Clustering Using REpresentatives: An Efficient Clustering Algorithm for Large Databases

Bilzã Marques de Araújo
bmarques@icmc.usp.br

SCC5895 - Análise de Agrupamento de Dados

Seminário
09 de Dezembro de 2010

Conteúdo

Principais Referências

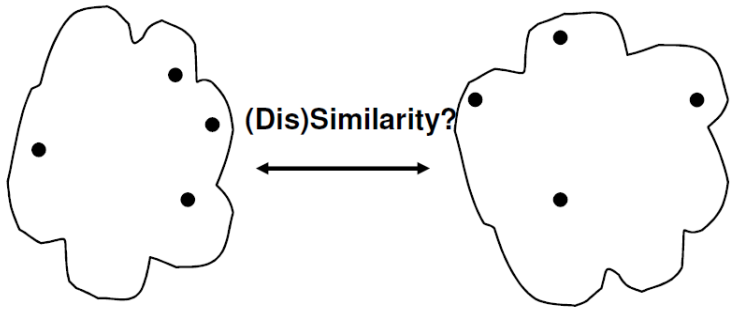

Guha, S., Rastogi, R., & Shim, K. (1998).
Cure: an efficient clustering algorithm for large databases.
In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, SIGMOD '98 (pp. 73–84). New York, NY, USA: ACM.


Guha, S., Rastogi, R., & Shim, K. (2001).
Cure: an efficient clustering algorithm for large databases.
Information Systems, 26, 35–58.


Theodoridis, S. & Koutroumbas, K. (2009).
Pattern Recognition, (pp. 683–685).
Academic Press, 4th edition.

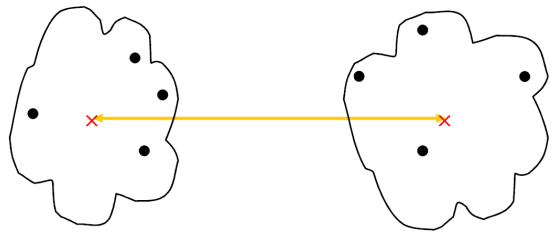
Introdução

Problema de *clustering*: Dado um conjunto de objetos, agrupar os objetos em *clusters* tal que cada objetos em um cluster seja mais similares aos demais objetos no mesmo *cluster* que a objetos em outros *clusters*.

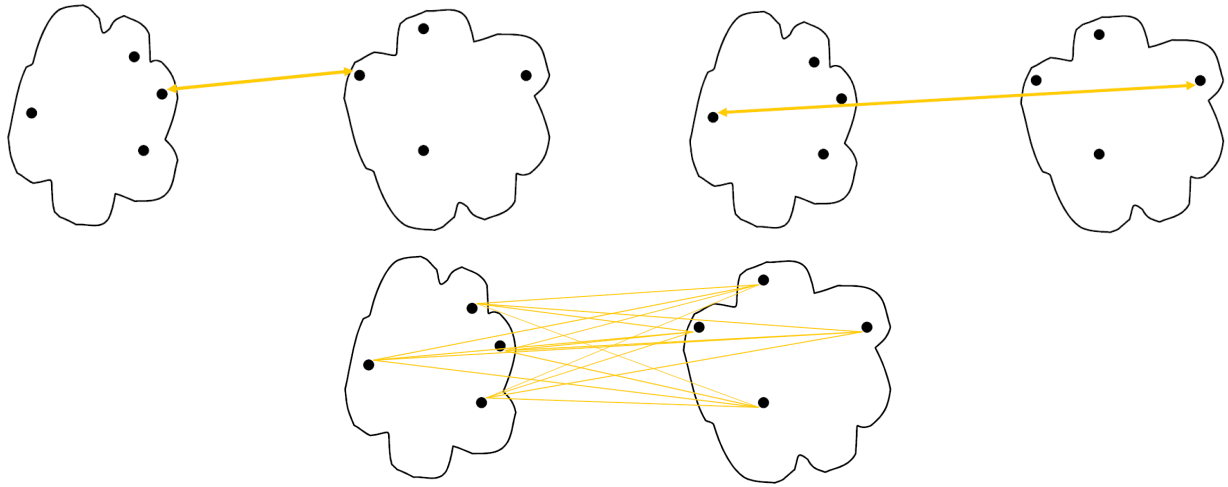


Representantes

- Quando unem grupos, algoritmos aglomerativos utilizam "representantes":
  - centroides, medoides: d_{mean}

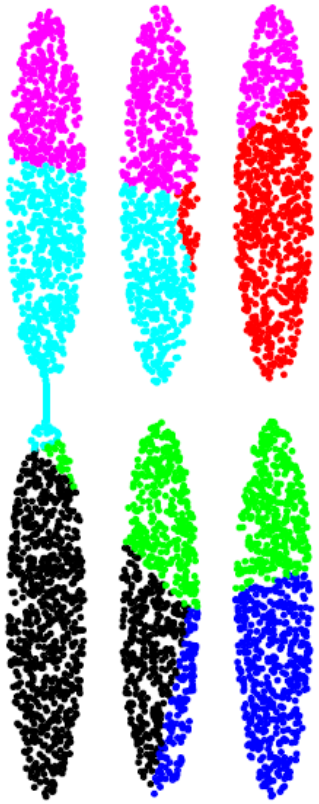


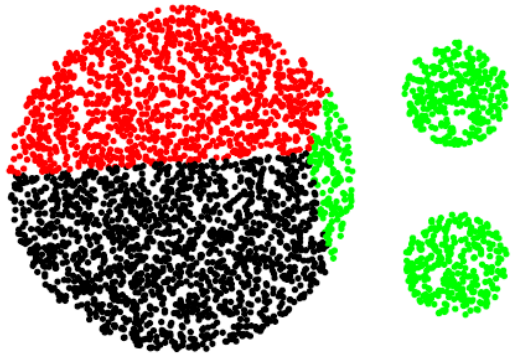
  - todos os objetos do grupo: d_{min}, d_{max}, d_{ave}

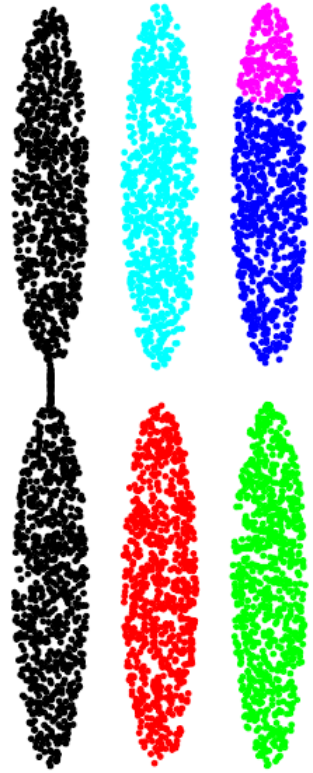


Deficiências abordagens clássicas

- Abordagens clássicas favorecem *clusters* esféricos e tamanhos similares ou são frágeis na presença de *outliers*







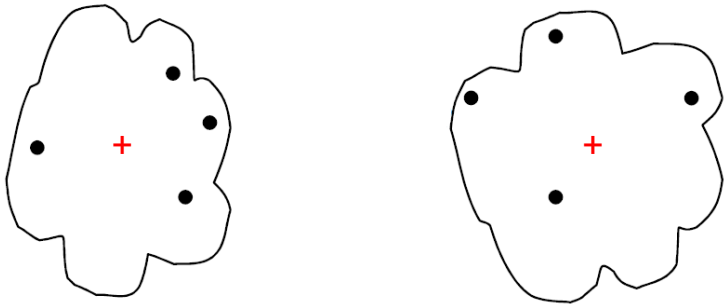
Clustering Using REpresentatives

- Técnica de Agrupamento de Dados Hierárquica
- Cada grupo é representado por $\min(c, |C_i|)$ pontos que descrevem a forma* do grupo
- Dois grupos, C_1 e C_2, são unidos se $\forall C_i, C_j$

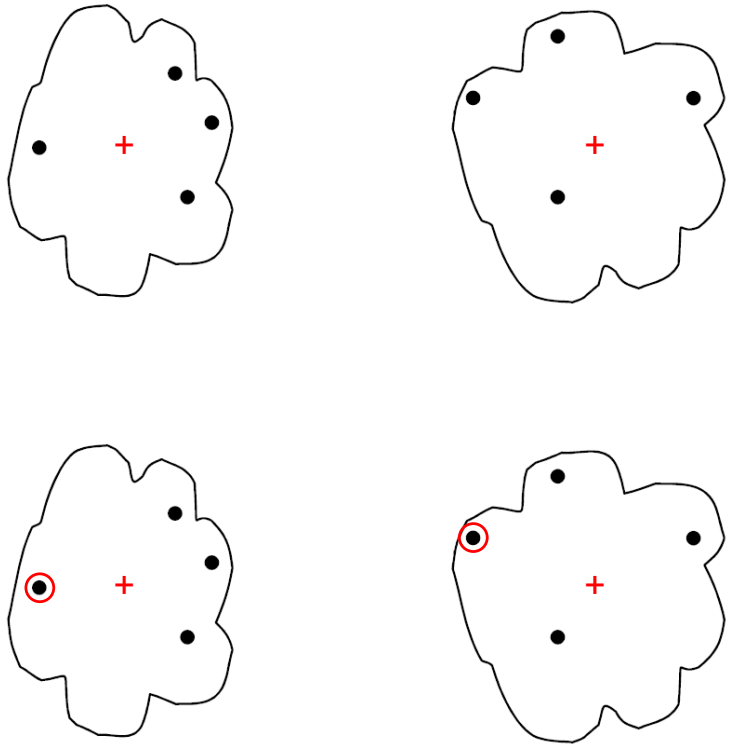
$$d_{cure}(C_1, C_2) == min(d_{cure}(C_i, C_j))$$

- $d_{cure}(C_i, C_j)$ é a menor distancia entre representantes de C_i e C_j
- Os grupos são unidos até que sejam obtidos k grupos

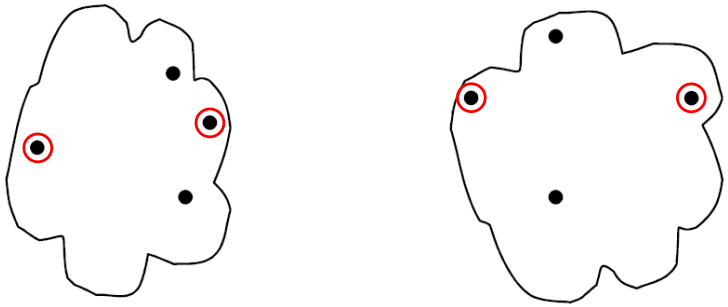
c objetos bem distribuídos



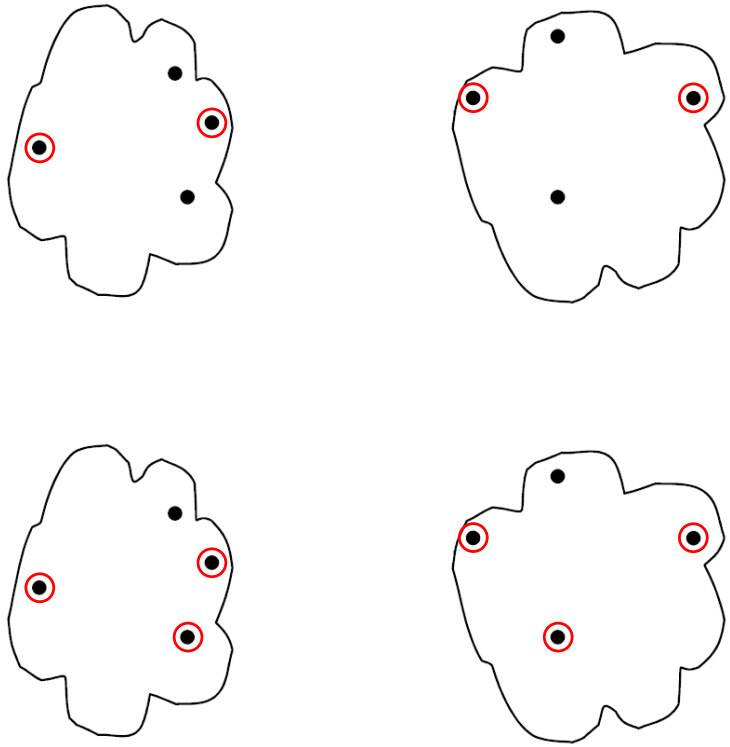
c objetos bem distribuídos



c objetos bem distribuídos



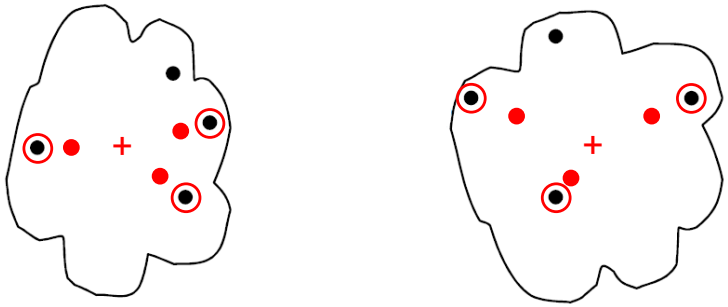
c objetos bem distribuídos



c representantes

- Cada representante é encolhidos sentido ao centroide em α

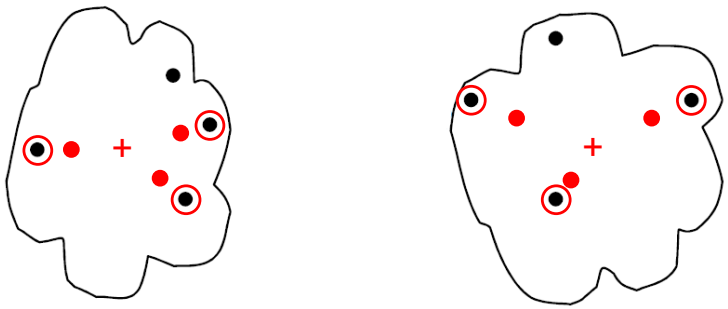
$$p = p + \alpha(\overline{x} - p)$$



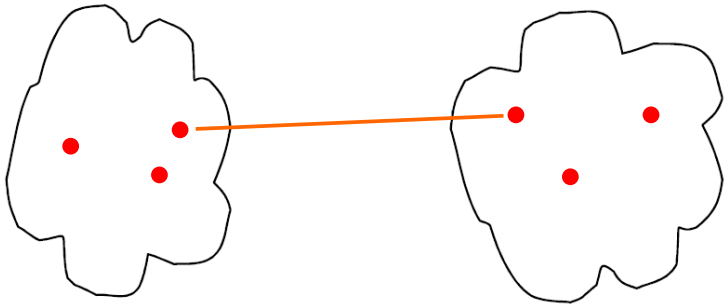
c representantes

- Cada representante é encolhidos sentido ao centroide em α

$$p = p + \alpha(\overline{x} - p)$$



- Dois grupos são unidos de acordo com os representantes



Algoritmo

```
procedure cluster(S, k)
begin
1.  T := build_kd_tree(S)
2.  Q := build_heap(S)
3.  while size(Q) > k do {
4.     u := extract_min(Q)
5.     v := u.closest
6.     delete(Q, v)
7.     w := merge(u, v)
8.     delete_rep(T, u); delete_rep(T, v); insert_rep(T, w)
9.     w.closest := x /* x is an arbitrary cluster in Q */
10.    for each x ∈ Q do {
11.       if dist(w, x) < dist(w, w.closest)
12.          w.closest := x
13.       if x.closest is either u or v {
14.          if dist(x, x.closest) < dist(x, w)
15.             x.closest := closest_cluster(T, x, dist(x, w))
16.          else
17.             x.closest := w
18.          relocate(Q, x)
19.       }
20.       else if dist(x, x.closest) > dist(x, w) {
21.          x.closest := w
22.          relocate(Q, x)
23.       }
24.    }
25.    insert(Q, w)
26. }
end
```

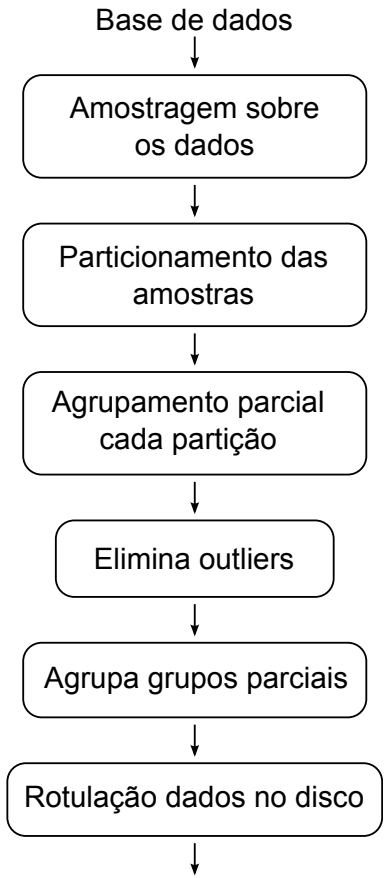
Algoritmo

```
procedure cluster(S, k)
begin
1.  T := build_kd_tree(S)
2.  Q := build_heap(S)
3.  while size(Q) > k do {
4.     u := extract_min(Q)
5.     v := u.closest
6.     delete(Q, v)
7.     w := merge(u, v)
8.     delete_rep(T, u); delete_rep(T, v); insert_rep(T, w)
9.     w.closest := x /* x is an arbitrary cluster in Q */
10.    for each x ∈ Q do {
11.       if dist(w, x) < dist(w, w.closest)
12.          w.closest := x
13.       if x.closest is either u or v {
14.          if dist(x, x.closest) < dist(x, w)
15.             x.closest := closest_cluster(T, x, dist(x, w))
16.          else
17.             x.closest := w
18.          relocate(Q, x)
19.       }
20.       else if dist(x, x.closest) > dist(x, w) {
21.          x.closest := w
22.          relocate(Q, x)
23.       }
24.    }
25.    insert(Q, w)
26. }
end
```

```
procedure merge(u, v)
begin
1.  w := u ∪ v
2.  w.mean := (|u|u.mean + |v|v.mean) / (|u| + |v|)
3.  tmpSet := ∅
4.  for i := 1 to c do {
5.     maxDist := 0
6.     foreach point p in cluster w do {
7.        if i = 1
8.           minDist := dist(p, w.mean)
9.        else
10.          minDist := min{dist(p, q) : q ∈ tmpSet}
11.       if (minDist ≥ maxDist){
12.          maxDist := minDist
13.          maxPoint := p
14.       }
15.    }
16.    tmpSet := tmpSet ∪ {maxPoint}
17. }
18. foreach point p in tmpSet do
19.    w.rep := w.rep ∪ {p + α*(w.mean-p) }
20. return w
end
```

Melhorias - Large Scale Databases

- Bases de dados com centenas de milhares de objetos
- Limitações de memória
- Inviabilidade Computacional $O(n^2 \log n)$



Amostragem

- Amostra suficientemente grande preserva características dos grupos e elimina *outliers*
- De acordo com *Chernoff bounds*, para *clusters* bem definidos (densos intra e esparso inter):

$$s_{min} = \xi k\rho + k\rho \log(\frac{1}{\delta}) + k\rho\sqrt{(\log(\frac{1}{\delta}))^2 + 2\xi \log(\frac{1}{\delta})}$$

- Não depende do número de objetos n, mas sim do número de *clusters* bem definidos, k
- Com grupos de densidades variadas é necessário assumir k de acordo com regiões densas

Particionamento

- Particionamento da amostra
- Agrupamento hierárquico até $\frac{n}{pq}$
- Permite melhor tratamento de *outliers**
- Reduz ainda mais custo computacional

$$a^2 + b^2 < (a + b)^2$$

Remoção de *outliers*

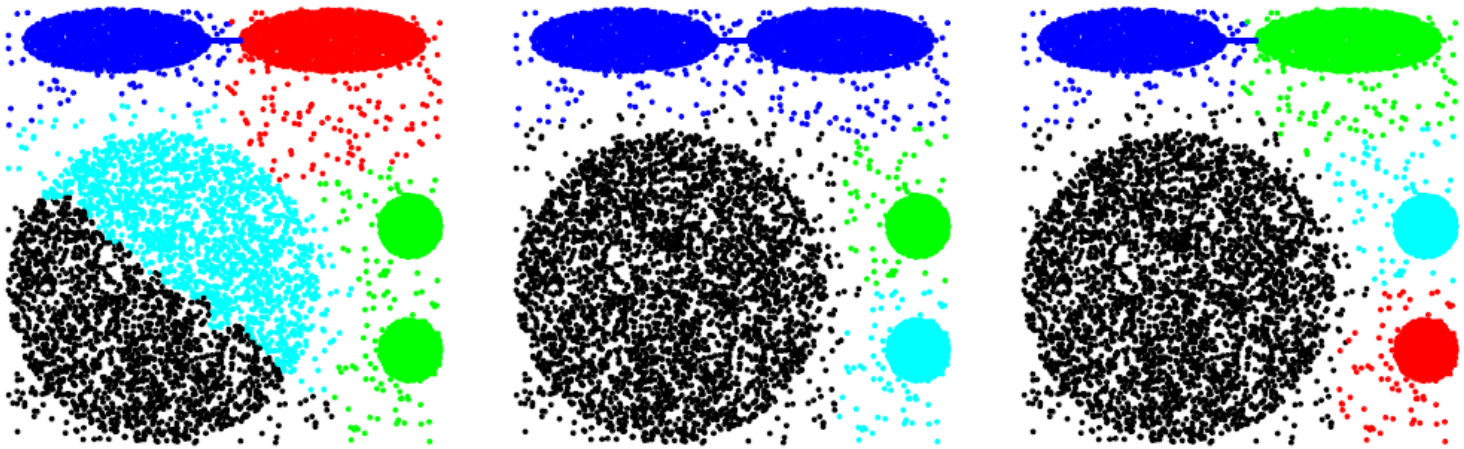
- Modelo ainda susceptível a outliers
- Amostragem e parâmetro α podem não eliminar todo efeito de
- Uma vez que *outliers* tendem a ser agrupados no final da hierarquia
- Grupos com poucos objetos são removidos da amostra
  - ao final de cada partição
  - quando atinge o número de grupos k

Rotulação em disco

- Objetos em disco são rotulados de acordo com os representantes de grupos
- Cada objeto é atribuído ao grupo do representante mais próximo

Comparação com BIRCH e MST

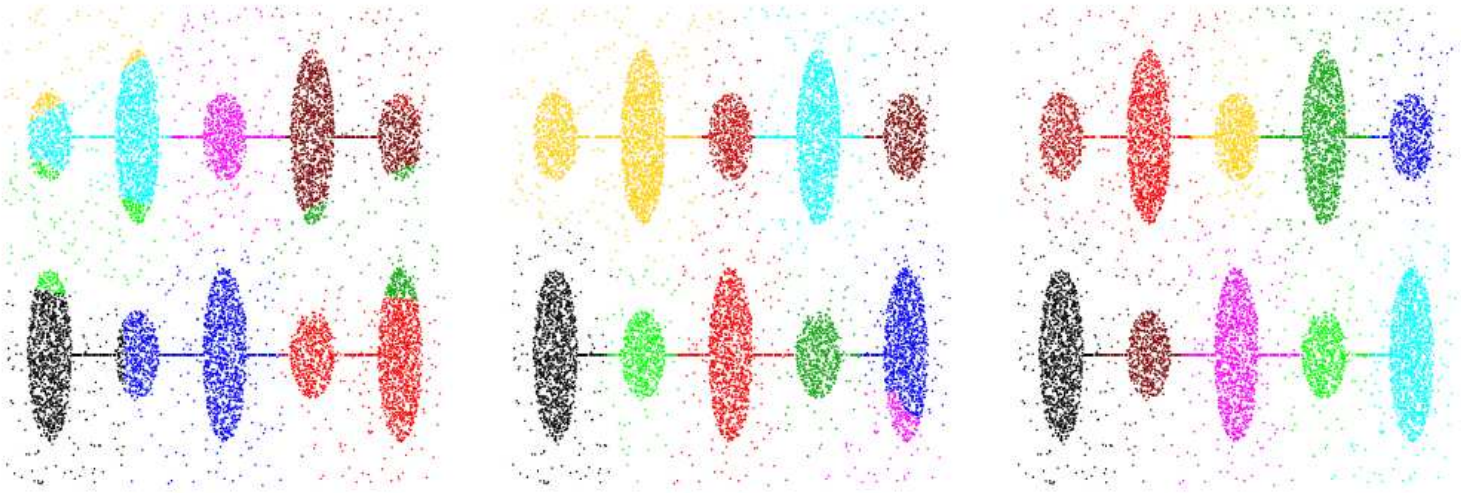
- $n = 100000,\ s = 2500,\ c = 10,\ \alpha = 0.3,\ p = 1$



(a) BIRCH (b) MST METHOD (c) CURE

Comparação com BIRCH e MST

- $n = 100000,\ s = 2500,\ c = 10,\ \alpha = 0.3,\ p = 1$

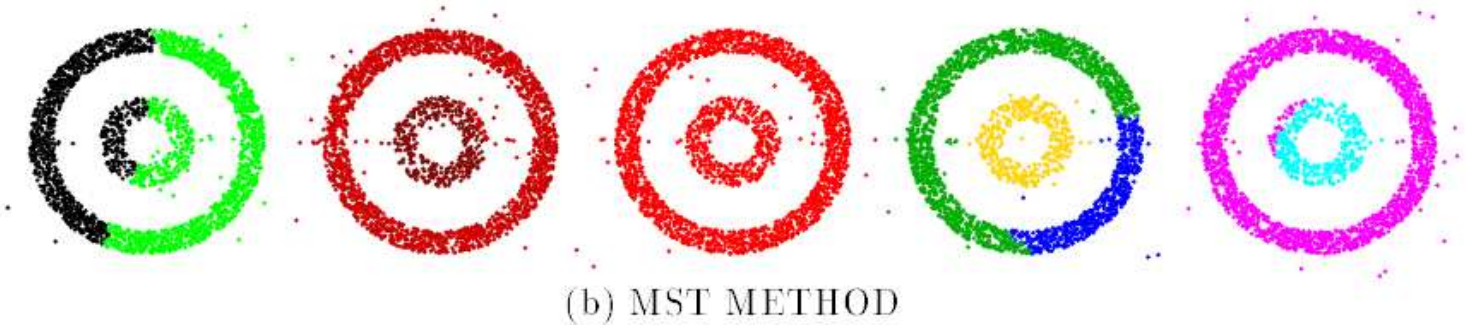


(d) BIRCH (e) MST METHOD (f) CURE

Comparação com BIRCH e MST

- $n = 121560,\ s = 3000,\ c = 100,\ \alpha = 0.15,\ p = 1$

(a) BIRCH

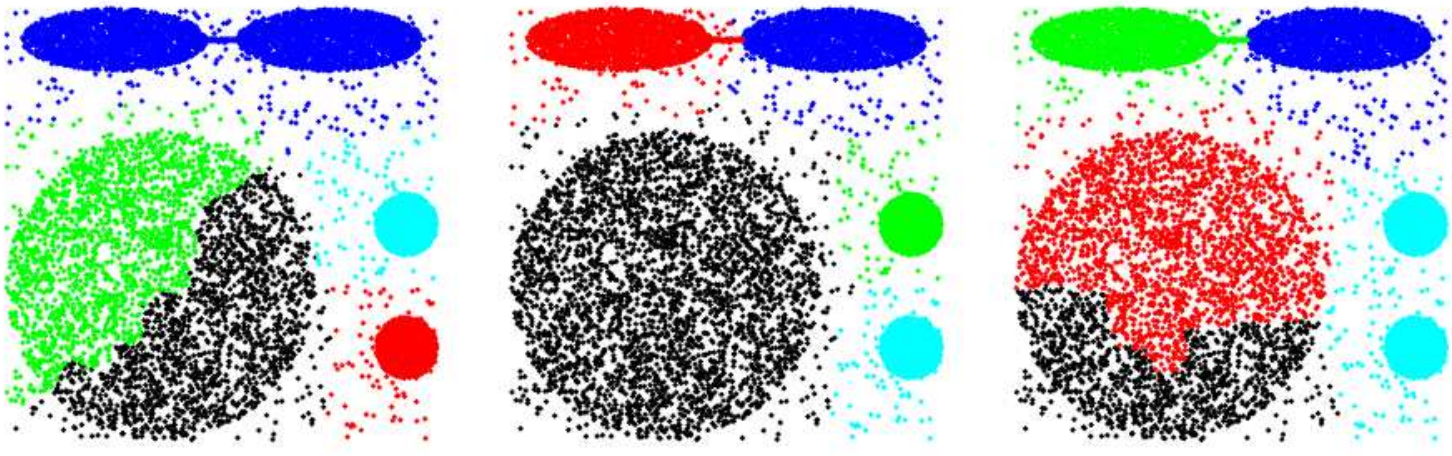


(b) MST METHOD

(c) CURE

Fator de encolhimento α

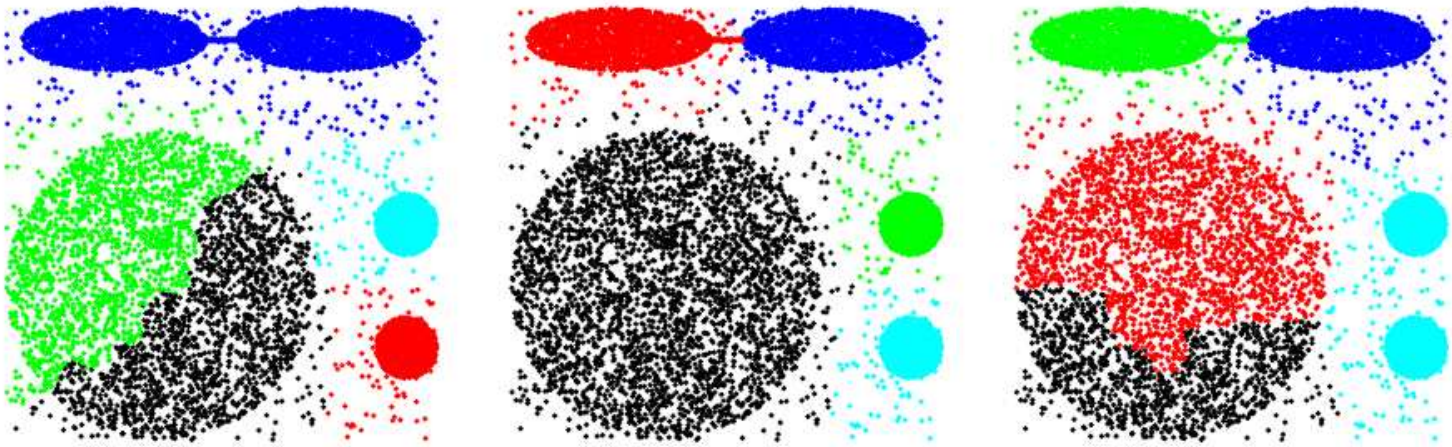
- $n = 100000,\ s = 2500,\ c = 10,\ p = 1,\ \alpha = [0.1, 0.9]$



(a) $\alpha = 0.1$ (b) $\alpha = 0.2 - 0.7$ (c) $\alpha = 0.8 - 0.9$

Fator de encolhimento α

- $n = 100000,\ s = 2500,\ c = 10,\ p = 1,\ \alpha = [0.1, 0.9]$

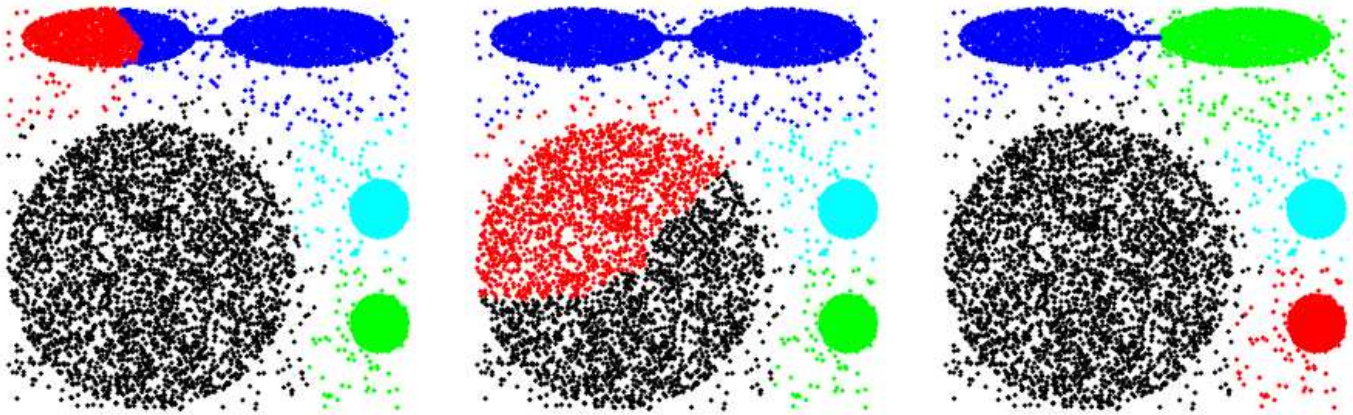


(a) $\alpha = 0.1$ (b) $\alpha = 0.2 - 0.7$ (c) $\alpha = 0.8 - 0.9$

- Quando $\alpha = 0$ similar ao MST, quando $\alpha = 1$, similar ao BIRCH

Número de representantes, c

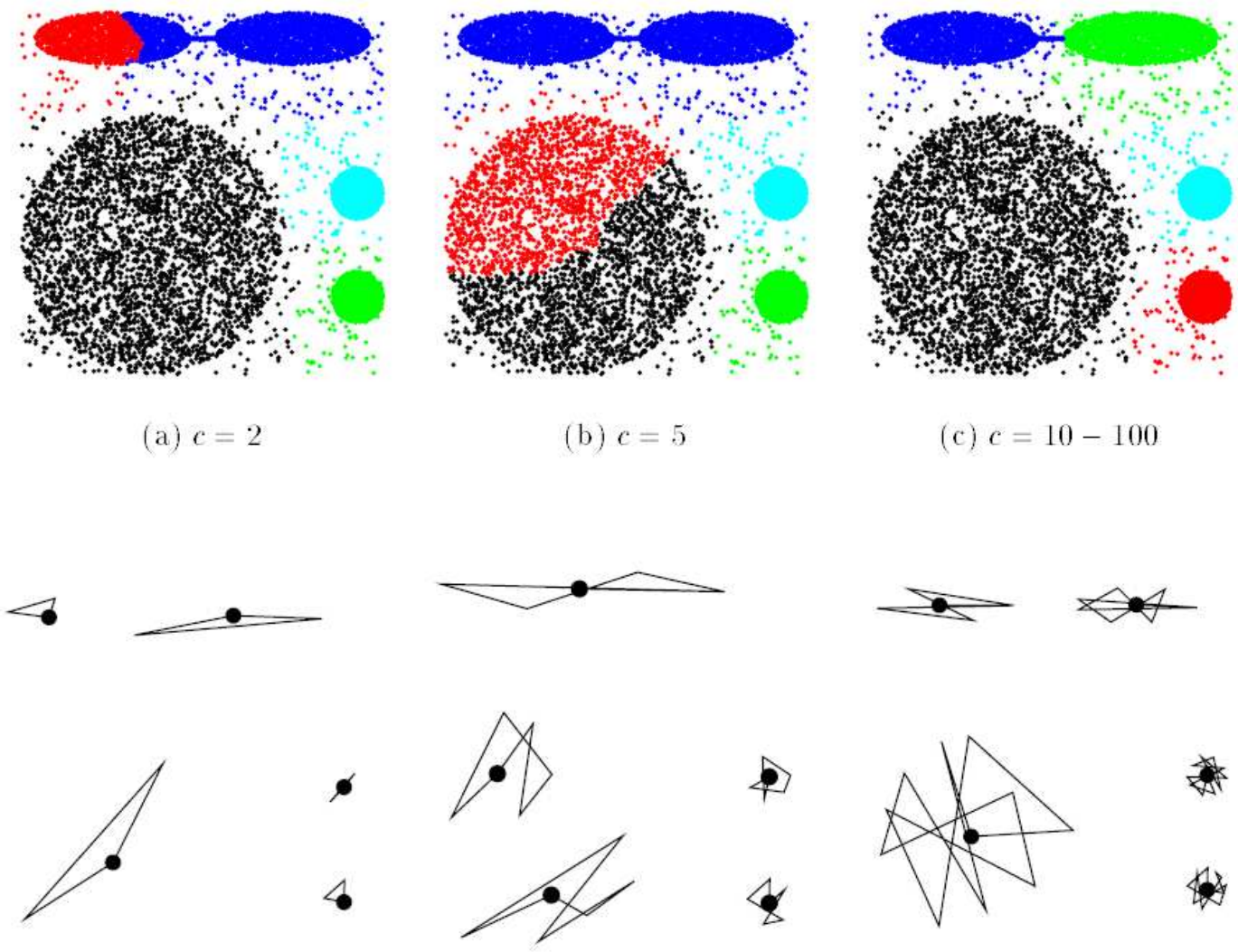
- $n = 100000,\ s = 2500,\ \alpha = 0.3,\ p = 1,\ c = [1, 100]$



(a) $c = 2$ (b) $c = 5$ (c) $c = 10 - 100$

Número de representantes, c

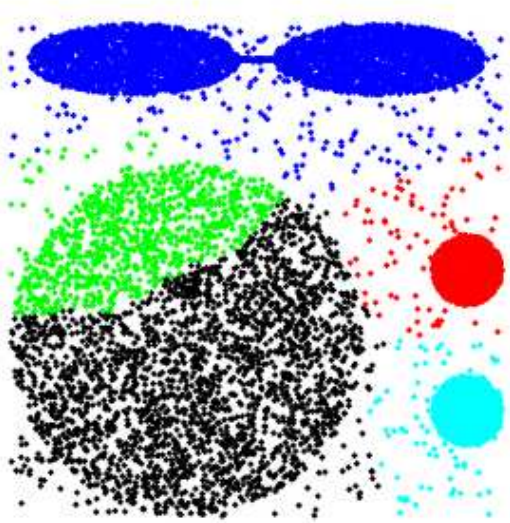
- $n = 100000,\ s = 2500,\ \alpha = 0.3,\ p = 1,\ c = [1, 100]$



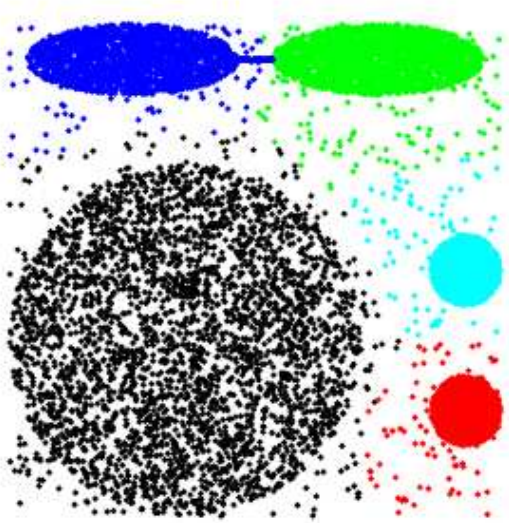
(a) $c = 2$ (b) $c = 5$ (c) $c = 10 - 100$

Tamanho da amostra de representantes, s

- $n = 100000,\ \alpha = 0.3,\ c = 10,\ p = 1,\ s = [500, 5000]$



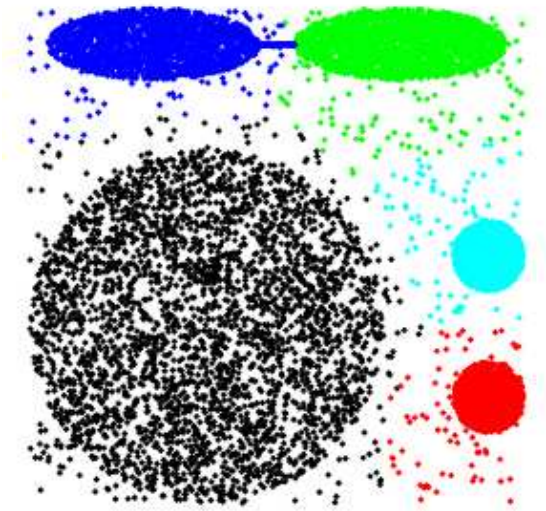
(a) $s = 2000$



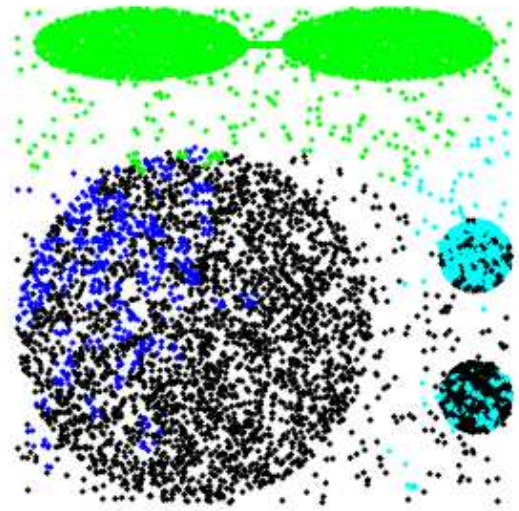
(b) $s = 2500$

Número de partições, p

- $n = 100000,\ \alpha = 0.3,\ c = 10,\ s = 2500,\ p = [1, 100]$,

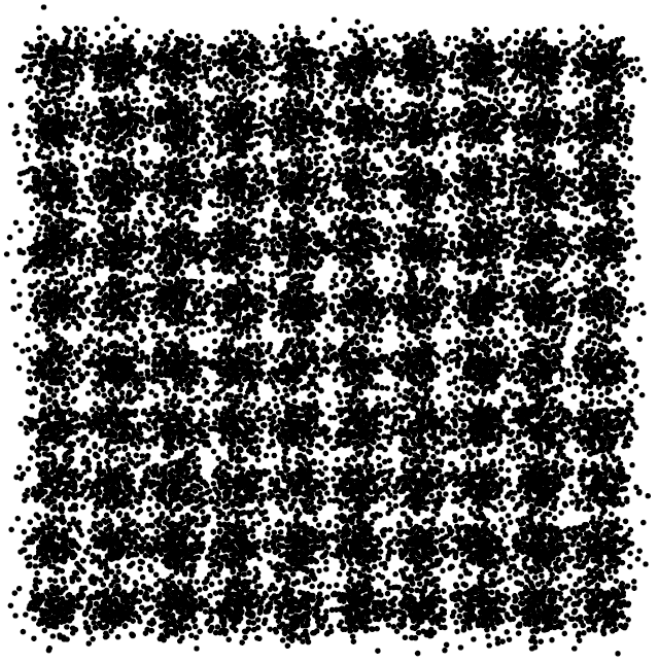


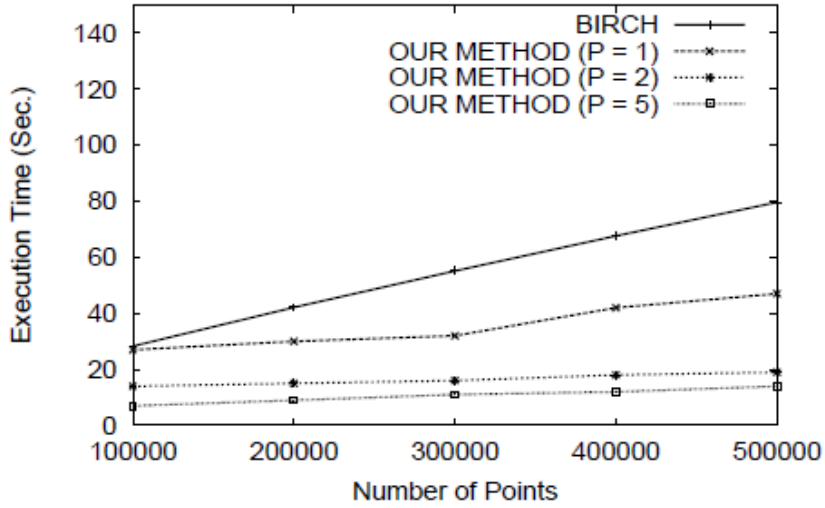
(a) $p = 1 - 50$



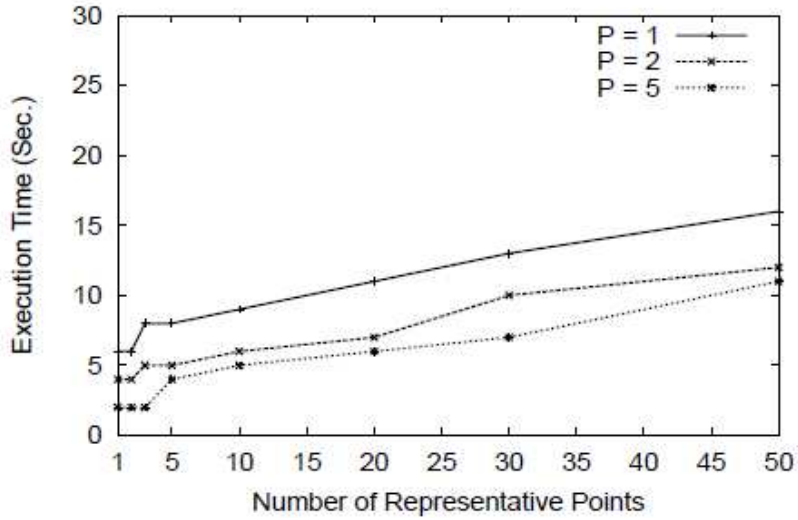
(b) $p = 100$

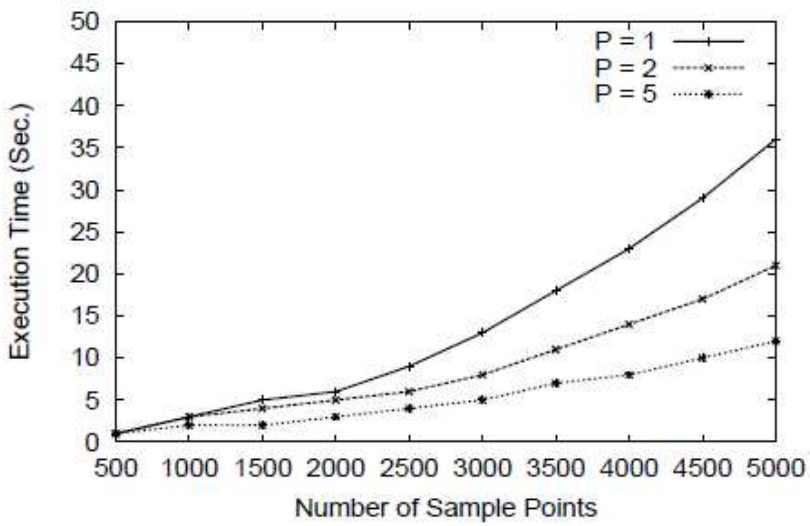
Performance comparado ao BIRCH

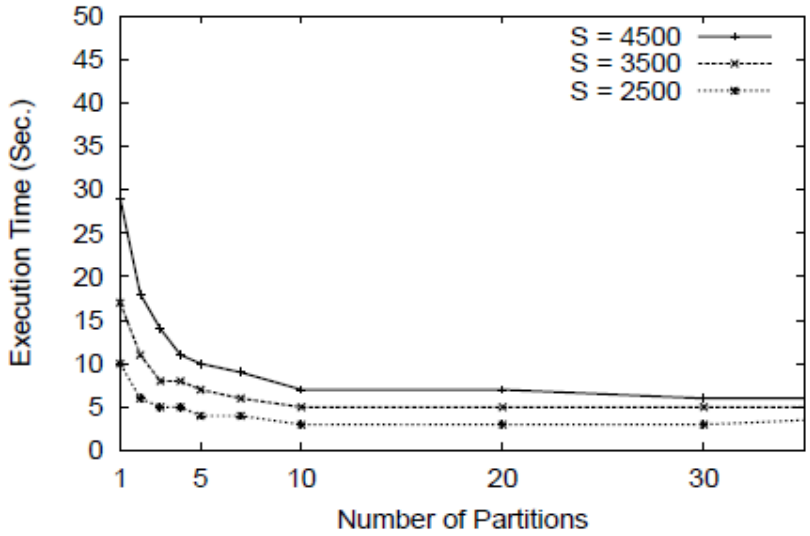




Performance







Obrigado pela atenção!

Bibliografia I


Guha, S., Rastogi, R., & Shim, K. (1998).
Cure: an efficient clustering algorithm for large databases.
In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, SIGMOD '98 (pp. 73–84). New York, NY, USA: ACM.


Guha, S., Rastogi, R., & Shim, K. (2001).
Cure: an efficient clustering algorithm for large databases.
Information Systems, 26, 35–58.


Theodoridis, S. & Koutroumbas, K. (2009).
Pattern Recognition, (pp. 683–685).
Academic Press, 4th edition.