositório para serem utilizados na experimentação:

- Bank Marketing
- Wine Quality
- Online Shoppers Purchasing Intention Dataset
- Pen-Based Recognition of Handwritten Digits
- MAGIC Gamma Telescope
- Rice (Cammeo and Osmancik)
- Statlog (Shuttle)
- Skin Segmentation
- Dry Bean
- Predict Students' Dropout and Academic Success

4.2. Resultados

Ao final da experimentação, foram gerados gráficos que comparam as métricas de desempenho para cada um dos algoritmos avaliados. A seguir, apresentaremos esses gráficos e realizaremos uma análise das comparações. C

4.2.1. Raio Ótimo

- **UCI:** Podemos notar que o K-Means apresentou os piores resultados, ou seja, os maiores raios ótimos, especialmente para $p = 2$. Além disso, notamos que o algoritmo do refinamento tende a piorar sua solução à medida que o parâmetro de convergência aumenta.

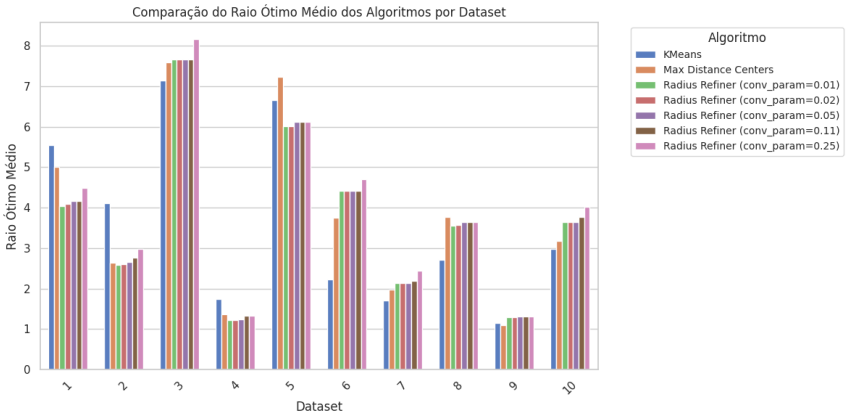

Figura 1. Distância calculada para p=1

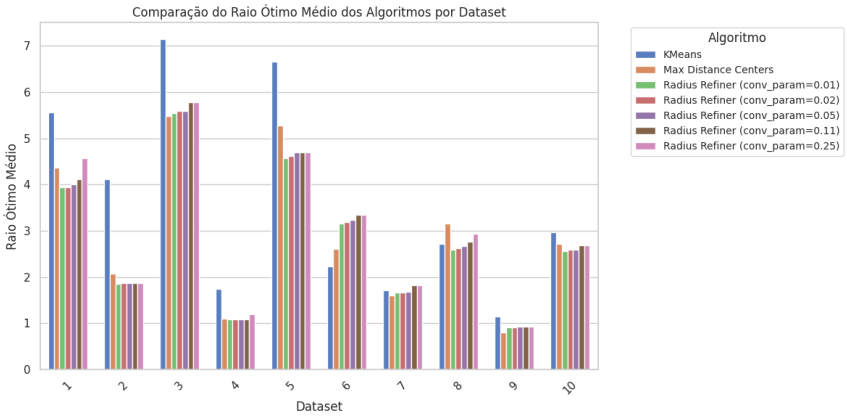

Figura 2. Distância calculada para p=2

- **Scikit-Learn:** O mesmo padrão é observado nos dados do Scikit-Learn. No entanto, de forma geral, o algoritmo que seleciona os pontos maximizando a distância entre eles apresentou os piores raios ótimos, mesmo quando comparado ao algoritmo de refinamento com um parâmetro de convergência de 25%. Essa tendência se acentua ao analisarmos os resultados para $p = 2$, como ilustrado nos gráficos a seguir.

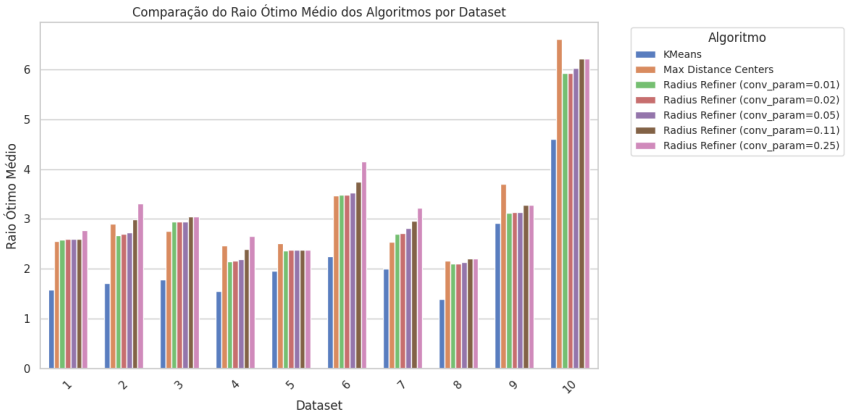

Figura 3. Distância calculada para p=1

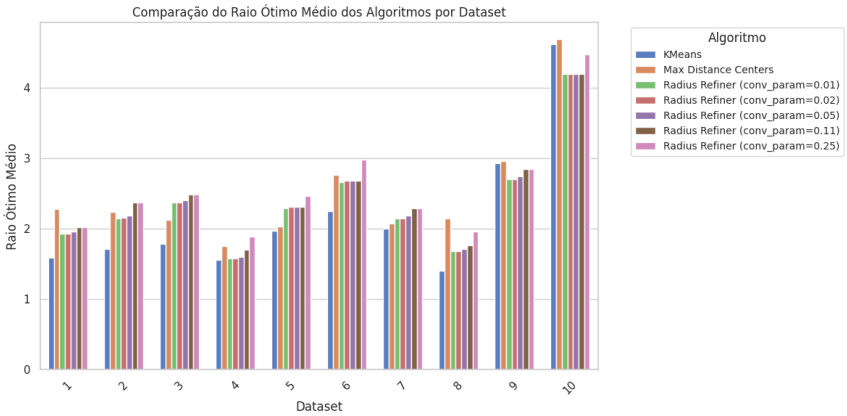

Figura 4. Distância calculada para p=2

- **Normal Multivariada:** Observou-se um comportamento semelhante ao identificado nos dados do Scikit-Learn.

De modo geral, observamos que as soluções para $p = 1$ apresentaram raios ótimos maiores em comparação com $p = 2$, indicando que a solução para $p = 2$ foi superior. No entanto, no caso do K-Means, o resultado permaneceu inalterado.

4.2.2. Silhueta (Silhouette Score)

A silhueta média não apresentou uma diferença significativa ao compararmos os resultados para $p = 1$ e $p = 2$. Por outro lado, observamos que as soluções geradas a partir dos dados do Scikit-Learn retornaram uma silhueta média inferior àquela obtida com os outros conjuntos de dados. Em adição, o K-means, em algumas ocasiões, apresentou uma silhueta média superior à dos demais algoritmos, indicando um desempenho de agrupamento mais eficaz.

- **UCI:** em torno de 0.5 (mesma média que para os dados sintéticos da Normal Multivariada).

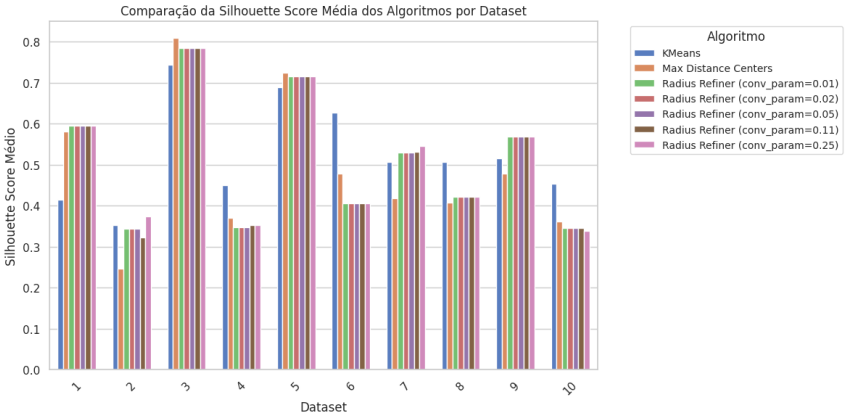

Figura 5. Distância calculada para p=1

- **Scikit-Learn:** em torno de 0.4.

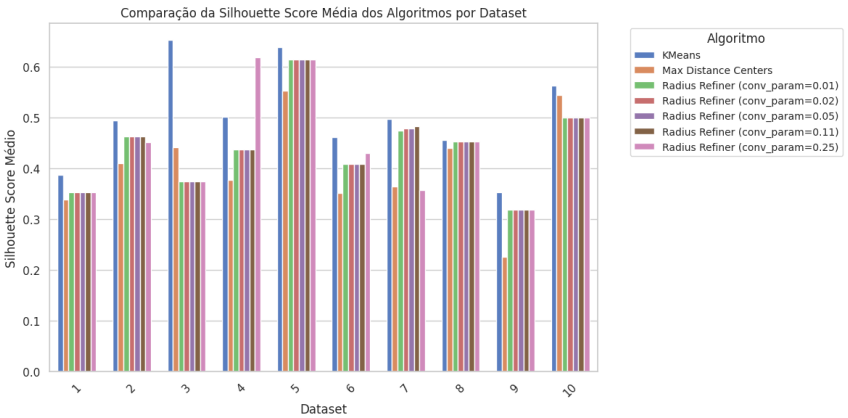

Figura 6. Distância calculada para p=1

4.2.3. Índice de Rand Ajustado (Rand Index)

Nesta análise, também não identificamos uma diferença significativa no Índice de Rand ajustado ao comparar $p = 1$ e $p = 2$. Por outro lado, observamos que o algoritmo K-Means, de maneira geral, apresentou os maiores índices quando comparado aos demais algoritmos. Além disso, nos dados sintéticos gerados pela Normal Multivariada, os Índices de Rand ajustados foram superiores aos dos outros conjuntos de dados.

- **UCI:**

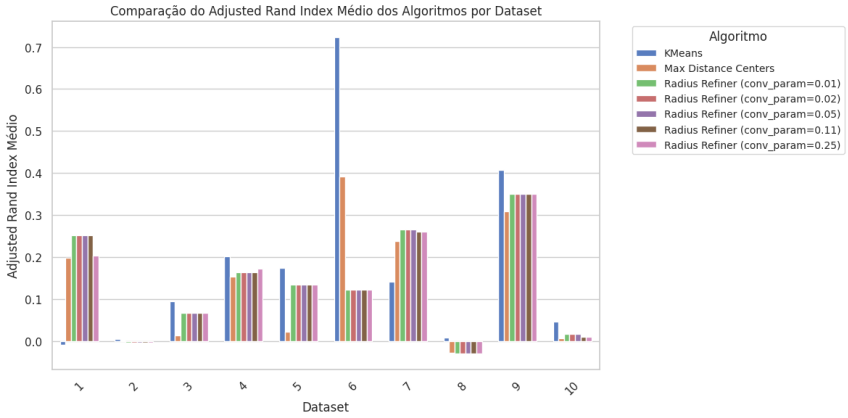

Figura 7. Distância calculada para p=2

- **Normal Multivariada:**

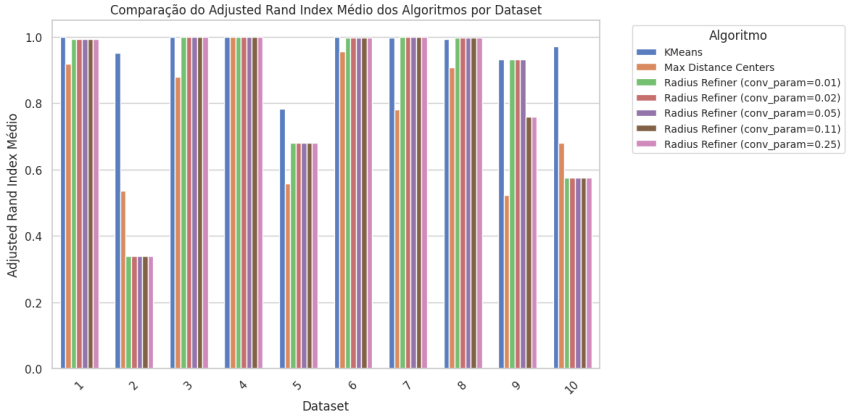

Figura 8. Distância calculada para p=2

4.2.4. Tempo de execução

O valor do parâmetro p não teve uma influência significativa no tempo médio de execução para os dados analisados. Observamos que, para os dados gerados pela Normal Multivariada, o tempo de execução foi maior. No entanto, em todos os conjuntos de dados, a diferença no tempo de execução entre os algoritmos foi bastante evidente.

No gráfico abaixo, apresentamos o tempo de execução para os dados da Normal Multivariada. Neste gráfico, podemos observar que o algoritmo de refinamento do raio ótimo foi o que mais consumiu tempo, embora não tenhamos notado diferenças significativas na variação do parâmetro de convergência em relação a esse resultado. Por outro lado, o algoritmo que busca maximizar a distância entre os pontos escolhidos apresentou o melhor desempenho em termos de tempo de execução. Esse padrão também foi evidenciado nas demais bases de dados.

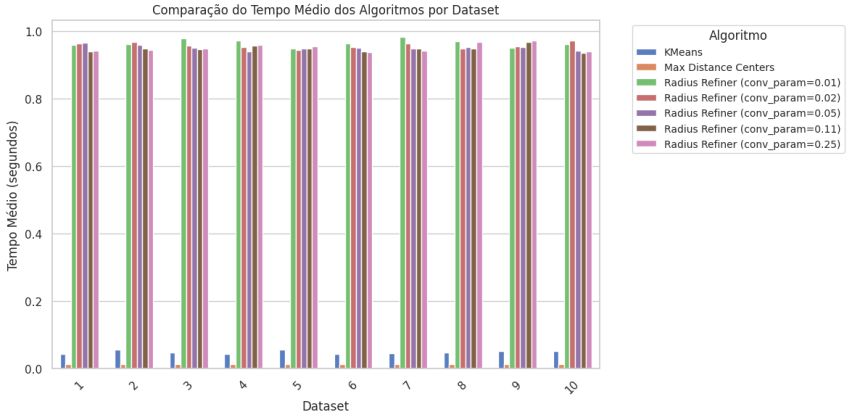

Figura 9. Distância calculada para p=2

Apesar do menor tempo de execução do algoritmo de maximização de distâncias em comparação com o refinador de raios ótimo, podemos perceber que, frequentemente, os resultados do primeiro são piores que o do último, mesmo para parâmetros de convergências maiores, o que reforça o compromisso entre tempo e otimalidade da solução.

5. Conclusão

Através deste trabalho, exploramos diversas abordagens aproximativas para o problema dos k-centros, com foco nos algoritmos K-Means, Maximização da Distância entre os Pontos Selecionados e Refinamento do Raio Ótimo.

Os resultados evidenciam que, embora o algoritmo k-means seja amplamente utilizado e tenha uma implementação relativamente simples, ele pode apresentar limitações significativas em cenários onde os clusters não são esféricos ou possuem tamanhos variados. Por outro lado, os algoritmos de maximização da distância e refinamento do raio ótimo mostraram-se mais suscetíveis a falhas quando consideramos a Distância de Manhattan, além de apresentarem pontuações piores tanto no cálculo da Silhueta quanto no Índice de Rand.

Em conclusão, este trabalho demonstra que não existe uma abordagem única que se sobressaia em todos os cenários, e a escolha do algoritmo deve considerar tanto a natureza dos dados quanto as exigências do problema em questão. A exploração de diferentes técnicas e a combinação de métricas de avaliação são essenciais para o desenvolvimento de soluções de clustering que sejam eficientes e adequadas às necessidades específicas de cada aplicação.

Referências

Kleinberg, J. and Éva Tardos (2005). *Algorithm Design*. Addison-Wesley, 1st edition.

Repository, U. M. L. (2023). Uci machine learning repository. https://archive.ics.uci.edu. Acessado em: 10 de agosto de 2024.

Scikit-learn. Clustering comparison example. https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html. Acessado em: 10 de agosto de 2024.

Wikipedia. k-means clustering. https://en.wikipedia.org/wiki/K-means_clustering. Acessado em: 10 de agosto de 2024.

Wikipedia. Minkowski distance. https://en.wikipedia.org/wiki/Minkowski_distance. Acessado em: 10 de agosto de 2024.

Wikipedia. Rand index. https://en.wikipedia.org/wiki/Rand_index. Acessado em: 10 de agosto de 2024.

Wikipedia. Silhouette (clustering). https://en.wikipedia.org/wiki/Silhouette_(clustering). Acessado em: 10 de agosto de 2024.