Segundo Trabalho Prático de Algoritmos II (2024/1): Soluções para Problemas Difíceis

Julio Assis Souza Amorim (2022043590), Caio Jorge Carvalho Lara (2022043744)

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte – MG – Brasil

caio.lara@ufmg.br, gmndaxdo@ufmg.br

Abstract. *The present paper answers a demand from Prof. Renato Vimieiro in the script for the second practical work on the subject of 'Algoritmos II'. In reference to such work, we will here introduce the problem, describe the methods and metrics used, their implementation and the experiments run with them, present and analyze the results, as well as, finally, arrive at a series of relevant conclusions regarding our research.*

Resumo. *O presente artigo responde a uma requisição do Prof. Renato Vimieiro no roteiro do segundo trabalho prático da matéria de 'Algoritmos II'. Em menção a tal trabalho, iremos aqui introduzir o problema, descrever os métodos e métricas usados, sua implementação e os experimentos com eles realizados, apresentar e analisar os resultados, além de, por fim, chegar a uma série de conclusões relevantes acerca da nossa pesquisa.*

1. Introdução

Com o objetivo de compreender e avaliar os aspectos práticos relacionados à implementação de algoritmos aproximativos, foi dado o seguinte desafio para os alunos da disciplina de Algoritmos II: abordar o problema dos 'K-Centros'; dado um grafo completo com custos nas arestas, respeitando a desigualdade triangular, deve-se encontrar os k pontos, chamados centros, que minimizem a distância máxima de um vértice ao conjunto de centros. Muito embora seja um problema clássico da área da Computação, e somente é possível encontrar a solução exata usando algoritmos assintoticamente exponenciais, existem soluções alternativas capazes de chegar em resultados próximos do ótimo, conhecidos como algoritmos aproximativos.

Assim, para este trabalho, foi proposto aos alunos implementar (na linguagem *Python 3*) dois algoritmos capazes de encontrar soluções aproximativas para o problema dos 'K-Centros', que usam abordagens diferentes para resolver o mesmo problema, e avaliar como cada abordagem performa com diferentes distribuições de dados e valores de parâmetros. Para viés de comparação empírica em termos de demanda computacional e qualidade de solução, instruiu-se também o uso de um algoritmo clássico para a resolução do problema K-Means, como implementado na biblioteca *Scikit-Learn*.

2. Algoritmos

2.1 K-Centros

Os algoritmos principais são dois, apelidados de 'Max-Dist' e 'Intervalo', nomes recebidos em virtude da forma como cada um encontra os centros no grafo. O primeiro, e mais simples

de ser entendido, recebe como parâmetro os pontos, um inteiro positivo k e um valor de ‘p >= 1’ , usado no cálculo da distância de Minkowski, esta sendo a nossa métrica. Depois, ele escolhe um ponto, o define como centro inicial e, a cada iteração, procura o ponto mais distante de todos os centros já escolhidos para ser adicionado ao conjunto de centros, até totalizar k elementos. Posteriormente, dados os centros, calcula-se o raio da solução usando uma função auxiliar.

O segundo algoritmo, ‘Intervalo’, recebe tal nome pois usa uma lógica de intervalo para encontrar um valor para o raio e para os centros, diferentemente do que ocorre no algoritmo anterior, que se limita a encontrar apenas os centros. Neste, usa-se uma função auxiliar que checa se é possível encontrar k centros, dado um valor de raio, retornando o valor booleano ‘verdadeiro’ e os centros caso haja solução para tais valores paramétricos, e ‘falso’ caso contrário. Assim, inicia-se definindo o raio mínimo igual a zero e o raio máximo igual à maior distância entre dois pontos no grafo. Depois, a cada iteração, verifica-se se é possível encontrar k centros com a média dos raios (mínimo e máximo). Se for possível, incrementamos o valor do raio máximo. Se não for, aumentamos o valor do raio mínimo. O algoritmo termina de executar quando o raio mínimo e o máximo possuem o mesmo valor e, para que isso ocorra, foi utilizado um mecanismo sutil de convergência, para evitar que o programa ficasse preso em um loop.

2.1 Distância de Minkowski

Essencial para os algoritmos implementados neste trabalho, foi utilizada uma função para servir de métrica entre os pontos, conhecida como Distância de Minkowski. Resultado da generalização da Distância Euclidiana, a função de cálculo da Distância de Minkowski recebe dois pontos e um valor ‘p’ positivo, calculando a raiz p-ésima da soma das diferenças elevadas à potência de p das coordenadas.

$$d(x,y) \mapsto \|x-y\|_p = \left(\sum_i^n |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Figura 1. Fórmula geral do cálculo da distância de Minkowski

Assim, com o objetivo de analisar o impacto do valor de p no cálculo das distâncias, decidiu-se por executar todas as instâncias de teste dos algoritmos de aproximação dos ‘K-Centros’ com dois valores distintos de p, 1 e 2, correspondendo às distâncias de Manhattan e Euclidiana, respectivamente, e notou-se que resultados diferentes foram obtidos. Esta divergência será melhor explorada mais a frente.

2.3 K-Means

Como já foi explicado, foi instruído o uso de uma implementação pronta do algoritmo K-Means para gerar resultados que sirvam de comparativo empírico, isto é, um parâmetro justo para julgar a qualidade das soluções e custo computacional dos algoritmos de ‘K-Centros’ em uso. A diferença entre ‘K-Means’ e ‘K-Centros’ está na forma como cada

método agrupa os dados: o primeiro busca minimizar a distância média entre os pontos de um *cluster*[1] e o centro (média) desse *cluster*, resultando em grupos mais compactos. Por outro lado, 'K-Centros' foca em minimizar a distância máxima entre qualquer ponto do cluster e o seu centro, o que pode levar a clusters mais espalhados.

3. Conjuntos de Dados (*Datasets*)

Com o objetivo de analisar empiricamente a performance dos algoritmos, optou-se por buscar bases de dados de três diferentes fontes, a saber, (1) exemplos da biblioteca *Scikit-Learn*, (2) exemplos gerados a partir de uma Distribuição Normal Multivariada, e (3) dados extraídos do *UC Irvine Machine Learning Repository*.

3.1 Dados Sintéticos - Scikit-Learn

Utilizando a biblioteca Python Scikit-Learn, foram gerados seis conjuntos de dados, usados para testar empiricamente os algoritmos e analisar como cada um se comportava, dados diferentes tipos de distribuições, agrupamentos e valores de p, seguindo o exemplo fornecido no roteiro.

O primeiro *dataset* é composto por dois círculos concêntricos com ruído, utilizando parâmetros como fator e nível de ruído para influenciar a forma e a distribuição dos pontos. O segundo é composto por dois semicírculos ruidosos em forma de luas. O terceiro é formado por grupos de diferentes formas e densidades, onde distintas variâncias são aplicadas aos clusters, resultando em tamanhos e densidades diferentes para cada grupo. O quarto conjunto de dados é uma versão deformada de *blobs*[2] simples, em que uma transformação linear é aplicada para alongar os *clusters*, criando uma distribuição anisotrópica. O quinto é formado por pontos agrupados em torno de centros definidos aleatoriamente, com uma distribuição gaussiana simples, que resulta em *blobs* bem separados. E, por fim, o sexto conjunto de dados é composto por pontos que originalmente eram gerados aleatoriamente e distribuídos uniformemente, sem padrão ou estrutura definida. Contudo, após a aplicação de uma transformação anisotrópica, os pontos se tornam mais alongados ou elípticos. Cada *dataset* carrega 500 pontos.

3.2 Dados Sintéticos - Distribuição Normal

Como requisitado no roteiro do trabalho, para a segunda leva de dados em análise, geramos dez conjuntos de dados aleatórios definidos sob uma Distribuição Normal Multivariada, constituindo então dez trios de *datasets* com valores de desvio-padrão distintos. Devido ao número elevado de conjuntos, limitamos o número de pontos por conjunto a 300.

3.3 Dados Empíricos - UC Irvine Machine Learning Repository

O repositório de *Machine Learning* da *UC Irvine* (ou *UCI*) contém milhares de sub-repositórios que podem ser usados para diversas tarefas, desde agrupar dados em conjuntos, como é desejado nesse trabalho, até avaliar correlações entre diferentes tipos de dados. Assim, para os dez testes, foram escolhidos nove conjuntos de dados, sendo que um

[1] Cluster é um agrupamento de elementos com características similares.
[2] Blob define uma distribuição de pontos que se aglomeram em círculos (em um plano 2D).

foi repetido, em virtude de sua abundância de especificidades e da falta de bons *datasets* (fácil tratamento, instâncias numéricas) para a tarefa de clusterização que tivessem tamanho de até 10.000 linhas. Nesse sentido, tabelas contendo centenas de milhares de linhas foram evitadas, pois o tempo necessário para baixá-las e usá-las seria excessivo.

Assim, optou-se por escolher os seguintes conjuntos de dados, todos disponíveis para importação usando *Python* e presentes no arquivo de *Notebook*[3] em anexo, na seção 'UCI Datasets':

- *Estimation of Obesity Levels Based On Eating Habits and Physical Condition*
    - *Age x TUE* (Idade e tempo de uso de aparelhos eletrônicos)
    - *Age x NCP* (Idade e número de refeições principais)
- *Facebook Live Sellers in Thailand*
    - *Likes x Comments* (Número de likes e número de comentários)
- *Travel Reviews Ratings*
    - *Burger/Pizza Shops x Gyms* (Avaliação pessoal média de hamburguerias e pizzarias e avaliação pessoal média de academias, ambos na Europa)
- *Travel Reviews*
    - *Art Galleries x Dance Clubs* (Avaliação pessoal média de galerias de arte e avaliação média pessoal de clubes de dança, ambos da no Leste Asiático)
- *Statlog (Image Segmentation)*
    - *Intensity Mean x Hedge Mean* (Intensidade média e cobertura média)
- *Iranian Churn*
    - *Customer Value x Age* (Duração da inscrição e idade)
- *Rice (Cammeo and Osmancik)*
    - *Perimeter x Eccentricity* (Perímetro e excentricidade)
- *Wine Quality*
    - *Fixed Acidity x Residual Sugar* (Acidez fixa e açúcar residual)
- *Drug Consumption (Quantified)*
    - *nscore x ascore* (Neuroticismo e agradabilidade)

[3] Notebook é uma interface interativa que permite combinar código, texto, visualizações e execução de programas em um único documento.

Como é possível notar, há conjuntos de dados relacionados aos mais variados assuntos, cujos pontos possuem múltiplas dimensões. Assim, na seção de testes, optou-se por utilizar apenas duas dimensões de cada um desses pontos, como explicitado na listagem acima, a fim de gerar imagens em um plano 2D. Por fim, o valor de k, referente ao número de centros, foi obtido através do cálculo do número de rótulos de cada *dataset* de acordo com o número de valores distintos de uma dada variável categórica. Por exemplo, no caso do repositório sobre arroz, haviam dois rótulos e, portanto, 'k = 2'. As amostras obtidas de cada conjunto de dados se limitaram a 700 pontos.

4. Experimentação

4.1 Métricas Usadas

Para avaliar a execução dos algoritmos 'Max-Dist', 'Intervalo' e 'K-Means', foram usadas algumas métricas, nem todas usadas em cada um dos conjuntos de dados, pois há casos em que a aplicação dos algoritmos não condiz com o objetivo da métrica. Como regra geral, sempre é calculada a média do raio ótimo, que descreve o valor do raio dos centros, bem como seu desvio-padrão, e também o tempo de execução de cada algoritmo. No caso dos conjuntos de dados sintéticos, não havia um rótulo associado aos dados e, portanto, os coeficientes de silhueta e o Índice de Rand Ajustado somente foram calculados para os *datasets* UCI.

Todas essas informações, bem como os nomes dos algoritmos, valores de p, k e, no caso do algoritmo 'Intervalo', o parâmetro de largura, foram adicionados a uma tabela no Notebook Colab, de forma a centralizar os dados e facilitar a leitura e visualização. Todos os experimentos feitos com os algoritmos implementados foram repetidos com 'p = 1' e 'p = 2', e seus valores foram salvos na tabela. No caso do algoritmo 'K-Means', cuja implementação estava contida na biblioteca Scikit-Learn, os testes foram feitos apenas uma vez (no que tange à repetição relativa à variação de p), pois ele não recebe um parâmetro para calcular a distância.

Ademais, os algoritmos por nós implementados usam de uma regra fixa para fazer a escolha arbitrária do centro inicial, ou seja, a execução de ambas as aproximações sempre gera o mesmo resultado, descartando qualquer necessidade de repetir a execução do algoritmo por trinta vezes, como recomendado no roteiro, a fim de obter médias e um valor de desvio-padrão: não há desvio e a média já é o valor informado execução única. O 'K-Means', no entanto, retorna respostas variadas a cada execução, possibilitando então o registro de médias e desvios-padrão por repetição de sua execução.

4.2 Experimentos

Na primeira leva de experimentos, referente aos dados de exemplo da biblioteca *Scikit-Learn*, realizamos trinta execuções únicas, cinco para cada um dos seis conjuntos de dados gerados. Destas cinco, uma faz uso do método 'K-Means', da biblioteca antes citada, enquanto a outras quatro fazem uso dos métodos 'Max-Dist' e 'Intervalo', variando entre 'p = 1' e 'p = 2'.

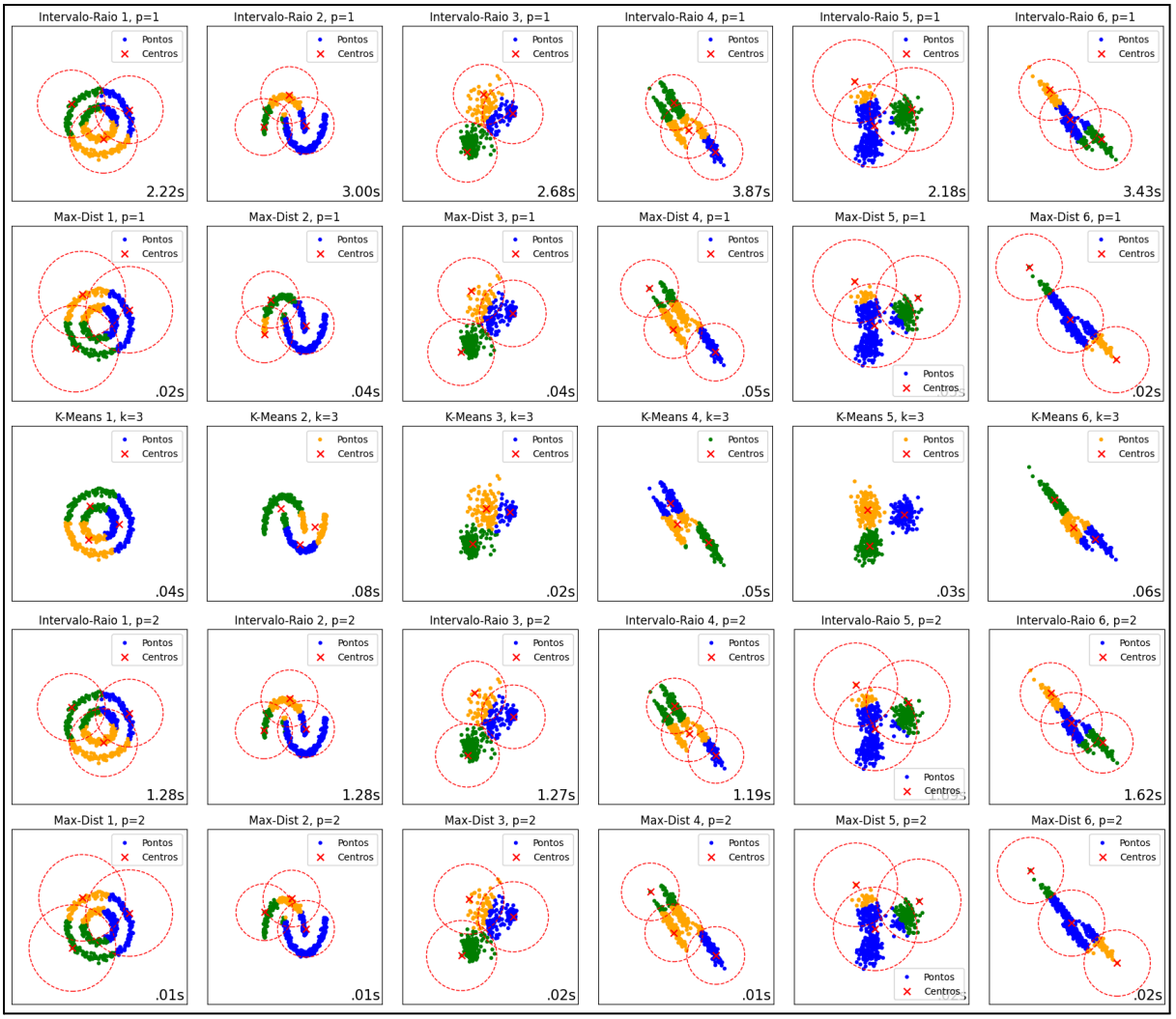

Figura 2. Montagem com os trinta plots únicos resultantes do primeiro experimento. Note que os plots do 'K-Means' não se preocupam em mostrar o raio, já que o algoritmo se preocupa apenas em formar os clusters mais amorfos indicados pela coloração. A coloração é, porém, amostrada nos algoritmos de aproximação para permitir uma certa comparação.

No segundo experimento, nos voltamos a executar os três algoritmos (incluindo as duas versões de cada uma das duas aproximações dos 'K-Centros', variando no valor de 'p') sobre os trinta conjuntos dados sintéticos gerados a partir de Distribuição Normal Multivariada. Desse modo, chegamos a 150 execuções únicas, cinco para cada um dos trinta conjuntos.

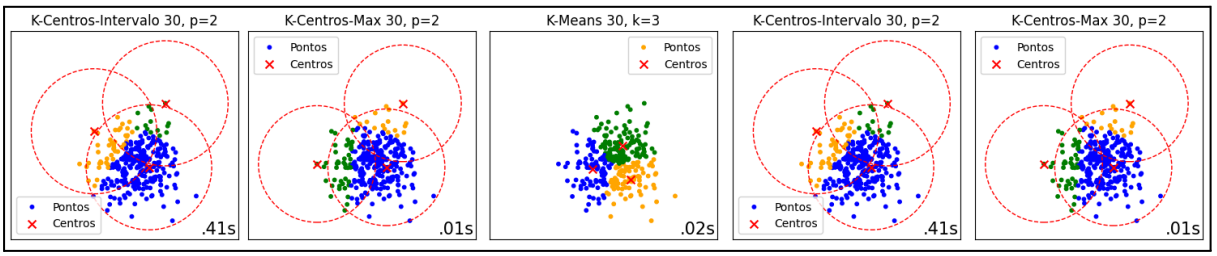

Figura 3. Montagem com cinco plots únicos resultantes do segundo experimento, referentes ao último conjunto de dados sintéticos gerados.

Por fim, na terceira leva de experimentos, aplicamos nossos três métodos sob cinco formas, como nos experimentos anteriores, mas agora sobre os dados empíricos reais disponibilizados publicamente pela *UC Irvine* em seu repositório dedicado a *Machine Learning*. São apenas dez *datasets*, gerando então um total de 50 execuções únicas.

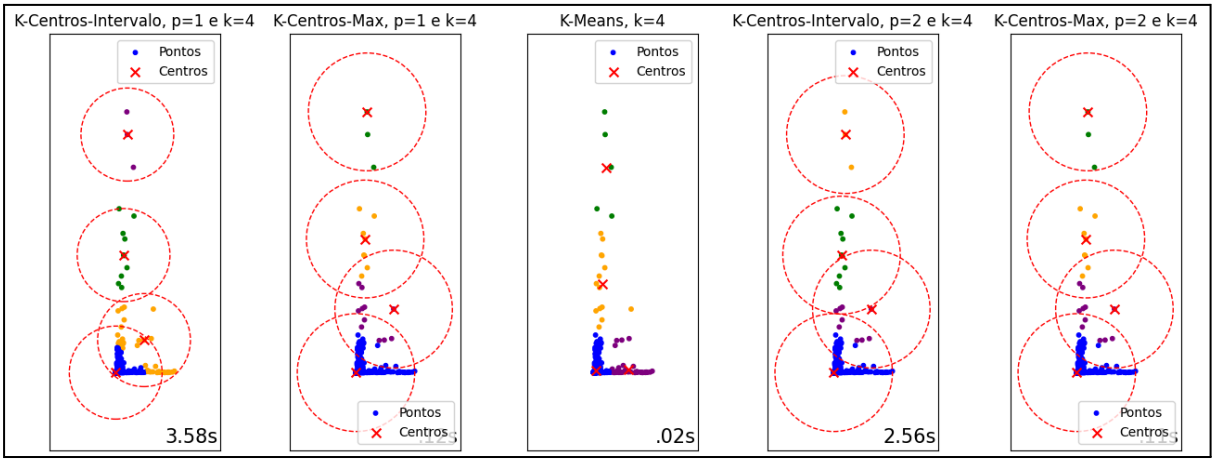

Figura 4. Montagem com cinco plots únicos resultantes do terceiro e último experimento, referentes às colunas que quantificam número de likes (eixo x) e comentários (eixo y) no conjunto de dados empíricos '*Facebook Live Sellers in Thailand*'.

Todos os dados referentes a cada execução única foram registrados como linhas em uma tabela contendo as seguintes colunas: (1) *Descrição*, (2) *Método*, (3) *Raio*, (4) *Rand*, (5) *Silhueta*, (6) *Tempo*, (7) *K*, (8) *Largura*, (9) *Desvio*. (1) é uma *string* descrevendo o conjunto de dados em questão; (2) é uma string descrevendo o método (variando em termos de 'p') usado na execução; (3) é o raio máximo médio da totalidade dos *clusters* encontrados como solução, e inclui também um valor referente a 'K-Means' nesse mesmo sentido; (4) e (5) são os índices de Rand ajustado e silhueta referentes à execução do 'K-Means' nos dados empíricos (igualados a '-2' nas demais execuções); (6) é o tempo de execução do método; (7) é o número de centros encontrados; (8) é o parâmetro de largura do algoritmo (não foi coletado); (9) é o desvio-padrão das múltiplas execuções de cada execução única; como já explicado, apenas o método 'K-Means' apresenta qualquer variação, e por isso só ele por vezes se desvia de 0 em (9).

	Descrição	Método	Raio	Rand	Silhueta	Tempo	K	Largura	Desvio
0	Dado Scikit 1	K-Centros-Intervalo P = 1	2.442131	-2.000000	-2.000000	2.217053	3.0	0.0	0.000000
1	Dado Scikit 1	K-Centros-Max P = 1	2.442131	-2.000000	-2.000000	0.020552	3.0	0.0	0.000000
2	Dado Scikit 1	K-Means	2.442131	-2.000000	-2.000000	0.040773	3.0	0.0	0.000000
3	Dado Scikit 2	K-Centros-Intervalo P = 1	2.442131	-2.000000	-2.000000	2.998458	3.0	0.0	0.000000
4	Dado Scikit 2	K-Centros-Max P = 1	2.442131	-2.000000	-2.000000	0.039186	3.0	0.0	0.000000
...	...	...	...	...	...	...	...	...	...
225	Dados Empíricos 10	K-Means	2.297222	-0.002323	0.331252	0.040063	7.0	0.0	0.135426
226	Dados Empíricos 10	K-Centros-Intervalo P = 1	2.196406	-2.000000	-2.000000	2.351935	7.0	0.0	0.000000
227	Dados Empíricos 10	K-Centros-Max P = 1	2.001501	-2.000000	-2.000000	0.213956	7.0	0.0	0.000000
228	Dados Empíricos 10	K-Centros-Intervalo P = 2	1.898814	-2.000000	-2.000000	3.079847	7.0	0.0	0.000000
229	Dados Empíricos 10	K-Centros-Max P = 2	2.040217	-2.000000	-2.000000	0.307442	7.0	0.0	0.000000

230 rows × 9 columns

Figura 5. Uma amostragem simples da tabela de resultados no *Notebook* em anexo.

4.3 Parâmetro de Largura

Com o intuito de analisar como a mudança no parâmetro de largura afeta a performance do algoritmo ‘Intervalo’, fizemos alguns testes fixando o conjunto de pontos e variando apenas o parâmetro ‘z’, que define o intervalo de largura entre o raio mínimo e o máximo, usados para se chegar ao raio ótimo.

Nesse sentido, algumas hipóteses haviam sido formuladas sobre como o comportamento do algoritmo seria modificado a partir dessa mudança. Inicialmente, esperava-se que o raio ótimo encontrado fosse melhorar quanto maior fosse o valor do parâmetro de largura, pois o algoritmo poderia chegar a um valor mais exato, e também que o tempo de execução fosse aumentar ou decrescer linearmente de acordo com a variação desse parâmetro.

Para fins de demonstração, e também para manter a didática do artigo, preferiu-se mostrar a execução do algoritmo variando o parâmetro ‘z’ de 0.01 a 0.25, de 0.05 em 0.05, sobre três distribuições de pontos variadas e complexas, para evitar que um certo tipo de arranjo de pontos afetasse os resultados finais e a conclusão.

Abaixo, podem ser vistos os resultados finais da execução, com os respectivos raios e tempos de execução. Muito embora a hipótese inicial formulada nesse trabalho fosse de que houvesse alguma variação nos valores de raio ótimo encontrados, ao analisar a tabela de dados em que foram compiladas as informações das execuções, viu-se que o valor final coincidiu perfeitamente; ou seja, o valor final foi o mesmo para todos os valores de ‘z’.

Além disso, foram detectadas algumas anomalias no tempo de execução, o que foi uma surpresa curiosa. Contraintuitivamente, os tempos de execução não foram crescentes ou decrescentes de forma linear; ora cresciam, ora decresciam.

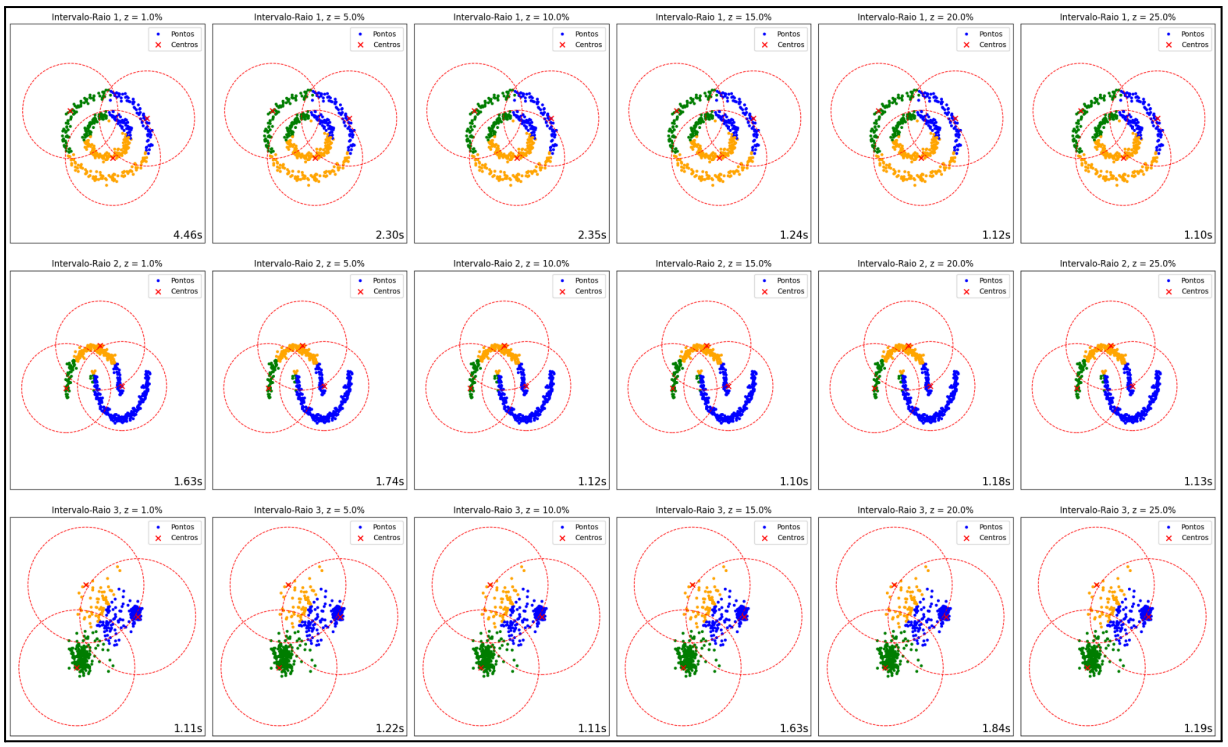

Figura 5. Plot da execução do algoritmo 'Intervalo' com variação do parâmetro 'z'.

Por fim, viu-se que, em geral, valores maiores de 'z' resultaram em execuções mais rápidas e, como o raio ótimo se manteve estável, concluiu-se que é preferível usar valores de 'z' mais próximos de 0.25 do que de 0.01.

5. Análise dos Resultados

Buscando automatizar o próprio método de análise, usufruímos de métodos da biblioteca '*Scipy*' para plotar diferentes gráficos bidimensionais referentes às múltiplas colunas da tabela e obter os coeficientes de correlação correspondentes. Para tal, realizamos um processo de *Label Encoding*[4] na coluna de dados categóricos '*Método*' e montamos um loop inteligente a fim de plotar apenas pares de dados únicos e com coeficiente de correlação real, excluindo das comparações de índice de Rand, silhueta e desvio-padrão os métodos por nós implementados.

Primeiro produzimos plots sem discriminar entre métodos, mas em seguida separamos cada um no sentido de não perder a vista de correlações internas a cada método. Nesse sentido, chegamos a um total de 39 gráficos, todos os quais estão disponíveis no arquivo de *Notebook* em anexo. Muitas das correlações encontradas, entretanto, não são relevantes. Dentre as relevantes, destacam-se apenas as correlações (1) entre método e tempo, (2) entre silhueta e Rand, (3) entre silhueta e raio, (4) entre silhueta e número de centros, (5) entre silhueta e desvio, (6) entre tempo e Rand, (7) entre desvio e raio, e por fim, (8) número de centros e tempo.

Embora o coeficiente obtido não tenha sido significativo, (1) é uma das correlações mais claras em qualquer observação superficial dos experimentos. O tempo de execução de

[4] Espécie de One-Hot-Encoding, em que transforma-se colunas categóricas em numéricas, de acordo com o número de rótulos.

‘K-Means’ é razoavelmente inferior ao do algoritmo ‘Max-Dist’ em todas as instâncias testadas, mas ambos são polinomialmente inferiores em tempo ao ‘Intervalo’. De fato, todas as instâncias deste último levam mais tempo para executar do que qualquer instância individual dos outros dois.

As correlações apresentadas em (7) e (8) talvez sejam as mais intuitivas: o raio tende a aumentar junto do desvio-padrão, afinal, este quantifica a variação da dispersão de dados em relação ao raio médio, sendo ela portanto maior na medida em que esse mesmo raio é maior, tratando de distâncias maiores entre os pontos; já o tempo tende a crescer com o número de centros, afinal, este último é um parâmetro de todos os métodos usados (é o ‘k’, introduzido no nome de ambos os problemas explorados) e influencia a complexidade de todos os algoritmos, embora em graus distintos (‘K-Means’ é o menos afetado).

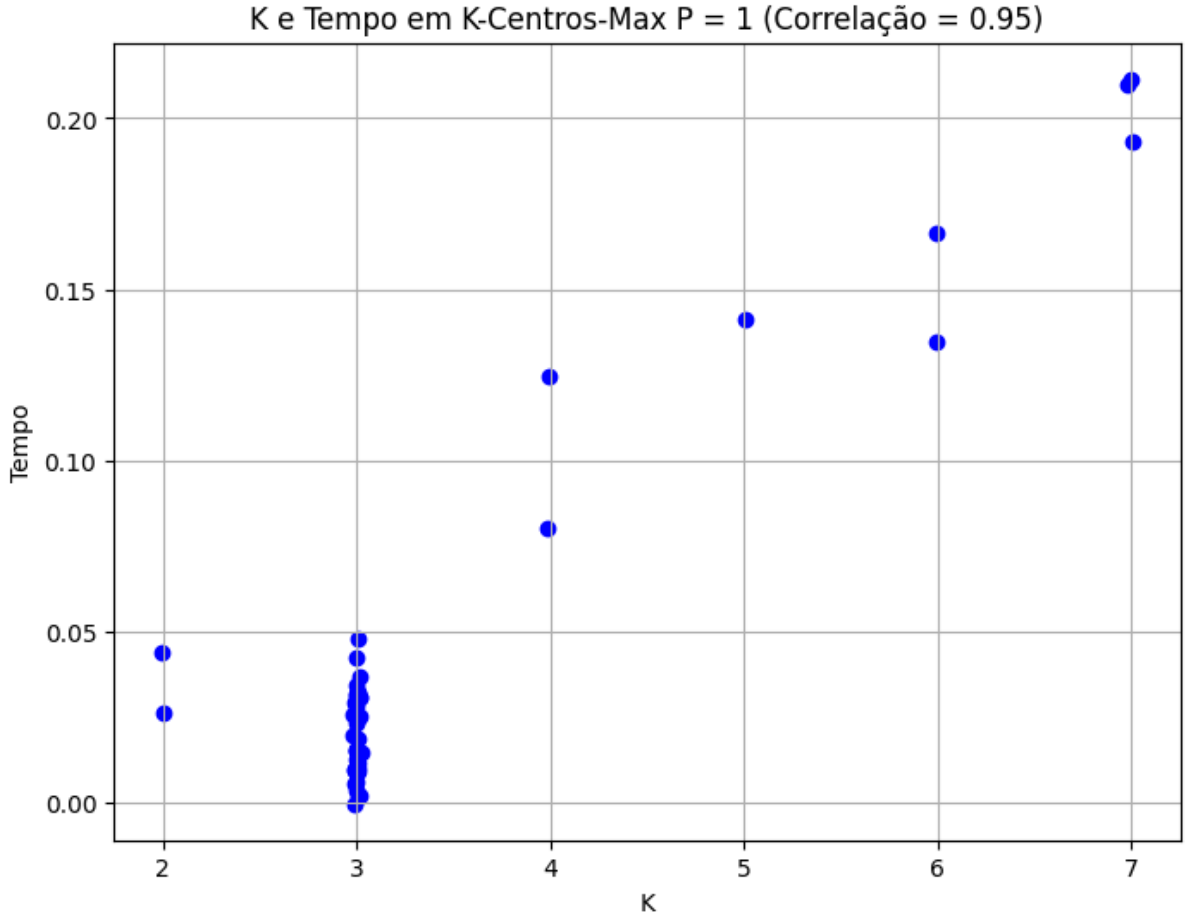

Figura 6. Plot dos valores de ‘k’ (número de centros) e tempo de cada execução pelo método ‘Max-Dist’ para ‘p = 1’. Note a correlação quase perfeita (0.95), visível na própria disposição dos dados na tela.

O índice de silhueta e o índice de Rand, porém, não apresentam uma correlação direta. A silhueta avalia a coesão e a separação interna dos clusters, variando de -1 a +1, onde valores mais altos indicam melhor agrupamento interno. Já o índice de Rand mede a similaridade entre a solução de agrupamento e uma referência externa, uma *label*[5] do próprio *dataset*, variando de 0 a 1, com 1 indicando concordância perfeita. A pequena correlação que

[5] Rótulo conferido a um dado ou conjunto de dados.

detectamos, nesse sentido, nasce do fato de que as *labels* usadas para o cálculo do Rand, que são dados quantificados na própria seleção empírica do repositório de *Machine Learning*, estão naturalmente ligadas aos respectivos valores das colunas plotadas. Do mesmo modo, as demais correlações, todas as quais incluem uma destas duas métricas, também não parecem refletir uma correlação direta, ou ao menos refletem uma correlação muito fraca.

6. Conclusões

Concluímos com este trabalho que, dentre os métodos de aproximação do problema 'K-Centros', aquele que usufrui da maximização da distância entre os centros é o mais eficiente, tendo o mesmo fator de aproximação, fato estabelecido durante as aulas da disciplina, e apresentando tempos de execução muito inferiores aos da alternativa em nossos variados experimentos. Por outro lado, caso seja possível trocar a abordagem dos 'K-Centros' por uma mais alinhada ao problema dos 'K-Means', é preferível fazê-lo, posto a superioridade técnica do algoritmo em relação aos demais.

7. Referências

Vimieiro, R. (2024) "Aula 14 – Soluções aproximadas para problemas difíceis (Parte 2)", apresentação de slides utilizada na disciplina.

Vimieiro, R. (2024) "Trabalho Prático 2 – Soluções para problemas difíceis", roteiro do presente trabalho.

SCIKIT-LEARN. Plot cluster comparison. Disponível em: https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html#sphx-glr-auto-examples-cluster-plot-cluster-comparison-py. Acesso em: 11 ago. 2024.

SCIKIT-LEARN. K-means clustering. Disponível em: https://scikit-learn.org/stable/modules/clustering.html#k-means. Acesso em: 11 ago. 2024.

WIKIPEDIA. Minkowski distance. Disponível em: https://en.wikipedia.org/wiki/Minkowski_distance. Acesso em: 11 ago. 2024.

UCI MACHINE LEARNING REPOSITORY. Disponível em: https://archive.ics.uci.edu/ml/index.php. Acesso em: 11 ago. 2024.

SBC. Templates para artigos e capítulos de livros. Disponível em: https://www.sbc.org.br/documentos-da-sbc/category/169-templates-para-artigos-e-capitulos-de-livros. Acesso em: 11 ago. 2024.

SCIKIT-LEARN. K-means clustering. Disponível em: https://scikit-learn.org/stable/modules/clustering.html#k-means. Acesso em: 11 ago. 2024.

SCIKIT-LEARN. Plot cluster comparison. Disponível em: https://scikitlearn.org/stable/auto_examples/cluster/plot_cluster_comparison.html#sphx-glr-autoexamples-cluster-plot-cluster-comparison-py. Acesso em: 11 ago. 2024.