Trabalho Prático 2 - Algoritmos 2 - Soluções para o problema de k-centros

Vinicius L. C. Faria[1], **Luiz F. G. Rocha**[2]

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

Resumo. *Este trabalho apresenta o problema dos k-centros, um algoritmo de* clustering *NP-Difícil, estabelecendo dois algoritmos 2-aproximativos bem estudados na literatura para o problema. Com isso, implementou-se esses algoritmos na linguagem Python e, utilizando também a implementação do algoritmo de Lloyd para o problema das k-médias, buscou-se comparar diferentes métricas em relação à qualidade do raio máximo bruto, classificação e tempo de execução para as soluções encontradas entre esses algoritmos levando em conta diferentes instâncias sintéticas e reais.*

1. Introdução

O problema dos k-centros se trata de um problema clássico de agrupamento de dados, sendo utilizado em diversas áreas como aprendizado de máquina e mineração de dados e possuindo diversas variantes bem similares. Informalmente, temos um conjunto de pontos, uma função de distância entre eles, e um parâmetro inteiro k, e assim devemos estabelecer k centros de forma que minimize a maior distância de qualquer ponto ao seu centro mais próximo, ou seja, minimizar o maior raio entre os centros.

O problema retratado, assim como suas variantes, se encontra como NP-difícil, e logo não possui algoritmo polinomial conhecido para resolvê-lo. Apesar disso, há diversos algoritmos aproximativos estudados na literatura que conseguem estabelecer soluções até, no máximo, 2 vezes pior que a solução ótima, ou seja, com o raio no máximo 2 vezes maior comparado ao raio ótimo.

Dado o exposto, o objetivo desse trabalho se trata de implementar alguns algoritmos 2-aproximativos do problema e utilizá-lo, juntamente com outro algoritmo utilizado pelo método de k-médias, em diversos cenários reais e sintéticos diferentes com o objetivo de compará-los em questão de qualidade da solução, tempo de execução e outras métricas estabelecidas, com o objetivo de estudar melhor o comportamento de cada algoritmo.

2. Formulação do problema e algoritmos

Formalmente, no problema dos k-centros, é fornecido um conjunto de pontos $S = \{s_1, s_2, ..., s_n\}$, uma função de distância $dist : S \times S \rightarrow R^+$ e um número inteiro K. A saída consiste em um conjunto de pontos $C = \{c_1, c_2, ..., c_k\}$, considerados centros, particionando o conjunto S em k grupos (clusters). Um ponto s_i pertence ao grupo de seu centro c_j mais próximo, logo o raio de cada cluster é considerado a distância entre c_j e seu ponto pertencente mais distante $dist(s_i, C) = min\ dist(s_i, c_j)$. Com isso, o problema consiste em minimizar o raio máximo entre todos os clusters, $r(C) = max\ dist(s_i, C)$.

Como o problema dos k-centros é NP-Difícil, serão utilizados algoritmos 2-aproximativos para resolver o problema em tempo hábil, e então comparados em diversas métricas. Para a resolução direta do k-centros, será utilizado o algoritmo heurístico

e o algoritmo de refinamento denotados pelos algoritmos 1 e 2, respectivamente, ambos trazendo uma perspectiva diferente para a resolução do problema. Para motivos de comparação, também será utilizado o algoritmo de Lloyd [Kanungo et al. 2002] para a resolução pelo método de k-médias.

No caso do algoritmo de refinamento, é encontrado inicialmente o raio máximo $r_{max} = max\ dist(s_i, s_j)$ e definido uma largura l. Dessa forma, iniciando no intervalo $[0, r_{max}]$, executamos uma busca binária no intervalo, optando recursivamente pela parte superior do intervalo caso uma solução for encontrada utilizando o algoritmo 2, e pela parte inferior caso contrário. Essa busca continua até que a largura do intervalo seja $l\%$ do intervalo original, em que a resposta final do algoritmo é obtida.

Algorithm 1 Algoritmo guloso

Input: S, dist

if $|S| \leq k$ **then**
    return S
end if
$s \leftarrow$ valor arbitrário de S
$C \leftarrow \{s\}$
while $|C| < k$ **do**
    $c \leftarrow s$ que maximize $dist(s, C)$
    $C \leftarrow C \cup \{c\}$
end while
return C

Algorithm 2 Algoritmo de refinamento

Input: S, dist, r

$S' \leftarrow S$
$C \leftarrow \emptyset$
while $S' \neq \emptyset$ **do**
    $s \leftarrow$ ponto arbitrário de S'
    $C \leftarrow C \cup \{s\}$
    Remova de S' todos os pontos que estiverem a uma distância 2r de S
end while
if $|C| \leq k$ **then**
    return C
else
    return -1
end if

Para avaliar os algoritmos serão utilizadas, além das métricas numéricas tradicionais, as métricas de silhueta e índice de Rand ajustado. A métrica de silhueta, para cada cluster, é calculada pela equação $\frac{(b-a)}{max(a,b)}$, sendo a a distância média intra-cluster, e b distância média para o segundo cluster mais próximo (já que o mais próximo é o qual o ponto pertence) b, ambos entre todos os pontos do cluster. A métrica final de silhueta é então obtida pela média da silhueta de todos os seus clusters, e é situada no intervalo

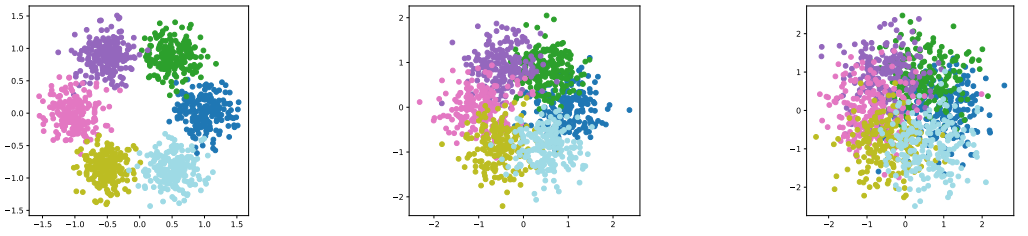
[1,-1], no qual valores próximos de 0 indicam clusters que se sobrepõem, próximos de -1 indicando erros de atribuição, e próximos de 1 sendo o melhor resultado.

No caso do índice de Rand ajustado, a métrica compara o grau de similaridade entre o clustering original esperado (dado pela instância) com o clustering realizado pelo algoritmo, sendo ajustado de acordo com a chance utilizando a equação $\frac{RI - RI_{esperado}}{max(RI) - RI_esperado}$ sendo RI o indíce de Rand sem ajustes. O valor esperado é 1 caso o clustering seja igual, por volta de 0 quando o clustering é feito aleatoriamente, e por volta de -0.5 no caso de clusterings discordantes.

3. Metodologia

Ambos os algoritmos, juntamente com a lógica de refinamento necessária, foram implementados utilizando a linguagem Python-3.12, sem grandes modificações. No caso do algoritmo de Lloyd, silhueta e índice de Rand ajustado, foi utilizada a implementação da biblioteca *scikit-learn* [Pedregosa et al. 2011]. Métodos auxiliares também foram estabelecidos, como a construção da função de distância através da distância de minkowski e a atribuição de cada cluster para cada ponto. Cada instância proposta do problema foi então executada em uma thread do processador Ryzen 5 3600, com frequência 3.6GHz (4.2GHz turbo), no sistema operacional Arch Linux. Para o caso específico do algoritmo de refinamento, foram utilizada uma distribuição exponencial de larguras $L = \{1.5625, 3.125, 6.25, 12.5, 25\}$, em que cada uma será testada individualmente. Cada teste, composto por uma instância, método, valor de p na distância de Minkowski e largura (no caso do algoritmo de refinamento), foi executado um total de 30 vezes, permutando os valores de P a cada execução, visto que a ordem impacta a qualidade do algoritmo.

Para a testagem dos 3 algoritmos, foram utilizadas instâncias sintéticas e reais. Para a criação das instâncias sintéticas, inicialmente foram feitas 10 instâncias em duas dimensões, em que foram estabelecidos centros nos quais, juntamente com uma matriz de covariância, foram criados pontos em volta desses pontos seguindo uma distribuição normal multivariada. Para cada instância, as posições dos centros foram estabelecidas manualmente, geralmente relembrando alguma forma geométrica, e a matriz de covariância foi ajustada para criar situações em que pontos de um cluster possam apresentar comportamento em que eles estejam completamente separados ou até completamente sobrepostos como demonstrado na figura 1. Foram estabelecidos, em média, 250 pontos por cluster em cada caso.

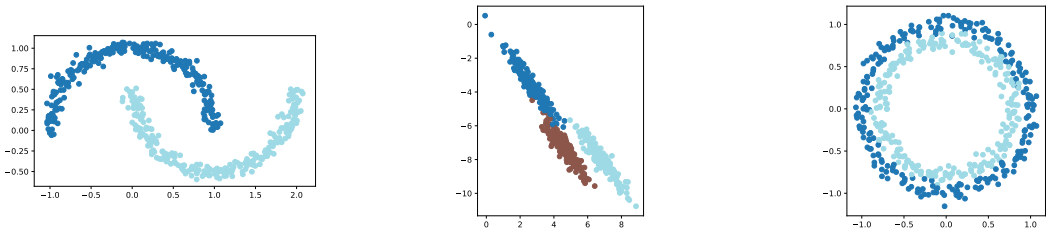


(a) Praticamente separados **(b) Levemente sobrepostos** **(c) Fortemente sobrepostos**

Figura 1. Exemplo de instâncias sintéticas geradas

Além disso, também foram estabelecidas 10 instâncias sintéticas adicionais utili-

zando o *scikit-learn*, que fornece algumas distribuições de pontos específicas. Em particular, foram utilizados os datasets *moons*, *blobs* e *circles*, assim como uma distribuição totalmente aleatória, para 5 instâncias em duas dimensões, nos quais foram gerados com um valor pequeno de *noise*. Além disso, também foram feitas 5 instâncias com pontos em 10 dimensões, representando situações mais complexas, através do *make_classification*, variando a quantidade de dimensões que possuem valores informativos (úteis), redundantes, duplicados e totalmente inúteis (aleatórios).



(a) moons (b) blobs anisotrópico (c) circles

Figura 2. Exemplo de instâncias sintéticas geradas

Finalmente, também foram escolhidas também 10 instâncias utilizando dados reais disponibilizados na UCI Machine Learning Repository, que inclui problemas de classificação de diversas áreas do conhecimento. Por questões de memória, instâncias foram limitadas a um número de 4000 pontos, sendo extraído um *sample* de 4000 pontos caso a instância ultrapasse esse limite. Além disso, foram extraídas as classificações das instâncias, e os valores numéricos apresentados foram convertidos em pontos, ignorando outros valores discretos. Foram utilizados os datasets: [Nash and Ford 1995], [Lohweg 2013], [bea 2020], [Coraddu and Figari 2014], [obe 2019], [Çinar and Tasdemir 2023], [ric 2019], [Bock 2007], [Cortez and Reis 2009], [Nakai 1996].

4. Resultados

Iniciando pela distância de Minkowski com $p = 2$ (distância euclideana), os valores métricos brutos encontrados para as 30 execuções por teste estão nas tabelas 1, 2 e 3, denotando as instâncias sintéticas, do scikit e reais, respectivamente. Em seguida, na Tabela 4 temos, por questão de tamanho, somente os resultados para as instâncias sintéticas do scikit utilizando a distância de Minkowski com $p = 1$. Os dois valores de p apresentaram resultados similares, logos serão analisados conjuntamente.

4.1. Análise comparativa entre os 3 algoritmos

Inicialmente, é possível analisar como o algoritmo guloso apresenta consistentemente, entre todas as instâncias, o maior raio médio máximo quando comparado a outros métodos, exceto em algumas instâncias reais, em que o método do k-means do scikit acaba perdendo, enquanto o desvio padrão do raio permanece inconsistente. Isso se dá pela simplicidade do algoritmo, o qual geralmente também apresenta o menor tempo de execução, podendo ser também bem sensível à ordem de S. A escolha iterativa do ponto mais distante geralmente acaba gerando clusters mais parcialmente vazios, que apresentariam

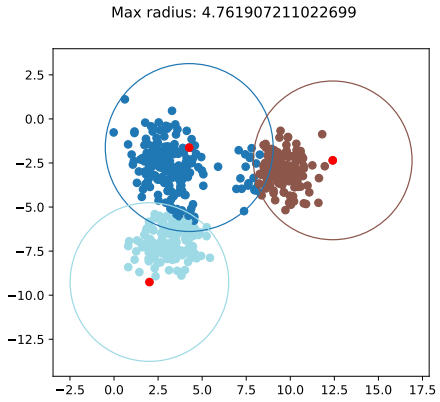
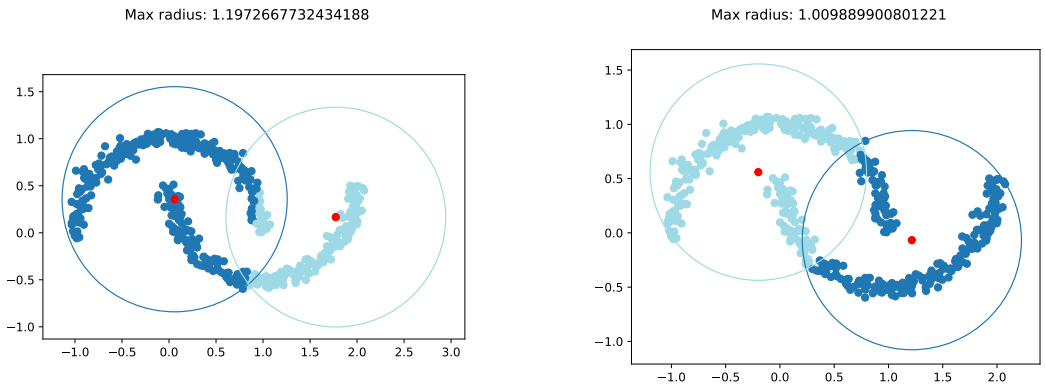


Figura 3. Execução de *blobs* utilizando o algortimo guloso

raios menores casos fossem posicionados em um ponto mais próximo, como demonstrado na figura 3, em que metade dos clusters azul-claro e marrom estão vazios.

Em relação ao algoritmo de Lloyd implementado pelo scikit, apesar de se tratar de um algoritmo para outro problema que não busca minimizar exatamente o maior raio (mas sim a ”inércia”de cada ponto), ele demonstrou raio médios muito bons com desvios padrão baixos, geralmente menores comparados aos outros métodos em instâncias sintéticas geradas e pelo scikit, com tempos de execução pequenos. Porém, ele não se demonstrou muito bom em instâncias reais, apresentando em certos casos um raio médio maior do que o algoritmo guloso, o que pode ser ao objetivo do problema de k-means ser minimizar a inércia intra-cluster e não o raio máximo: em situações com muitos pontos, um número baixo de pontos posicionados relativamente longe de seu cluster não afeta muito a inércia intra-cluster, porém pode gerar um raio máximo muito maior. Note também que o algoritmo de Lloyd pode gerar centros fora do conjunto de pontos S, o que não ocorre nos dois outros algoritmos implementados, o que pode dar uma vantagem excepcional a esse algoritmo em algumas instâncias, como demonstrado na figura 4.

Figura 4. Posição dos clusters em *moons* utilizando diferentes algoritmos



(a) Refinamento com $l = 0.015625$ **(b) Lloyd**

O algoritmo de refinamento, por sua vez, demonstrou-se como o algoritmo com maior de tempo de execução, mas demonstrou-se comparável ao algoritmo de LLoyd em

instâncias sintéticas, principalmente as maiores, enquanto se apresentou como o melhor na maioria das instâncias reais, superando o algoritmo de Lloyd pelas razões explícitas no parágrafo anterior.

Em relação às métricas de silhueta e índice de Rand ajustado, ambos se demonstraram aproximadamente condizentes com as análises feitas pelo raio máximo nas instâncias sintéticas, com o algoritmo de Lloyd apresentando valores maiores, seguido pelo algoritmo de refinamento e finalmente, o guloso. Um detalhe a se notar é que, nas instâncias reais, apesar do raio obtido possívelmente sendo muito maior nas instâncias reais pelo algoritmo Lloyd, muitas vezes ele acaba apresentando um score de silhueta e índice de Rand maior ou similar do que o algoritmo de refinamento/guloso, indicando que o problema das k-médias pode ser mais apropriado para a tarefa de classificação na maioria das instâncias.

4.2. Análise na variação do parâmetro de largura

Quanto à variação do parâmetro de largura no algoritmo de refinamento, não foi observado um resultado consistente no que diz respeito à qualidade das soluções. Isso pode ter ocorrido devido ao fato de o maior parâmetro de largura utilizado ter sido $0.25 * r_{max}$, o que claramente é suficiente para entregar bons resultados para 4 grupos dispostos em linha, o que seria o pior caso para essa quantidade de grupos. Além disso, apesar de L ser exponencial, como o refinamento é feito por busca binária, a redução de uma largura pela metade implica apenas em uma iteração adicional. Pode haver, também, uma sensibilidade do algoritmo à ordem de S. Dessa forma, para um k pequeno, um refinamento até 25% do intervalo original parece ser suficiente na maior parte dos casos, sendo as diferenças nos resultados atribuíveis à ordem arbitrária da escolha dos pontos em cada execução.

5. Conclusão

Nesse trabalho, foi abordado o problema dos k-centros, um problema NP-difícil clássico de agrupamento. Foram comparados em diversas métricas e utilizando diversos datasets dois algoritmos 2-aproximativos e o algoritmo de Lloyd para o k-means.

Foi obsevado que o algoritmo guloso, apesar de possuir muito maior eficiência em tempo de execução, consistentemente perde para os outros na qualidade das soluções. O algoritmo de Lloyd, por outro lado, produz ótimos resultados em relação aos algoritmos aproximativos nas instâncias sintéticas, mas apresentou raios máximos piores nas instâncias reais, apesar de ainda apresentar métricas de silhueta e indíce de Rand ajustado superiores ou similares aos outros algortimos, indicando que o problema das k-médias é mais apropriado para o problema de classificação nesses casos. Por fim o algoritmo de refinamento apresenta soluções com uma qualidade intermediária nas instâncias sintéticas, porém apresentou também os melhores resultados em relação ao raio máximo nas instâncias reais, mas com um tempo de execução consideravelmente maior.

De fato, foi observado que, para as instâncias utilizadas, o intervalo do algoritmo de refinamento impactou muito pouco a solução, o que pode ser devido ao k pequeno das instâncias utilizadas, sensibilidade do algoritmo à ordem de S, e a redução exponencial do intervalo.

Tabela 1. Valores obtidos nas instâncias sintéticas geradas, com $p = 2$

		radius		silhouette		adj_rand_score		exec_time	
		mean	std	mean	std	mean	std	mean	std
instance	method								
triangle	greedy	1.67	0.06	0.70	0.03	0.94	0.06	0.00	0.00
	refining0.015625	1.50	0.17	0.72	0.03	0.98	0.05	0.17	0.01
	refining0.03125	1.48	0.17	0.72	0.02	0.99	0.04	0.17	0.01
	refining0.0625	1.43	0.17	0.73	0.00	0.99	0.01	0.17	0.01
	refining0.125	1.42	0.13	0.73	0.01	0.99	0.01	0.17	0.01
	refining0.25	1.43	0.15	0.73	0.01	0.99	0.01	0.17	0.01
	scikit	1.08	0.00	0.73	0.00	1.00	0.00	0.01	0.00
squares-leve-sobreposto	greedy	2.53	0.16	0.34	0.08	0.50	0.15	0.00	0.00
	refining0.015625	2.13	0.22	0.40	0.05	0.63	0.10	0.28	0.02
	refining0.03125	2.08	0.20	0.41	0.05	0.66	0.11	0.28	0.03
	refining0.0625	2.24	0.30	0.36	0.10	0.58	0.16	0.27	0.03
	refining0.125	2.18	0.24	0.39	0.07	0.62	0.12	0.28	0.03
	refining0.25	2.17	0.23	0.38	0.07	0.59	0.14	0.27	0.03
	scikit	2.11	0.00	0.48	0.00	0.85	0.00	0.02	0.01
hexagonsobreposto	greedy	1.50	0.10	0.17	0.04	0.28	0.07	0.01	0.00
	refining0.015625	1.43	0.11	0.26	0.03	0.40	0.05	0.59	0.06
	refining0.03125	1.38	0.11	0.27	0.03	0.40	0.06	0.59	0.06
	refining0.0625	1.40	0.11	0.26	0.03	0.39	0.04	0.57	0.06
	refining0.125	1.44	0.09	0.26	0.03	0.39	0.04	0.59	0.07
	refining0.25	1.40	0.08	0.26	0.03	0.39	0.04	0.58	0.05
	scikit	1.31	0.04	0.37	0.01	0.56	0.04	0.04	0.01
square	greedy	1.79	0.08	0.65	0.03	0.91	0.05	0.00	0.00
	refining0.015625	1.42	0.12	0.68	0.02	0.97	0.03	0.30	0.03
	refining0.03125	1.44	0.13	0.67	0.02	0.96	0.04	0.29	0.03
	refining0.0625	1.39	0.15	0.68	0.02	0.97	0.03	0.30	0.03
	refining0.125	1.49	0.14	0.67	0.02	0.96	0.04	0.30	0.03
	refining0.25	1.46	0.12	0.67	0.02	0.95	0.04	0.29	0.03
	scikit	1.10	0.00	0.69	0.00	1.00	0.00	0.02	0.00
trianglesobreposto	greedy	3.01	0.15	0.31	0.06	0.31	0.13	0.00	0.00
	refining0.015625	2.79	0.27	0.37	0.04	0.46	0.10	0.16	0.02
	refining0.03125	2.81	0.20	0.36	0.06	0.45	0.13	0.16	0.01
	refining0.0625	2.84	0.24	0.36	0.06	0.45	0.13	0.15	0.02
	refining0.125	2.74	0.24	0.37	0.05	0.48	0.11	0.16	0.02
	refining0.25	2.81	0.36	0.36	0.06	0.45	0.13	0.15	0.02
	scikit	2.25	0.00	0.45	0.00	0.64	0.00	0.08	0.04
linemessy	greedy	2.82	0.22	0.31	0.09	0.19	0.11	0.00	0.00
	refining0.015625	2.41	0.30	0.36	0.06	0.25	0.08	0.15	0.01
	refining0.03125	2.49	0.24	0.36	0.05	0.26	0.06	0.15	0.02
	refining0.0625	2.44	0.26	0.35	0.06	0.24	0.08	0.15	0.01
	refining0.125	2.39	0.23	0.35	0.05	0.26	0.07	0.16	0.02
	refining0.25	2.41	0.28	0.37	0.06	0.27	0.07	0.16	0.01
	scikit	2.72	0.02	0.40	0.00	0.34	0.01	0.14	0.08
hexagon	greedy	1.09	0.03	0.41	0.07	0.67	0.10	0.01	0.00
	refining0.015625	0.95	0.09	0.49	0.05	0.79	0.08	0.62	0.05
	refining0.03125	0.92	0.08	0.49	0.04	0.79	0.06	0.63	0.05
	refining0.0625	0.91	0.06	0.49	0.04	0.79	0.07	0.60	0.07
	refining0.125	0.93	0.09	0.49	0.05	0.79	0.09	0.62	0.05
	refining0.25	0.91	0.06	0.49	0.04	0.79	0.07	0.61	0.07
	scikit	0.79	0.09	0.55	0.05	0.91	0.09	0.03	0.01
lineblobs	greedy	2.75	0.23	0.28	0.07	0.25	0.10	0.00	0.00
	refining0.015625	2.43	0.18	0.32	0.03	0.36	0.06	0.23	0.02
	refining0.03125	2.49	0.17	0.30	0.06	0.30	0.10	0.23	0.03
	refining0.0625	2.39	0.18	0.32	0.03	0.34	0.05	0.24	0.02
	refining0.125	2.51	0.23	0.30	0.04	0.30	0.07	0.22	0.02
	refining0.25	2.45	0.19	0.30	0.05	0.29	0.08	0.22	0.03
	scikit	2.09	0.01	0.39	0.00	0.52	0.02	0.01	0.00
hexagonmuitosobre	greedy	1.82	0.12	0.14	0.04	0.13	0.06	0.01	0.00
	refining0.015625	1.72	0.13	0.25	0.03	0.23	0.05	0.58	0.06
	refining0.03125	1.68	0.17	0.24	0.04	0.22	0.05	0.57	0.06
	refining0.0625	1.71	0.09	0.25	0.02	0.24	0.03	0.59	0.05
	refining0.125	1.65	0.12	0.24	0.03	0.23	0.05	0.58	0.04
	refining0.25	1.67	0.13	0.25	0.03	0.23	0.05	0.58	0.05
	scikit	1.82	0.08	0.32	0.01	0.28	0.02	0.04	0.01
squaresmuitosobre	greedy	3.16	0.19	0.23	0.07	0.25	0.13	0.00	0.00
	refining0.015625	2.74	0.19	0.29	0.05	0.35	0.08	0.26	0.03
	refining0.03125	2.84	0.17	0.27	0.06	0.33	0.09	0.26	0.03
	refining0.0625	2.95	0.27	0.25	0.08	0.28	0.12	0.26	0.02
	refining0.125	2.86	0.32	0.28	0.07	0.35	0.11	0.26	0.02
	refining0.25	2.86	0.22	0.28	0.06	0.33	0.09	0.26	0.03
	scikit	2.89	0.01	0.37	0.00	0.57	0.00	0.03	0.01

Tabela 2. Valores obtidos nas instâncias do scikit com $p = 2$

		radius		silhouette		adj_rand_score		exec_time	
		mean	std	mean	std	mean	std	mean	std
instance	method								
moons	greedy	1.55	0.14	0.44	0.03	0.20	0.08	0.00	0.00
	refining0.015625	1.35	0.14	0.47	0.02	0.26	0.06	0.11	0.01
	refining0.03125	1.61	0.13	0.47	0.01	0.23	0.06	0.11	0.01
	refining0.0625	1.51	0.16	0.47	0.01	0.23	0.05	0.11	0.01
	refining0.125	1.56	0.19	0.45	0.02	0.21	0.07	0.11	0.01
	refining0.25	1.43	0.19	0.46	0.03	0.26	0.08	0.11	0.01
	scikit	1.02	0.00	0.49	0.00	0.24	0.00	0.49	0.51
blobs	greedy	5.21	0.41	0.46	0.08	0.60	0.14	0.00	0.00
	refining0.015625	4.98	0.18	0.59	0.03	0.83	0.07	0.13	0.01
	refining0.03125	4.53	0.50	0.58	0.06	0.82	0.14	0.13	0.01
	refining0.0625	4.08	0.19	0.63	0.02	0.92	0.05	0.13	0.01
	refining0.125	4.66	0.80	0.57	0.06	0.79	0.12	0.12	0.01
	refining0.25	4.49	0.41	0.56	0.08	0.79	0.17	0.12	0.01
	scikit	4.21	0.00	0.65	0.00	0.97	0.00	0.01	0.01
circles	greedy	1.48	0.05	0.39	0.00	-0.00	0.00	0.00	0.00
	refining0.015625	1.56	0.03	0.40	0.00	-0.00	0.00	0.11	0.01
	refining0.03125	1.45	0.11	0.39	0.00	-0.00	0.00	0.11	0.01
	refining0.0625	1.63	0.15	0.40	0.00	-0.00	0.00	0.11	0.01
	refining0.125	1.60	0.12	0.40	0.00	-0.00	0.00	0.11	0.01
	refining0.25	1.47	0.07	0.40	0.00	-0.00	0.00	0.11	0.01
	scikit	1.20	0.03	0.40	0.00	-0.00	0.00	0.01	0.01
high-order-informative	greedy	12.59	1.12	0.15	0.03	0.00	0.00	0.00	0.00
	refining0.015625	13.14	0.43	0.10	0.02	0.01	0.01	0.10	0.01
	refining0.03125	13.61	0.38	0.11	0.02	0.01	0.01	0.10	0.01
	refining0.0625	13.09	0.78	0.14	0.04	0.01	0.01	0.10	0.01
	refining0.125	12.61	0.52	0.15	0.04	0.00	0.01	0.10	0.01
	refining0.25	12.77	0.68	0.12	0.03	0.00	0.01	0.11	0.02
	scikit	10.48	0.07	0.11	0.00	0.02	0.00	0.01	0.01
ho-slightly-redundant	greedy	14.09	0.92	0.17	0.02	0.00	0.00	0.00	0.00
	refining0.015625	12.71	0.59	0.20	0.04	0.01	0.03	0.10	0.01
	refining0.03125	12.93	0.91	0.20	0.05	0.02	0.04	0.10	0.01
	refining0.0625	13.48	0.93	0.19	0.03	0.03	0.04	0.10	0.01
	refining0.125	13.92	0.61	0.20	0.03	0.02	0.04	0.10	0.01
	refining0.25	13.55	0.56	0.19	0.03	0.01	0.02	0.10	0.01
	scikit	12.29	0.05	0.20	0.00	0.06	0.00	0.01	0.01
ho-redundant	greedy	10.03	0.78	0.32	0.05	0.07	0.05	0.00	0.00
	refining0.015625	10.10	0.92	0.35	0.05	0.11	0.08	0.11	0.01
	refining0.03125	9.72	1.34	0.35	0.05	0.14	0.06	0.11	0.01
	refining0.0625	9.38	0.94	0.36	0.04	0.16	0.08	0.11	0.01
	refining0.125	9.40	1.02	0.35	0.05	0.14	0.06	0.11	0.01
	refining0.25	9.59	0.76	0.33	0.06	0.13	0.09	0.11	0.01
	scikit	7.97	0.00	0.41	0.00	0.21	0.00	0.01	0.01
ho-useless	greedy	8.81	1.00	0.15	0.03	0.03	0.03	0.00	0.00
	refining0.015625	8.22	0.61	0.16	0.03	0.02	0.01	0.10	0.01
	refining0.03125	8.26	0.46	0.17	0.03	0.02	0.01	0.10	0.01
	refining0.0625	8.56	0.56	0.15	0.03	0.03	0.03	0.10	0.01
	refining0.125	8.69	0.63	0.14	0.03	0.02	0.03	0.10	0.01
	refining0.25	8.57	0.61	0.14	0.03	0.04	0.06	0.10	0.01
	scikit	6.74	0.19	0.14	0.01	0.03	0.06	0.01	0.00
ho-general	greedy	11.11	0.86	0.23	0.04	0.01	0.02	0.00	0.00
	refining0.015625	10.47	0.56	0.22	0.04	0.01	0.01	0.10	0.01
	refining0.03125	10.76	0.63	0.23	0.05	0.00	0.01	0.10	0.01
	refining0.0625	10.77	0.74	0.23	0.05	0.01	0.01	0.10	0.01
	refining0.125	10.72	0.85	0.23	0.04	0.01	0.01	0.10	0.01
	refining0.25	11.03	1.32	0.19	0.04	0.01	0.02	0.10	0.01
	scikit	9.73	0.24	0.21	0.00	0.01	0.00	0.01	0.00
aniso	greedy	4.17	0.52	0.33	0.15	0.30	0.16	0.00	0.00
	refining0.015625	3.47	0.34	0.47	0.05	0.44	0.05	0.12	0.01
	refining0.03125	3.97	0.34	0.48	0.05	0.47	0.05	0.12	0.01
	refining0.0625	3.82	0.44	0.46	0.04	0.44	0.06	0.12	0.01
	refining0.125	3.72	0.52	0.48	0.05	0.43	0.06	0.12	0.01
	refining0.25	3.70	0.41	0.48	0.05	0.45	0.06	0.12	0.01
	scikit	4.78	0.02	0.53	0.00	0.65	0.01	0.01	0.01
random	greedy	0.87	0.08	0.31	0.02	-0.00	0.00	0.00	0.00
	refining0.015625	0.78	0.10	0.32	0.03	-0.00	0.00	0.10	0.01
	refining0.03125	0.81	0.08	0.32	0.03	0.00	0.00	0.10	0.01
	refining0.0625	0.84	0.09	0.32	0.03	0.00	0.00	0.10	0.01
	refining0.125	0.84	0.07	0.31	0.03	0.00	0.00	0.10	0.01
	refining0.25	0.79	0.09	0.32	0.03	-0.00	0.00	0.10	0.01
	scikit	0.56	0.00	0.37	0.01	-0.00	0.00	0.02	0.02

Tabela 3. Valores obtidos nas instâncias reais com $p = 2$

		radius		silhouette		adj_rand_score		exec_time	
		mean	std	mean	std	mean	std	mean	std
instance	method								
obesity	greedy	24.18	1.37	0.37	0.03	0.27	0.02	0.02	0.00
	refining0.015625	21.95	1.06	0.40	0.04	0.29	0.03	1.83	0.12
	refining0.03125	21.94	1.92	0.40	0.03	0.28	0.02	1.81	0.12
	refining0.0625	21.63	1.42	0.38	0.04	0.29	0.03	1.83	0.15
	refining0.125	22.09	1.53	0.38	0.05	0.28	0.03	1.82	0.14
	refining0.25	21.91	1.76	0.38	0.05	0.28	0.03	1.78	0.22
	scikit	36.44	1.74	0.45	0.02	0.31	0.01	0.05	0.00
maintenance	greedy	30.33	3.37	0.43	0.01	0.13	0.12	0.01	0.00
	refining0.015625	30.74	4.68	0.45	0.03	0.10	0.13	4.53	0.59
	refining0.03125	31.05	3.97	0.44	0.02	0.11	0.12	4.52	0.50
	refining0.0625	31.33	3.69	0.44	0.02	0.08	0.11	4.53	0.50
	refining0.125	29.89	4.27	0.45	0.02	0.11	0.12	4.66	0.40
	refining0.25	29.71	4.74	0.45	0.03	0.08	0.12	4.60	0.49
	scikit	29.37	0.06	0.49	0.00	0.00	0.00	0.04	0.00
telescope	greedy	502.37	26.02	0.69	0.02	0.00	0.00	0.01	0.00
	refining0.015625	523.60	39.46	0.65	0.04	0.01	0.02	4.33	0.53
	refining0.03125	516.72	24.96	0.65	0.11	0.01	0.01	4.30	0.52
	refining0.0625	522.46	26.49	0.64	0.12	0.01	0.01	4.41	0.53
	refining0.125	519.27	31.51	0.63	0.12	0.01	0.01	4.40	0.50
	refining0.25	502.09	35.53	0.67	0.03	0.00	0.01	4.53	0.34
	scikit	598.90	0.05	0.43	0.00	0.06	0.00	0.05	0.01
wine_quality	greedy	78.69	5.41	0.38	0.03	-0.01	0.00	0.04	0.00
	refining0.015625	73.17	3.98	0.42	0.03	-0.00	0.00	5.60	0.66
	refining0.03125	73.12	5.31	0.41	0.03	-0.00	0.00	5.67	0.60
	refining0.0625	73.58	5.55	0.42	0.03	-0.00	0.00	5.61	0.67
	refining0.125	73.78	6.45	0.42	0.03	-0.01	0.00	5.58	0.52
	refining0.25	73.73	4.19	0.42	0.03	-0.00	0.00	5.67	0.53
	scikit	301.80	46.88	0.37	0.01	0.00	0.00	0.07	0.01
yeast	greedy	0.58	0.03	0.17	0.05	0.06	0.04	0.03	0.00
	refining0.015625	0.55	0.02	0.22	0.05	0.10	0.04	0.77	0.09
	refining0.03125	0.54	0.02	0.20	0.06	0.10	0.05	0.76	0.09
	refining0.0625	0.54	0.02	0.19	0.05	0.10	0.04	0.76	0.09
	refining0.125	0.55	0.02	0.19	0.05	0.09	0.04	0.76	0.10
	refining0.25	0.54	0.02	0.20	0.05	0.08	0.03	0.76	0.08
	scikit	0.75	0.05	0.17	0.01	0.13	0.01	0.05	0.01
banknote	greedy	15.49	1.55	0.45	0.03	0.07	0.02	0.00	0.00
	refining0.015625	14.64	1.61	0.45	0.03	0.06	0.02	0.65	0.07
	refining0.03125	15.40	2.00	0.45	0.03	0.06	0.02	0.64	0.08
	refining0.0625	15.52	1.40	0.44	0.03	0.05	0.02	0.64	0.07
	refining0.125	14.78	1.59	0.44	0.03	0.06	0.03	0.66	0.08
	refining0.25	15.27	2.19	0.43	0.03	0.05	0.02	0.64	0.06
	scikit	15.46	0.01	0.43	0.00	0.05	0.00	0.02	0.00
rice	greedy	87.06	10.39	0.44	0.10	0.26	0.22	0.01	0.00
	refining0.015625	79.39	13.74	0.54	0.09	0.50	0.22	4.63	0.64
	refining0.03125	80.14	11.81	0.53	0.08	0.46	0.21	4.57	0.51
	refining0.0625	82.95	12.82	0.52	0.09	0.45	0.22	4.51	0.57
	refining0.125	72.89	12.09	0.54	0.09	0.50	0.22	4.73	0.58
	refining0.25	78.34	12.60	0.52	0.11	0.46	0.24	4.52	0.70
	scikit	75.19	0.03	0.61	0.00	0.69	0.00	0.04	0.00
abalone	greedy	0.30	0.01	0.23	0.05	0.07	0.00	0.45	0.04
	refining0.015625	0.28	0.01	0.29	0.03	0.07	0.00	6.23	0.61
	refining0.03125	0.28	0.01	0.28	0.06	0.07	0.00	6.17	0.60
	refining0.0625	0.28	0.01	0.30	0.04	0.07	0.00	6.24	0.66
	refining0.125	0.28	0.01	0.27	0.04	0.07	0.00	6.21	0.65
	refining0.25	0.28	0.01	0.28	0.05	0.07	0.00	6.21	0.65
	scikit	0.81	0.27	0.27	0.01	0.05	0.00	0.12	0.01
beans	greedy	169.62	9.40	0.39	0.03	0.30	0.06	0.04	0.00
	refining0.015625	163.31	15.60	0.45	0.04	0.35	0.05	6.24	0.67
	refining0.03125	166.58	14.11	0.42	0.04	0.33	0.05	6.25	0.64
	refining0.0625	168.20	15.26	0.44	0.04	0.34	0.06	6.21	0.72
	refining0.125	168.28	16.27	0.43	0.05	0.35	0.06	6.18	0.70
	refining0.25	167.15	14.13	0.43	0.04	0.35	0.06	6.18	0.57
	scikit	262.04	41.05	0.46	0.00	0.39	0.01	0.07	0.01
raisins	greedy	907.42	91.85	0.65	0.05	0.00	0.00	0.00	0.00
	refining0.015625	828.73	136.36	0.61	0.05	0.06	0.11	0.31	0.04
	refining0.03125	826.52	134.55	0.61	0.05	0.07	0.11	0.32	0.03
	refining0.0625	796.21	102.49	0.61	0.04	0.05	0.07	0.32	0.03
	refining0.125	833.61	120.38	0.60	0.04	0.10	0.10	0.32	0.03
	refining0.25	826.00	91.11	0.63	0.05	0.03	0.06	0.32	0.03
	scikit	1268.06	6.27	0.58	0.00	0.35	0.02	0.01	0.00

Tabela 4. Valores obtidos nas instâncias do scikit com $p = 1$

		radius		silhouette		adj_rand_score		exec_time	
		mean	std	mean	std	mean	std	mean	std
instance	method								
moons	greedy	2.16	0.23	0.44	0.06	0.27	0.13	0.00	0.00
	refining0.015625	1.89	0.30	0.44	0.05	0.22	0.09	0.11	0.01
	refining0.03125	1.88	0.20	0.42	0.07	0.23	0.12	0.11	0.01
	refining0.0625	2.17	0.27	0.43	0.07	0.28	0.14	0.10	0.01
	refining0.125	1.99	0.33	0.45	0.03	0.29	0.10	0.11	0.01
	refining0.25	2.11	0.24	0.45	0.02	0.27	0.11	0.11	0.01
	scikit	1.40	0.01	0.49	0.00	0.23	0.00	0.50	0.53
blobs	greedy	6.67	0.37	0.48	0.08	0.66	0.14	0.00	0.00
	refining0.015625	5.91	0.42	0.60	0.02	0.88	0.05	0.12	0.01
	refining0.03125	5.91	0.40	0.58	0.04	0.85	0.09	0.13	0.01
	refining0.0625	6.32	0.60	0.54	0.07	0.77	0.13	0.12	0.01
	refining0.125	6.01	0.59	0.56	0.06	0.80	0.12	0.12	0.01
	refining0.25	5.77	0.51	0.59	0.03	0.86	0.08	0.12	0.01
	scikit	5.82	0.00	0.62	0.00	0.97	0.00	0.02	0.02
circles	greedy	1.79	0.21	0.38	0.01	-0.00	0.00	0.00	0.00
	refining0.015625	1.86	0.18	0.39	0.01	-0.00	0.00	0.11	0.01
	refining0.03125	1.97	0.18	0.39	0.01	-0.00	0.00	0.11	0.01
	refining0.0625	1.99	0.20	0.39	0.02	-0.00	0.00	0.11	0.01
	refining0.125	1.94	0.18	0.38	0.01	-0.00	0.00	0.11	0.01
	refining0.25	1.94	0.13	0.38	0.01	-0.00	0.00	0.11	0.01
	scikit	1.55	0.05	0.39	0.01	-0.00	0.00	0.04	0.03
high-order-informative	greedy	35.50	2.94	0.13	0.04	0.01	0.01	0.00	0.00
	refining0.015625	34.92	2.54	0.13	0.04	0.00	0.01	0.10	0.01
	refining0.03125	34.77	2.17	0.12	0.03	0.00	0.01	0.10	0.01
	refining0.0625	35.14	3.05	0.14	0.03	0.01	0.01	0.10	0.01
	refining0.125	34.37	3.85	0.14	0.03	0.01	0.01	0.10	0.01
	refining0.25	34.95	2.53	0.13	0.03	0.01	0.01	0.10	0.01
	scikit	31.53	0.19	0.11	0.01	0.02	0.00	0.01	0.00
ho-slightly-redundant	greedy	37.09	2.65	0.16	0.03	0.02	0.03	0.00	0.00
	refining0.015625	36.18	3.24	0.17	0.03	0.02	0.03	0.10	0.01
	refining0.03125	36.66	3.05	0.17	0.02	0.01	0.01	0.10	0.01
	refining0.0625	36.73	3.07	0.17	0.02	0.02	0.02	0.10	0.01
	refining0.125	35.63	1.54	0.18	0.02	0.02	0.04	0.10	0.01
	refining0.25	35.97	2.49	0.16	0.02	0.02	0.03	0.10	0.01
	scikit	31.12	0.14	0.20	0.00	0.06	0.00	0.03	0.03
ho-redundant	greedy	28.09	3.17	0.33	0.05	0.10	0.07	0.00	0.00
	refining0.015625	25.24	2.86	0.36	0.07	0.15	0.08	0.11	0.01
	refining0.03125	26.82	4.15	0.36	0.06	0.15	0.09	0.11	0.01
	refining0.0625	26.80	3.30	0.37	0.05	0.13	0.07	0.11	0.01
	refining0.125	25.84	3.27	0.39	0.04	0.18	0.07	0.11	0.01
	refining0.25	25.93	2.39	0.36	0.06	0.16	0.09	0.11	0.01
	scikit	21.37	0.01	0.43	0.00	0.21	0.00	0.01	0.01
ho-useless	greedy	20.93	1.17	0.13	0.04	0.02	0.02	0.00	0.00
	refining0.015625	21.44	1.28	0.12	0.04	0.03	0.01	0.10	0.01
	refining0.03125	20.66	1.44	0.13	0.04	0.02	0.02	0.10	0.01
	refining0.0625	20.85	1.49	0.15	0.04	0.02	0.01	0.10	0.01
	refining0.125	20.73	1.45	0.13	0.04	0.02	0.01	0.10	0.01
	refining0.25	21.04	1.83	0.13	0.03	0.03	0.02	0.10	0.01
	scikit	17.07	0.05	0.14	0.00	-0.00	0.00	0.03	0.02
ho-general	greedy	29.99	2.26	0.21	0.04	0.01	0.02	0.00	0.00
	refining0.015625	27.65	2.26	0.22	0.03	0.01	0.01	0.10	0.01
	refining0.03125	27.42	2.79	0.23	0.04	0.01	0.01	0.10	0.01
	refining0.0625	28.53	2.45	0.23	0.04	0.01	0.01	0.10	0.01
	refining0.125	27.89	2.43	0.22	0.05	0.01	0.01	0.11	0.01
	refining0.25	29.00	3.19	0.21	0.05	0.01	0.02	0.10	0.01
	scikit	25.29	0.56	0.22	0.00	0.01	0.01	0.01	0.00
aniso	greedy	5.34	0.69	0.33	0.16	0.29	0.16	0.00	0.00
	refining0.015625	4.77	0.77	0.47	0.04	0.41	0.05	0.12	0.01
	refining0.03125	4.92	0.51	0.45	0.03	0.40	0.05	0.12	0.01
	refining0.0625	5.10	0.50	0.49	0.05	0.43	0.04	0.12	0.01
	refining0.125	5.11	0.56	0.46	0.07	0.41	0.08	0.12	0.01
	refining0.25	4.75	0.77	0.46	0.03	0.41	0.04	0.12	0.01
	scikit	6.62	0.03	0.53	0.00	0.65	0.01	0.01	0.01
random	greedy	1.03	0.05	0.30	0.02	0.00	0.00	0.00	0.00
	refining0.015625	0.97	0.11	0.33	0.02	-0.00	0.00	0.10	0.01
	refining0.03125	0.99	0.09	0.32	0.02	0.00	0.00	0.10	0.01
	refining0.0625	1.00	0.07	0.31	0.03	0.00	0.00	0.10	0.01
	refining0.125	1.00	0.08	0.32	0.02	0.00	0.00	0.10	0.01
	refining0.25	0.99	0.08	0.31	0.03	0.00	0.00	0.10	0.01
	scikit	0.74	0.01	0.39	0.01	-0.00	0.00	0.01	0.01

Portanto, temos que o algoritmo guloso pode ser adequado para situações em que o tempo de execução é absolutamente crítico, porém, o algoritmo de Lloyd também consegue produzir, principalmente em instâncias pequenas e médias, resultados em relação ao raio e a classificação ainda melhores do que o caso de refinamento em tempo comparável ao guloso. O algoritmo de refinamento, por sua vez, possui um tempo de execução consideravelmente maior, porém acaba gerando resultados comparáveis ou superiores ao algoritmo de Lloyd em relação ao raio máximo, porém com uma classificação ainda pior do que a solução baseada nas k-médias.

Referências

(2019). Estimation of Obesity Levels Based On Eating Habits and Physical Condition . UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5H31Z.

(2019). Rice (Cammeo and Osmancik). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5MW4Z.

(2020). Dry Bean. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C50S4B.

Bock, R. (2007). MAGIC Gamma Telescope. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C52C8B.

Coraddu, Andrea, O. L. G. A. S. S. A. D. and Figari, M. (2014). Condition Based Maintenance of Naval Propulsion Plants. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5K31K.

Cortez, Paulo, C. A. A. F. M. T. and Reis, J. (2009). Wine Quality. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56S3T.

Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., and Wu, A. (2002). An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892.

Lohweg, V. (2013). Banknote Authentication. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C55P57.

Nakai, K. (1996). Yeast. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5KG68.

Nash, Warwick, S. T. T. S. C. A. and Ford, W. (1995). Abalone. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C55C7W.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Çinar, İlkay, K. M. and Tasdemir, S. (2023). Raisin. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5660T.