Universidade Federal de Pernambuco - CIN

Aprendizagem de Máquina

Relatório do Projeto

Feito por :
Arnaldo Rafael Morais Andrade
arma
Giulio Carvalho Cavalcante
gcc

Conteúdo

1 Introdução

Este projeto tem como proposta replicar e comparar resultados do artigo *"How much can k-means be improved by using better initialization and repeats?"* (Fränti e Sieranoja, 2019 [1]). O objetivo dos autores no artigo analisado é comparar técnicas que podem ser utilizadas para suprir as principais limitações do algoritmo de clusterização k-Means, tentando identificar quais seriam as melhores e sob quais condições.

A popularidade do k-Means como algoritmo de clusterização não é uma surpresa. Embora existam outros algoritmos conhecidos que forneçam melhores resultados de agrupamento que ele, há boas razões que justificam sua alta recorrência. Primeiro, é de fácil implementação. Segundo, pessoas geralmente preferem usar um algoritmo extensivamente estudado com limitações conhecidas ao invés de um potencialmente melhor, porém menos estudado, algoritmo que possa ter limitações desconhecidas. Terceiro, a capacidade de ajuste fino local do k-Means é muito efetiva, o que o faz ser utilizado em outros algoritmos como o Algoritmo Genético, *Random Swap*, PSO, entre outros.

Entre as limitações do método em questão, há uma principal. O k-Means raramente obtém sucesso em otimizar as localizações dos centroides globalmente. A razão está na movimentação dos centroides, que não conseguem realizá-la entre os grupos se a distância deles é grande, ou se há outros grupos estáveis no intermédio impedindo o movimento. A Figura 1, encontrada no próprio artigo a ser analisado [1], ilustra esta desvantagem. Portanto, os resultados do k-Means dependem muito da inicialização. Uma inicialização ruim pode convergir em um mínimo local inferior.

Para compensar a fraqueza do k-Means, duas principais abordagens foram consideradas: (1) utilizar uma inicialização melhor e (2) repetir o algoritmo várias vezes. Diversas técnicas de inicialização já foram apresentadas na literatura, incluindo:

- *Random points*
- *Furthest point heuristic*
- *Sorting heuristic*
- *Density-based*
- *Projection-based*
- *Splitting technique*

Até então, estudos comparativos existiam, mas não há um consenso de qual técnica deveria ser usada. Um claro estado da arte estava faltando, como relatou o artigo.

Os próprios autores fizeram uma pequena investigação de como os artigos recentes

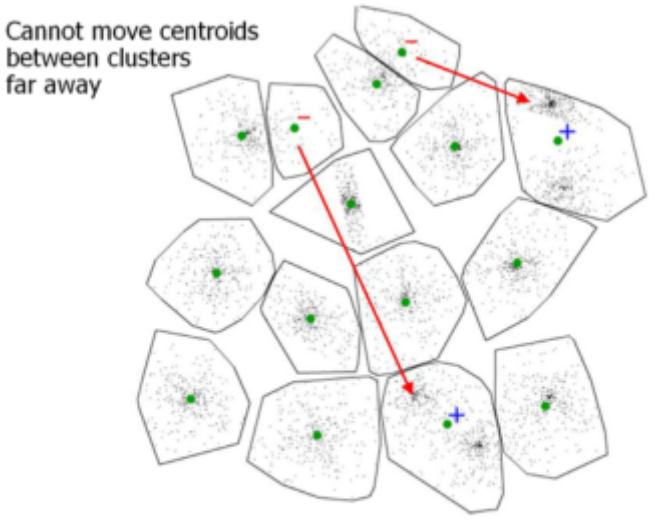

Fig. 1. K-means is excellent in fine-tuning cluster borders locally but fails to relocate the centroids globally. Here a minus sign (−) represents a centroid that is not needed, and a plus sign (+) a cluster where more centroids would be needed. K-means cannot do it because there are stable clusters in between.

Figura 1: Exemplo de dificuldade de movimento de centroides dada uma inicialização não-ideal.

aplicam o k-Means. O método de inicialização Random Centroids aparece na maioria deles, seguido do k-means++. Na questão da repetição, o número varia de uma quantidade relativamente pequena de 10-20 a valores como 100. Além disso, a maioria dos estudos utilizam *datasets* de classificação, que possuem aptidão limitada para avaliar o desempenho da clusterização.

No artigo, são estudadas as mais populares heurísticas de inicialização com o objetivo de responder as seguintes questões:

- A que extensão o k-Means pode ser melhorado por uma técnica inicialização melhor?
- A fraqueza fundamental do algoritmo pode ser eliminada simplesmente por repetições consecutivas?
- Pode-se predizer sob quais condições o k-Means funciona e em quais ele falha?

Sabe-se que o algoritmo tem um desempenho ruim quando os *clusters* estão bem separados. O artigo responde o quanto uma inicialização melhor ou repetições podem compensar esta fraqueza, além de também mostrar que a dimensionalidade não afeta na maioria das variantes e o desbalanço entre os tamanhos dos *clusters* deteriora o desempenho da maioria das inicializações.

2 Metodologia

A fim de evitar conclusões erradas, o artigo faz uma clara distinção entre o método de clusterização e o algoritmo. Segundo ele, o método de clusterização refere-se à função objetivo, como a Soma dos Erros Quadráticos (SSE), por exemplo. Já o algoritmo de clusterização otimiza o processo. Como o foco do artigo se trata da performance do k-Means ao invés da escolha da função objetivo, utiliza-se o *clustering basic benchmark* pois todos os seus *datasets* podem ser clusterizados corretamente com SSE. O que indica que os erros de clusterização obtidos são diretamente relacionados às propriedades do k-Means.

As bases de dados e suas características estão sumarizadas na tabela 1, assim como no artigo. Elas são projetadas para variar as propriedades abaixo:

- Sobreposição de *clusters* (*cluster overlap*)
- Quantidade de *clusters* (*number of clusters*)
- Dimensionalidade (*dimensions*)
- Desbalanço de tamanho de *clusters* (*unbalance of cluster sizes*)

Dataset	Varying	Size	Dimensions	Clusters	Per cluster
A	Number of clusters	3000-7500	2	20-50	150
B	Overlap	5000	2	15	333
Dim	Dimensions	1024	32-1024	16	64
G2	Dimensions, overlap	2048	2-1024	2	1024
Birch	Structure	100000	2	100	1000
Unbalance	Balance	6500	2	8	100-2000

Tabela 1: *Datasets* para *benchmark* de algoritmos de clusterização [3]. Dados disponíveis publicamente em http://cs.uef.fi/sipu/datasets/

Há diversas métricas que medem o sucesso do algoritmo, o valor da própria função objetivo é a mais óbvia delas, como no caso da SSE. Caso os centroides reais (*ground truth*) sejam conhecidos, outras métricas podem ser usadas, que é o caso dos índices externos. O problema, como relata o artigo, é que a SSE bem como alguns índices não dizem o quão significante o resultado é. Por isso é utilizado o *Centroid Index* (CI), que indica a quantidade de centroides localizados erroneamente. O ideal é que esse valor seja 0, indicando uma estrutura de clusterização muito próxima da fornecida pela *ground truth*.

Baseado no CI, uma Taxa de Sucesso pode ser calculada, que pode ser vista como

a probabilidade de encontrar a clusterização correta. Por exemplo, ao executar o k-Means 5000 vezes como *dataset* A2, CI = 0 nunca foi alcançado, e logo, sua Taxa de Sucesso é de 0%, como informa o artigo. A Taxa de Sucesso tem uma importante implicação. Qualquer valor maior que 0% indica que a clusterização correta pode ser obtida simplesmente repetindo o algoritmo.

Essas e outras métricas utilizadas são apresentadas na Tabela 2.

Métrica	Resumo
CI Inicial	Quantidade de centroides com localização errada logo após a inicialização
CI Final	Quantidade de centroides com localização errada após a melhor iteração do k-Means
Taxa de Sucesso	Total de execuções com CI = 0 sobre total de execuções
Número de Iterações	Total de iterações do k-Means necessárias para finalizar a execução
Tempo de Execução	Tempo, em segundos, gasto em uma execução

Tabela 2: Métricas utilizadas e seus resumos.

A Figura 2 ajuda a entender onde as métricas podem ser aplicadas. Para cada saída é possível calcular o CI, ou seja, o CI Inicial e o Final. Ao fim de todo o processo também é possível saber o número de iterações usadas no k-Means para atingir o critério de parada, que é quando a SSE não diminui mais.

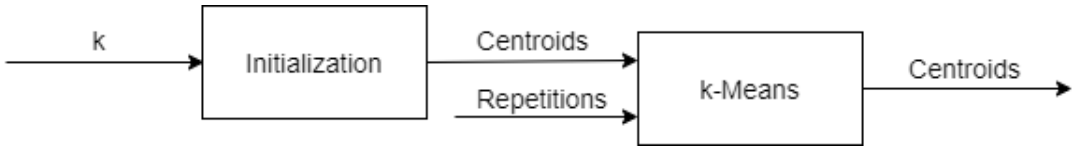

Figura 2: Fluxograma completo do algoritmo k-Means. k é a quantidade de *clusters*, Repetitions é a quantidade de vezes consecutivas que o algoritmo será repetido.

O artigo ainda resume as principais propriedades presentes em uma base dados para clusterização e seu efeito no algoritmo do k-Means, resultando na Tabela 3.

Propriedade	Efeito
Sobreposição de *clusters*	Sobreposição é bom
Quantidade de clusters	Dependência linear
Dimensionalidade	Sem efeito direto
Desbalanço de tamanho de *clusters*	Ruim

Tabela 3: Propriedades e efeitos no k-Means.

Para os experimentos, os métodos de inicialização são apresentados na Tabela 4.

Método de inicialização	Heurística
Random Partitions ou Rand-P	Random
Random Centroids ou Rand-C	Random
Maxmin	Furthest point
kmeans++	Furthest point
Bradley	Repeating
Sorting	Sorting
Projection	Projection
Luxburg	Density
Split	Splitting

Tabela 4: Métodos de inicialização utilizados e sua respectiva heurística

Eles contemplam as principais heurísticas presentes na literatura, como falado anteriormente. No artigo é possível verificar uma breve explicação de cada um dos métodos.

3 Limitações

Antes de prosseguir para os resultados, é importante destacar as dificuldades e limitações encontradas ao longo da reprodução do artigo. Não é informado se houve algum pré-processamento dos dados antes da execução dos experimentos. Também não é dito qual a estratégia utilizada ao encontrar *clusters* vazios durante o andamento do k-Means.

Sobre as técnicas de inicialização, algumas delas não estão claras. Como é no caso de Luxburg, em que o número de *clusters* preliminares é $k * SQRT(k)$, enquanto que no artigo original [2] este número é $k * log(k)$. Além disso, há um filtro aplicado neste conjunto preliminar a fim de remover os menores *clusters*. Entretanto, não é informado o quão pequeno deve ser o *cluster* para ele ser removido do agrupamento inicial. Esta informação também não está clara no artigo original.

Já na técnica Split, o modo como a divisão do *cluster* é feita não é transparente, então foi optado o uso do k-Means com k = 2 para separar o *cluster* em outros dois. O resumo das decisões tomadas para a reprodução dos experimentos estão disponíveis na tabela abaixo.

Método	Decisões
Luxburg	L = k*SQRT(k), Limiar = média(clusters)
Split	split = kmeans(k=2)

Tabela 5: Decisões tomadas de acordo com o método. No Luxburg, L é o tamanho do conjunto de *clusters* preliminar e o limiar é tamanho mínimo do *cluster* a ser considerado. No Split, a divisão é feita pelo algoritmo k-Means com k = 2.

A respeito dos resultados dos experimentos, não é esclarecido a quantidade de execuções feitas em cima do k-Means com repetição (RKM) para gerar a Tabela 5 do artigo. Levando em consideração o custo computacional para operar com o Birch, o número escolhido de execuções foi de 30, gerando assim a Tabela 8.

A Tabela 7 deste relatório é comparada à Tabela 6 do paper estudado, porém com algumas diferenças. Ao invés de ser mostrado a quantidade de repetições necessárias para atingir determinado nível de CI, foi optado por mostrar o menor CI obtido através das 200 iterações. Dessa forma os dados ficam mais concisos e melhor apresentados.

4 Resultados

Para analisar as técnicas de inicialização de forma geral, o artigo apresenta em sua Tabela 13 os resultados da execução de 5000 repetições de cada método em cada *dataset* através das métricas já apresentadas. Além das principais, algumas derivações são feitas: Média (*Aver.*, média aritmética dos valores obtidos pelo método em cada *dataset*), Melhoria (*Impr.*, percentual de melhora entre a média do CI Inicial pro CI Final), Falhas (*Fails*, quantidade de Taxas de Sucesso = 0%).

Para comparar a Tabela 13 do artigo original, foram realizadas as mesmas 5000 repetições e os resultados são apresentados na Tabela 6.

Analisando os valores CI, o Random Partition (12.1) é claramente pior que o Random Centroids (4.6). Das outras técnicas simples, o kmeans++ (0.9) e o Maxmin (2.3) se mostraram superiores. O Luxburg (1.0) e Split (0.6) possuem ótimos desempenhos, mas devem ser considerados separadamente por conta de suas implementações. A observação com a taxa de sucesso é semelhante. Maxmin (21%) e kmeans++ (45%) tem bons desempenhos. Vale destacar que em *datasets* com alto número de *clusters*, como o caso do a3, b1 e b2, os algoritmos têm um desempenho inferior. Por fim, investigando o número de iterações, é possível notar que quanto mais fácil a base de dados é (dim32, por exemplo) e há uma boa inicialização, menor é o número de iterações para atingir o critério de parada do k-Means.

Comparando os resultados da Tabela 6 com os da Tabela 13 do artigo, estes foram condizentes. Porém, devemos destacar que o desempenho do kmeans++ foi bem melhor nos experimentos aqui realizados (CI Final de 2,3 para 0,9 e Taxa de Sucesso de 22% para 45% com apenas uma falha), se tornando o método simples de inicialização com melhor desempenho geralmente. Também é observável a diferença positiva no desempenho do Split. E, em média, os métodos necessitaram de menos iterações para convergirem em comparação ao artigo.

CI-values (initial)													
Method	**s1**	**s2**	**s3**	**s4**	**a1**	**a2**	**a3**	**unb**	**b1**	**b2**	**dim32**	**Aver.**	
Rand-P	12.4	14.0	12.8	14.0	19.0	32.9	48.1	7.0	96.0	96.6	13.1	**33.3**	
Rand-C	5.3	5.4	5.3	5.4	7.1	12.6	18.2	4.6	36.6	36.5	5.7	**13.0**	
Maxmin	1.3	2.9	6.1	6.8	2.1	4.1	5.1	0.9	21.4	9.6	0.0	**5.5**	
kmeans++	0.3	0.7	1.6	1.7	1.1	1.7	2.4	0.1	8.1	1.8	0.0	**1.8**	
Bradley	1.4	1.2	1.2	1.1	1.9	4.3	6.7	2.9	11.7	15.5	1.6	**4.5**	
Sorting	4.1	4.3	4.4	4.5	5.7	10.8	16.0	4.9	34.5	23.9	3.3	**10.6**	
Projection	4.0	4.2	4.4	4.5	5.6	10.5	15.7	4.5	34.0	21.0	2.9	**10.1**	
Luxburg	0.1	0.0	0.1	0.3	1.7	1.7	1.8	4.5	1.7	1.9	0.0	**1.3**	
Split	0.0	0.0	0.0	0.1	0.1	0.0	0.1	4.0	7.5	0.0	0.0	**1.1**	
CI-values (final)													
Method	**s1**	**s2**	**s3**	**s4**	**a1**	**a2**	**a3**	**unb**	**b1**	**b2**	**dim32**	**Aver.**	**Impr.**
Rand-P	4.0	1.8	1.9	1.2	5.4	9.9	19.1	2.3	13.7	69.4	4.0	**12.1**	**64%**
Rand-C	1.9	1.5	1.3	1.0	2.5	4.6	6.6	3.9	7.3	17.3	3.4	**4.6**	**64%**
Maxmin	0.7	1.0	0.7	1.1	1.0	2.6	2.9	0.9	6.0	7.8	0.0	**2.3**	**59%**
kmeans++	0.2	0.4	0.7	0.5	0.6	1.1	1.6	0.1	3.1	1.6	0.0	**0.9**	**49%**
Bradley	1.2	0.9	0.9	0.8	1.6	3.6	5.5	2.4	6.2	13.8	0.8	**3.4**	**24%**
Sorting	1.5	1.2	1.0	0.8	1.7	3.7	5.6	3.9	6.3	9.0	2.3	**3.4**	**68%**
Projection	1.5	1.1	1.0	0.9	1.6	3.6	5.4	4.0	6.0	5.9	2.0	**3.0**	**70%**
Luxburg	0.0	0.0	0.1	0.1	1.2	1.4	1.5	3.7	1.4	1.7	0.0	**1.0**	**20%**
Split	0.0	0.0	0.0	0.1	0.1	0.0	0.1	4.0	2.8	0.0	0.0	**0.6**	**41%**
Success-%													
Method	**s1**	**s2**	**s3**	**s4**	**a1**	**a2**	**a3**	**unb**	**b1**	**b2**	**dim32**	**Aver.**	**Fails**
Rand-P	0%	4%	2%	9%	0%	0%	0%	0%	0%	0%	0%	**1%**	**8**
Rand-C	2%	9%	10%	20%	1%	0%	0%	0%	0%	0%	0%	**4%**	**6**
Maxmin	38%	13%	36%	8%	14%	1%	1%	22%	0%	0%	100%	**21%**	**2**
kmeans++	80%	60%	39%	48%	41%	16%	5%	95%	0%	7%	100%	**45%**	**1**
Bradley	9%	20%	22%	29%	3%	0%	0%	2%	0%	0%	31%	**11%**	**4**
Sorting	6%	16%	19%	29%	4%	0%	0%	0%	0%	0%	2%	**7%**	**5**
Projection	7%	15%	18%	26%	5%	0%	0%	0%	0%	0%	3%	**7%**	**5**
Luxburg	96%	99%	94%	88%	19%	17%	15%	0%	13%	14%	100%	**51%**	**1**
Split	100%	100%	100%	87%	94%	100%	90%	0%	0%	100%	100%	**79%**	**2**
Number of iterations													
Method	**s1**	**s2**	**s3**	**s4**	**a1**	**a2**	**a3**	**unb**	**b1**	**b2**	**dim32**	**Aver.**	
Rand-P	25	22	25	31	29	47	43	10	116	66	4	**38**	
Rand-C	12	16	20	28	17	18	19	5	40	9	5	**17**	
Maxmin	7	13	18	26	10	11	12	4	29	5	2	**12**	
kmeans++	4	8	13	18	7	8	8	3	18	2	2	**8**	
Bradley	8	9	12	16	9	12	13	6	32	7	3	**12**	
Sorting	10	14	19	26	14	16	17	5	36	8	4	**15**	
Projection	10	15	18	25	14	16	17	5	34	7	4	**15**	
Luxburg	3	3	6	9	9	8	8	4	9	2	2	**6**	
Split	2	3	8	10	5	4	4	2	17	2	1	**5**	

Tabela 6: Valor médio de CI antes e depois das iterações do k-Means, taxa de sucesso e o número de iterações feitas. Resultados produzidos com 5000 execuções. O registro de falhas (*Fails*) representa para quantas bases de dados a solução correta nunca foi encontrada (Taxa de Sucesso = 0%). *Impr.* é a melhoria do CI final em relação ao inicial.

Quando analisamos o impacto da sobreposição dos dados estamos observando os *datasets* G2. A sobreposição é calculada em Fränti e Sieranoja, 2018 [3], e apresentada na Figura 7 do artigo aqui analisado. Destes *datasets*, os com valor de sobreposição abaixo de 2% são considerados de "baixa sobreposição"e os outros de "alta sobreposição". Os resultados dos experimentos replicados são apresentados na Figura 3.

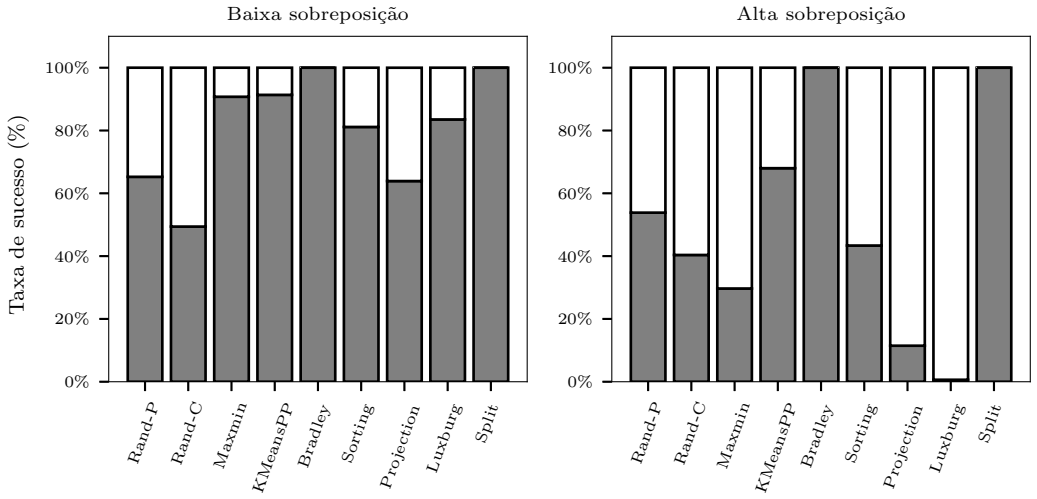

Figura 3: Taxa de sucesso média para todas as bases de dados G2, antes (cinza) e depois do k-Means (branco). As bases foram divididas em duas categorias: com baixa sobreposição (esquerda) e com alta (direita).

Pode-se observar a importância das inicializações quando há baixa sobreposição. Entretanto, se comparado ao artigo, existe uma diferença. A Taxa de Sucesso final é superior. Na alta sobreposição resultados diferentes também foram obtidos. Métodos como Split e Bradley se adaptaram facilmente aos diferentes *datasets*. Já inicializações como Projection e Luxburg obtiveram uma relevante melhora após a execução do k-Means. Todos os métodos conseguiram atingir uma Taxa de Sucesso quase perfeita em ambos os casos.

Ainda sobre a Tabela 6, nota-se que a medida que o número de *clusters* aumenta, como no caso de A1-A3, a performance do algoritmo e da inicialização cai. Além destes *datasets*, os B2-subsets também são utilizados para analisar o impacto de número de *clusters*. Os resultados para comparar com os do artigo estão nas Figuras 4 e 5.

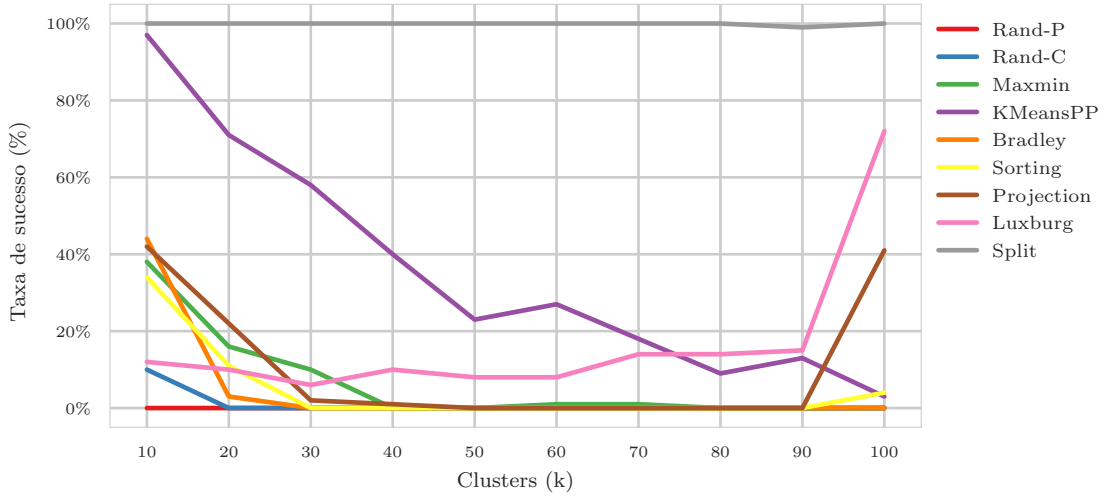

Figura 4: Dependência da taxa de sucesso e do número de *clusters* ao utilizar os subconjuntos do Birch2 (B2-sub).

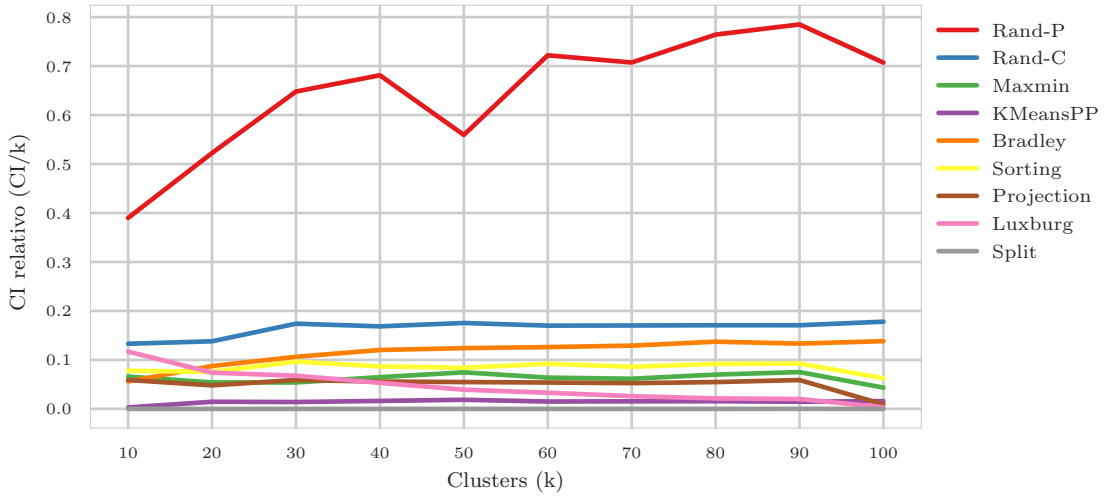

Figura 5: Dependência do valor relativo de CI (CI/k) e o número de *clusters* ao utilizar os subconjuntos do Birch2 (B2-sub).

Destaque positivo ao Split que manteve seus resultados, independente do número de *clusters*. No lado negativo, o Rand-P deteriorou seus resultados significativamente ao elevar a quantidade de *clusters*. Comparando os resultados podemos observar as mesmas tendências apontadas no artigo em relação à piora de desempenho correlacionada ao aumento do número de *clusters*.

Para uma melhor visualização, a figura 16 do artigo foi separada em 4, resultando nas imagens: 6, 7, 8 e 9.

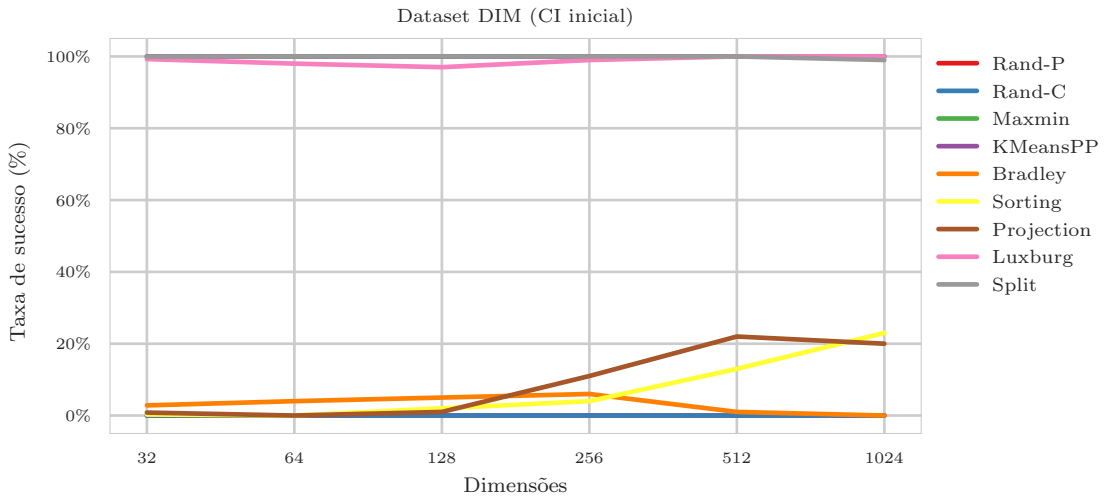

Figura 6: Dependência da taxa de sucesso (CI inicial) nas dimensões quando não há sobreposição de *clusters* (conjuntos DIM).

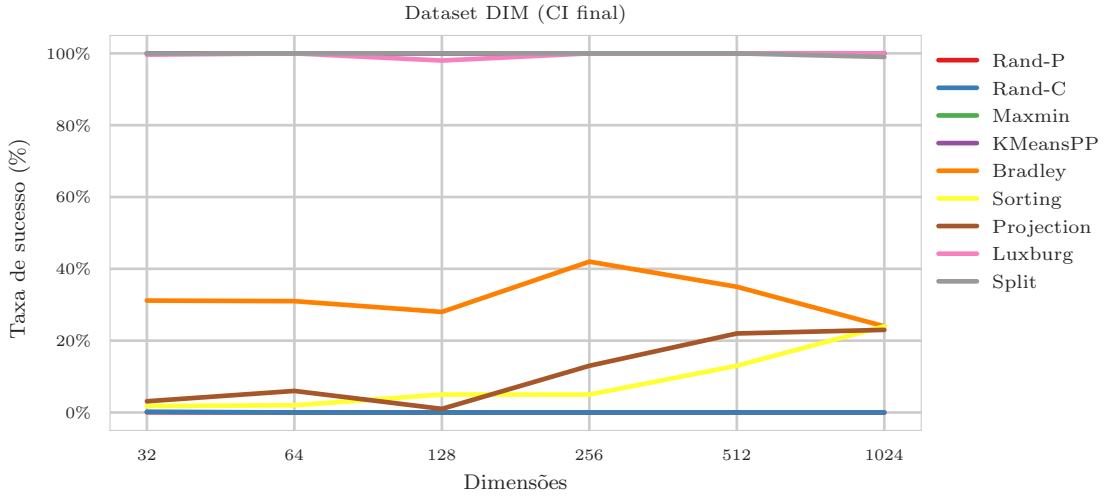

Figura 7: Dependência da taxa de sucesso (CI final) nas dimensões quando não há sobreposição de *clusters* (conjuntos DIM).

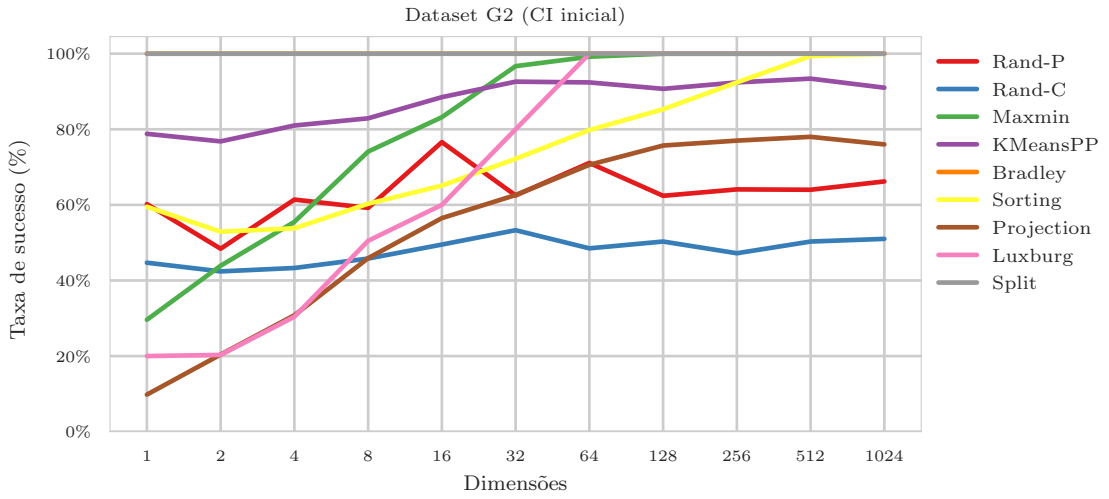

Figura 8: Dependência da taxa de sucesso (CI inicial) nas dimensões quando há sobreposição de *clusters* (conjuntos G2).

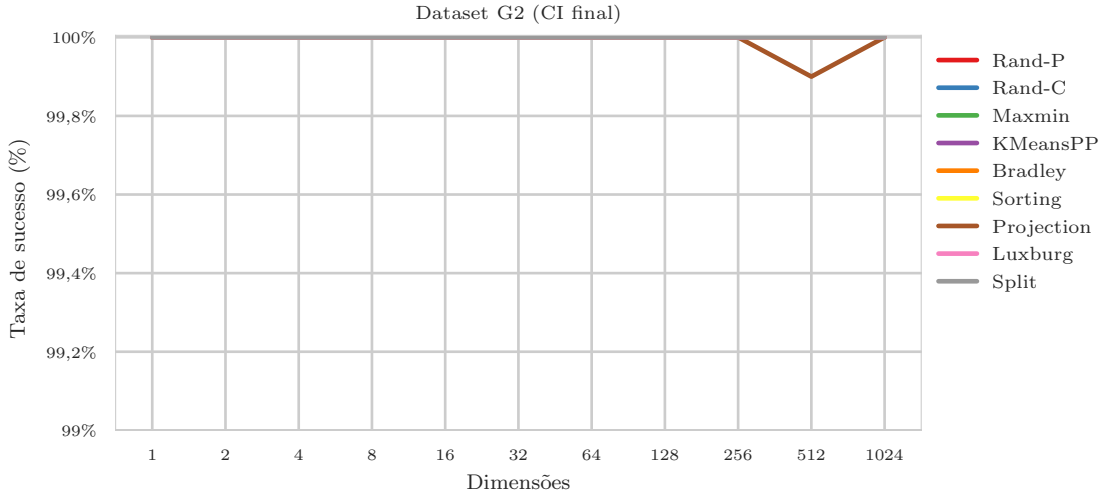

Figura 9: Dependência da taxa de sucesso (CI final) nas dimensões quando há sobreposição de *clusters* (conjuntos G2).

Uma relevante diferença é que enquanto no artigo os métodos Sorting e Projection apresentaram uma relação linear entre a taxa de sucesso e o número de dimensões, neste, não ocorreu o mesmo comportamento, apesar de existir um progresso nos resultados em dimensões mais altas.

Ao contrário de muitos outros algoritmos, a maldição da dimensionalidade parece

não afetar o k-Means. Em *datasets* que variam o número de dimensões, como no caso do DIM e G2, observou-se que os resultados não se degradaram. Outro relato pertinente é a ótima performance apresentada nas figuras 8 e 9, reafirmando o impacto benéfico da sobreposição à clusterização.

Analisando o desbalanço entre *clusters* na Tabela 6, uma fraqueza do k-Means fica perceptível. O *improvement* médio para a base Unbalance foi de 16%. Praticamente não há avanços após as iterações uma vez que a inicialização foi feita, ou seja, é ela que define a qualidade do algoritmo.

Finalmente, podemos notar que as repetições do algoritmo são totalmente positivas à clusterização, apesar do aumento no processamento de tempo. As tabelas 7 e 8 evidenciam isso.

	A3		Unbalance	
Método	CI mínimo	Iteração	CI mínimo	Iteração
Rand-P	7	11	0	0
Rand-C	1	4	0	2
Maxmin	0	1	0	0
kmeans++	0	0	0	0
Bradley	0	2	0	0
Sorting	1	2	2	3
Projection	0	3	2	3
Luxburg	0	0	0	1
Split	0	0	4	4

Tabela 7: CI mínimo atingido em 200 iterações no RKM (R = 100) para cada método nos *datasets* A3 e Unbalance.

CI-values													
Method	**s1**	**s2**	**s3**	**s4**	**a1**	**a2**	**a3**	**unb**	**b1**	**b2**	**dim32**	**KM**	**RKM**
Rand-P	2.0	0.1	0.1	0.0	3.5	7.4	11.2	0.7	8.5	64.1	1.3	**12.1**	**9.0**
Rand-C	0.1	0.0	0.0	0.0	0.5	1.7	3.0	2.3	3.2	11.5	0.8	**4.6**	**2.1**
Maxmin	0.0	0.0	0.0	0.0	0.0	0.4	0.6	0.0	3.1	4.3	0.0	**2.3**	**0.8**
kmeans++	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.8	0.0	0.0	**0.9**	**0.1**
Bradley	0.0	0.0	0.0	0.0	0.0	1.3	2.5	0.0	2.8	9.4	0.0	**3.4**	**1.5**
Sorting	0.0	0.0	0.0	0.0	0.0	1.1	2.3	3.4	2.5	3.4	0.2	**3.4**	**1.2**
Projection	0.0	0.0	0.0	0.0	0.0	1.0	2.1	3.3	2.4	2.7	0.0	**3.0**	**1.0**
Luxburg	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.1	0.0	0.0	0.0	**1.0**	**0.1**
Split	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.9	0.0	0.0	**0.6**	**0.4**
Success-%													
Method	**s1**	**s2**	**s3**	**s4**	**a1**	**a2**	**a3**	**unb**	**b1**	**b2**	**dim32**	**Aver.**	**Fails**
Rand-P	0%	93%	93%	100%	0%	0%	0%	27%	0%	0%	0%	**28%**	**7**
Rand-C	93%	100%	100%	100%	50%	0%	0%	0%	0%	0%	20%	**42%**	**5**
Maxmin	100%	100%	100%	100%	100%	57%	40%	100%	0%	0%	100%	**72%**	**2**
kmeans++	100%	100%	100%	100%	100%	100%	100%	100%	20%	97%	100%	**92%**	**0**
Bradley	100%	100%	100%	100%	97%	3%	0%	97%	0%	0%	100%	**63%**	**3**
Sorting	100%	100%	100%	100%	100%	0%	0%	0%	0%	0%	80%	**53%**	**5**
Projection	100%	100%	100%	100%	100%	7%	0%	0%	0%	0%	100%	**55%**	**4**
Luxburg	100%	100%	100%	100%	100%	100%	100%	27%	100%	100%	100%	**93%**	**0**
Split	100%	100%	100%	100%	100%	100%	100%	0%	10%	100%	100%	**83%**	**1**
Running time (s)													
Method	**s1**	**s2**	**s3**	**s4**	**a1**	**a2**	**a3**	**unb**	**b1**	**b2**	**dim32**		
Rand-P	2.0	2.4	3.0	1.7	1.3	5.1	6.9	1.5	180.2	100.5	0.7		
Rand-C	1.1	1.6	3.3	1.4	0.8	2.3	3.5	1.2	73.8	16.4	0.8		
Maxmin	9.4	12.5	15.4	12.3	1.1	15.6	16.3	7.2	124.6	47.0	3.9		
kmeans++	7.5	10.4	12.4	11.4	1.5	12.9	12.9	5.1	166.8	110.0	5.2		
Bradley	8.1	11.7	11.8	10.2	8.9	12.9	16.2	8.2	397.7	39.9	8.9		
Sorting	10.8	12.0	14.1	11.4	1.5	13.0	13.0	7.2	141.0	19.8	5.1		
Projection	7.0	8.9	9.1	6.6	5.5	8.0	12.2	8.5	166.5	115.3	2.2		
Luxburg	4.3	5.8	6.7	5.1	4.8	8.3	18.7	1.6	1459.3	269.6	1.7		
Split	8.4	12.6	10.3	10.8	14.0	23.1	35.4	4.8	102.2	65.5	7.3		

Tabela 8: Desempenho do k-Means com repetição (100). As últimas duas colunas mostram a média dos resultados de todos os bancos de dados com o algoritmo sem repetição (KM) e com repetição (RKM).

5 Conclusões

De uma maneira geral, os resultados foram similares aos relatados pelo artigo, indicando uma implementação correta do código dos experimentos, E também corroborase a tese das repetições serem uma ótima prática para atingir uma boa clusterização.

Uma diferença fundamental foi o ótimo desempenho do kmeans++, ficando próximo dos algoritmos de inicialização mais complexa, como no caso do Luxburg e Split. Também é possível apontar diferenças na análise de CI Final com pouca sobreposição e de CI Final para os G2 na análise de dimensionalidade.

Fica claro o quão bom pode vir a ser o desempenho do k-Means quando uma inicialização razoável é feita. Ela, juntamente com repetições nas execuções, podem compensar significativamente a principal fraqueza do algoritmo. Pode-se dizer, ainda, que a melhor condição para o funcionamento do método é quando há uma sobreposição de *clusters*, onde a maioria das inicializações tem, em média, desempenho melhor. E a pior é a presença de um desbalanço entre o tamanho deles.

Ao tentar responder as questões propostas no começo do artigo, nesta replicação chegaríamos às seguintes conclusões:

- **A que extensão o k-Means pode ser melhorado por uma técnica inicialização melhor?** Em geral, a técnica de inicialização faz diferença, comparando o desempenho da Random Projection e kmeans++, por exemplo. Mas, realizar repetição, mesmo que com uma técnica simples como Maxmin, é claramente mais importante que escolher a técnica de inicialização.
- **A fraqueza fundamental do algoritmo pode ser eliminada simplesmente por repetições consecutivas?** Totalmente, não. Mas combinada com uma técnica de inicialização como kmeans++ ou Luxburg só enfrenta problemas com *datasets* desbalanceados ou com muitos *clusters* como o B1.
- **Pode-se predizer sob quais condições o k-Means funciona e em quais ele falha?** Com as informações obtidas neste trabalho, entendemos que utilizando repetições e um método como kmeans++, Luxburg ou Split, o algoritmo se torna bem robusto. Mas, como mencionado, é preciso ter cuidado com *datasets* desbalanceados e com muitos *clusters*.

Bibliografia

[1] Pasi Fränti, Sami Sieranoja (2019). How much can k-means be improved by using better initialization and repeats?. Pattern Recognition, 93, 95–112.

[2] U.V. Luxburg (2010). Clustering stability: an overview. Found. Trends Mach. Learn., 2(3), 235–274.

[3] Pasi Fränti, Sami Sieranoja (2018). K-means properties on six clustering benchmark datasets. Applied Intelligence, 48(12), 4743–4759.