Relatório Ant Clustering

Matias Giuliano Gutierrez Benitez[1], **Vinícius Takeo Friedrich Kuwaki**[1]

[1]Departamento de Ciência da Computação – Universidade do Estado de Santa Catarina (UDESC)
Joinville – SC – Brazil

matiguti17@gmail.com,vtkwki@gmail.com

1. Introdução

A Ciência da Computação ao observar o comportamento de diversos fenômenos da natureza, têm se inspirado nela para modelar e solucionar os mais diversos tipos de problemas do mundo real. Um desses fenômenos que tem inspirado os cientistas a décadas em seus algoritmos são as formigas. As formigas, em sua natureza se organizam em uma sociedade, onde cada qual divide suas tarefas em prol do formigueiro. A inteligência de enxame, ou também conhecida como *Swarm Intelligence* (SI) é um paradigma de inteligência artificial baseado no estudo do comportamento emergente em sistemas auto-organizados e descentralizados que buscam imitar esses comportamentos e aplicá-los na solução de problemas computacionais difíceis [Handl and Meyer 2007]. Usando a abordagem de SI, é possível construir algoritmos de clusterização de dados: um algoritmo não-supervisionado que busca agrupar dados de acordo com suas similaridades.

Esse trabalho busca reportar a implementação de algoritmos de clusterização usando inteligência de enxame baseado no comportamento de formigas para a clusterização de dados homogêneos e heterogêneos. Na seção seguinte serão descritos a metodologia e o desenvolvimento, bem como as estratégias utilizadas e as justificativas em torno da escolha dessas estratégias. Em seguida serão descritos os experimentos realizados e os resultados obtidos com os experimentos. Após, os resultados serão analisados e considerações e críticas em cima deles serão efetuados, apresentado conclusões finais e possibilidades de futuras pesquisas em seguida.N

2. Metodologia de Desenvolvimento

A ideia principal do algoritmo de *clustering* de dados é agrupar os dados a partir de probabilidades de "pegar"(p_p) ou "largar"(p_d) os dados. Para tal, o Pseudocódigo 1 foi utilizado implementando as ideias propostas em [Gao 2016] e [Lumer and Faieta 1994].

Utilizou-se a linguagem de programação Python [Van Rossum and Drake Jr 1995] para a implementação do pseudocódigo, bem como a biblioteca PyGame[1] para a visualização dos resultados. Duas bases de dados foram utilizadas nos testes, uma contendo 4 classes de dados e outra 15 classes de dados, bem como dados sem classes.

2.1. Dados homogêneos

Para os dados sem classes, calculou-se a função f (Equação 1) que dá a probabilidade do dado em conjunto com seus vizinhos. Tal probabilidade é utilizada para calcular as probabilidades de se pegar um item (Equação 2) ou largar um item (Equação 3). Para dados homogêneos, i.e., dados sem classe, utilizou-se a função Sigmoide (Equação 4),

[1]https://www.pygame.org

Algorithm 1: Ant clustering

Para cada formiga F:
  Se F **não estiver carregando** um dado e **estiver em cima** de um dado:
    f $\leftarrow p_p$
    prob $\leftarrow$ random(0,0.5)
    Se f for maior prob:
      F $\leftarrow$ Pegue o dado;
    Fim Se
  Fim Se
  Se F **estiver carregando** um dado e **estiver em cima** de uma célula vazia:
    f $\leftarrow p_d$
    prob $\leftarrow$ random(0,0.5)
    Se f for maior prob:
      F $\leftarrow$ Largue o dado;
    Fim Se
  Fim Se
  Escolha uma célula vizinha em um raio de 1 sem uma formiga em cima e vá para ela;
Fim para

baseando-se na metodologia proposta em [Gao 2016]. A escolha da função Sigmoide se deu pelo fato da necessidade de escolha de apenas uma constante, a constante **c** (ver Equação 4).

$$f(i,j) = \frac{near(i,j)}{(2*r+1)^2 - 1} \tag{1}$$

$$P_p(i,j) = 1 - Sigmoid(f(i,j)) \tag{2}$$

$$P_d(i,j) = Sigmoid(f(i,j)) \tag{3}$$

$$Sigmoide(x) = \frac{1 - e^{-cx}}{1 + e^{-cx}} \tag{4}$$

A Equação 1 que descreve a probabilidade de um dado homogêneo, é baseada na quantidade de dados no entorno de uma célula analisada. A função retorna a divisão entre a quantidade de dados presentes nas células vizinhas com determinado raio e o número total de células vizinhas com determinado raio.

2.2. Dados heterogêneos

Já para os dados heterogêneos, utilizou-se das abordagens propostas em [Lumer and Faieta 1994], onde a probabilidade de pegar (Equação 6) ou largar (Equação 7) um item é calculado em função de duas constantes k1 e k2. A função f proposta (Equação 5), utiliza da distância euclidiana entre os dados e uma constante

α, esta calculada de acordo com o Pseudocódigo 2. A função f descrita na Equação 5 retorna um somatório calculado a partir dos dados vizinhos dado um raio r. Escolheu-se essa abordagem para os dados heterogêneos, visto que ao utilizar a abordagem baseada na Sigmoide, os resultados não atenderam as expectativas.

$$f(i,j) = MAX(0, \frac{\sum_j (1 - \frac{d(i,j)}{\alpha})}{s^2}) \tag{5}$$

$$P_p(i,j) = (\frac{k1}{k1 + f(i,j)})^2 \tag{6}$$

$$P_d(i,j) = \begin{cases} 2 * f(i,j) & \text{Se} f(i,j) < k2 \\ 0 & \text{Caso contrário} \end{cases} \tag{7}$$

Algorithm 2: Cálculo da constante alfa

```
1 soma ← 0
2 contador ← 0
3 Para cada dado X na grid:
4    Para cada dado Y diferente de X na grid:
5       soma ← distancia_euclidiana(X,Y);
6       contador ← contador + 1;
7    Fim para
8 Fim para
9 Retorne soma / contador;
```

2.3. Bases de dados

Para a realização dos experimentos, serão utilizados, como mencionado anteriormente, duas bases de dados heterogêneos e uma base de dados homogênea. A base de dados homogênea é gerada aleatoriamente em cada execução, sendo cada dado colocado em uma posição ij da *grid* (uma matriz 50x50).

As bases heterogêneas, por outro lado, são constituídas de uma coordenada x e uma coordenada y, bem como um rótulo z, do qual uma cor é atribuída para a sua representação visual. Na Figuras 1 é possível ver a distribuição espacial das bases de dados A (à esquerda) e B (à direita), contendo 400 e 600 dados, respectivamente.

Para tornar a proporção de células vazias em relação a quantidade de dados, utilizou-se uma matriz 50x50 para a base de dados A e uma matriz 61x61 para a base de dados B.

3. Experimentos

Para a condução dos experimentos, utilizou-se *grid's* com os dados dispostos de forma aleatória. Para tornar os experimentos próximos entre si (os homogêneos e heterogêneos) gerou-se 400 dados aleatórios (mesmo número da base de dados A) para os testes com dados homogêneos.

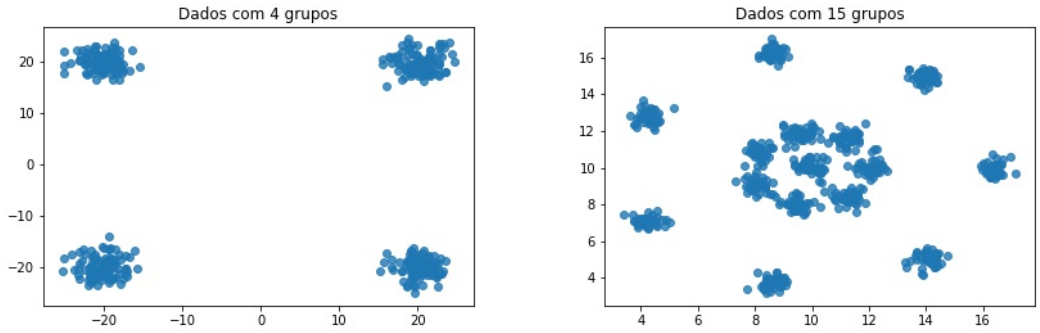

Figura 1. Distribuição espacial das bases de dados.

3.1. Dados homogêneos

Como a abordagem discutida na seção anterior para dados homogêneos, utilizou da função Sigmoide, era necessário ajustar a constante c para valores no intervalo (0,1). Para tal, foram testados os valores de $c \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. Também foram conduzidos experimentos com raios variando entre 1 e 5. As variações foram submetidas a 10.000 iterações (eras) ao algoritmo, cujos resultados podem ser vistos nas Figuras 2, 3 e 4.

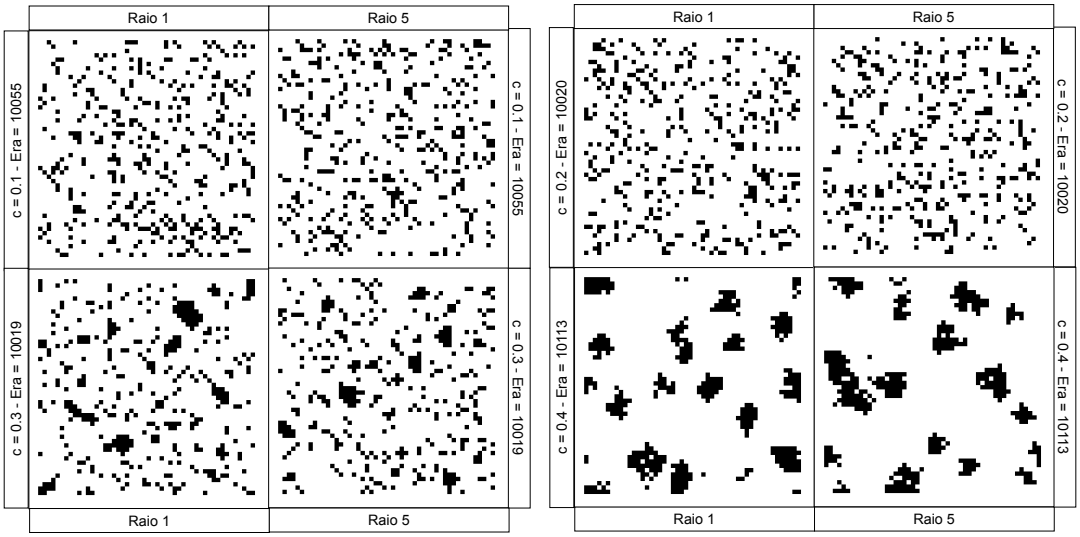

Figura 2. Raio 1 e 5 para as constantes de c = 0.1 à c = 0.4.

Ao analisar as saídas para 10.000 iterações, podemos notar que após 10.000 iterações os grupos são formados para ambos os raios e constantes $c >= 0.4$. A partir disso, é possível observar nas Figura 5 e 6 a execução de 2000 em 2000 iterações, para os parâmetros (i) raio 1 e c = 0.4 e (ii) raio 5 e c = 0.7, respectivamente.

3.2. Dados heterogêneos

Por fim, para dados heterogêneos, utilizando a abordagem das constantes k1 e k2, combinou-se os valores de k1 e k2 variando também de 0.1 à 0.9 com raio 1. Entretanto, os resultados não geraram grupos fechados como nas execuções para dados homogêneos

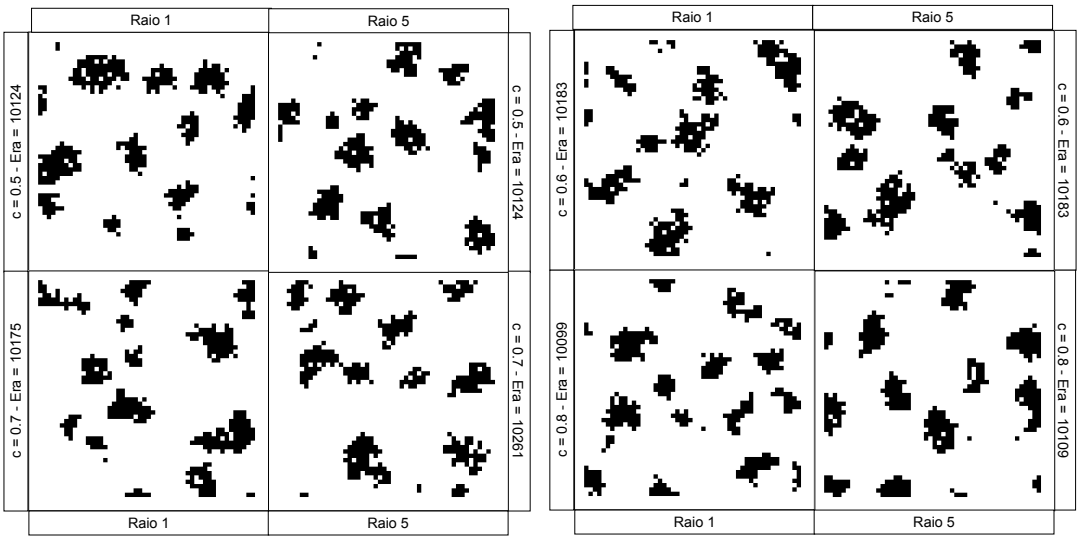

Figura 3. Raio 1 e 5 para as constantes de c = 0.5 à c = 0.8.

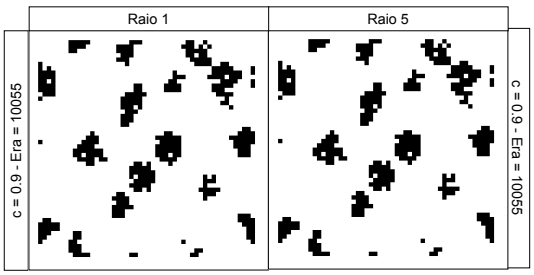

Figura 4. Raio 1 e 5 para a constantes c = 0.9.

para nenhuma das bases de dados A (Figura 7) e B (Figura 8). Entretanto, é possível notar que os dados foram agrupados de acordo com os seus similares (cores iguais) contendo espaços vazios entre os indivíduos.

4. Considerações finais

Algoritmos de *clustering* de dados são de suma importância na Ciência da Computação, em especial na era do *Big Data*. Ao implementar as duas abordagens descritas nas seções anteriores, é possível notar que o algoritmo para dados homogêneos converge rapidamente (em menos de 10 mil iterações já era possível notar a convergência), enquanto que para dados heterogêneos, o algoritmo não convergiu em nenhuma das execuções para 30.000 iterações.

Como a implementação dos algoritmos foi realizada utilizando a linguagem Python, conhecida por possuir tempos de execução elevados em contraste com linguagens clássicas como C e C++, esse pode ter sido o motivo pelo qual o algoritmo não convergiu para as bases heterogêneas: a necessidade de executar os testes por mais iterações.

Para trabalhos futuros se buscará atingir a convergência do algoritmo nas bases heterogêneas para a formação de *clusters* bem definidos, buscando permutar novamente as constantes k1 e k2 por mais de 100 mil iterações para cada, assim como testar o raio de visão das formigas em 1 e 5. Explorar recursos como a paralelização dos cálculos de

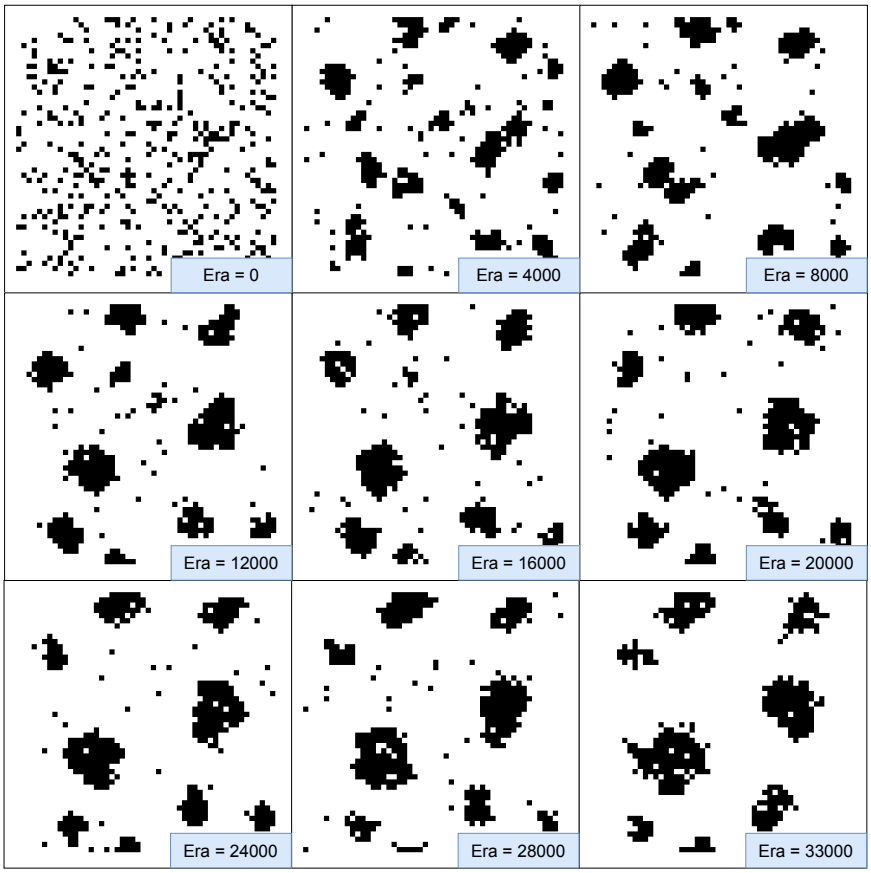

Figura 5. Raio 1 e c = 0.4

probabilidade e movimentação das formigas são tópicos de interesse para etapas futuras, de forma a tornar a execução mais rápida e permitir testes mais longos.

Referências

Gao, W. (2016). Improved ant colony clustering algorithm and its performance study. *Computational Intelligence and Neuroscience*, 2016.

Handl, J. and Meyer, B. (2007). Ant-based and swarm-based clustering. *Swarm Intelligence*, 1(2):95–113.

Lumer, E. D. and Faieta, B. (1994). Diversity and adaptation in populations of clustering ants. In *Proceedings of the third international conference on Simulation of adaptive behavior: from animals to animats 3: from animals to animats 3*, pages 501–508.

Van Rossum, G. and Drake Jr, F. L. (1995). *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.

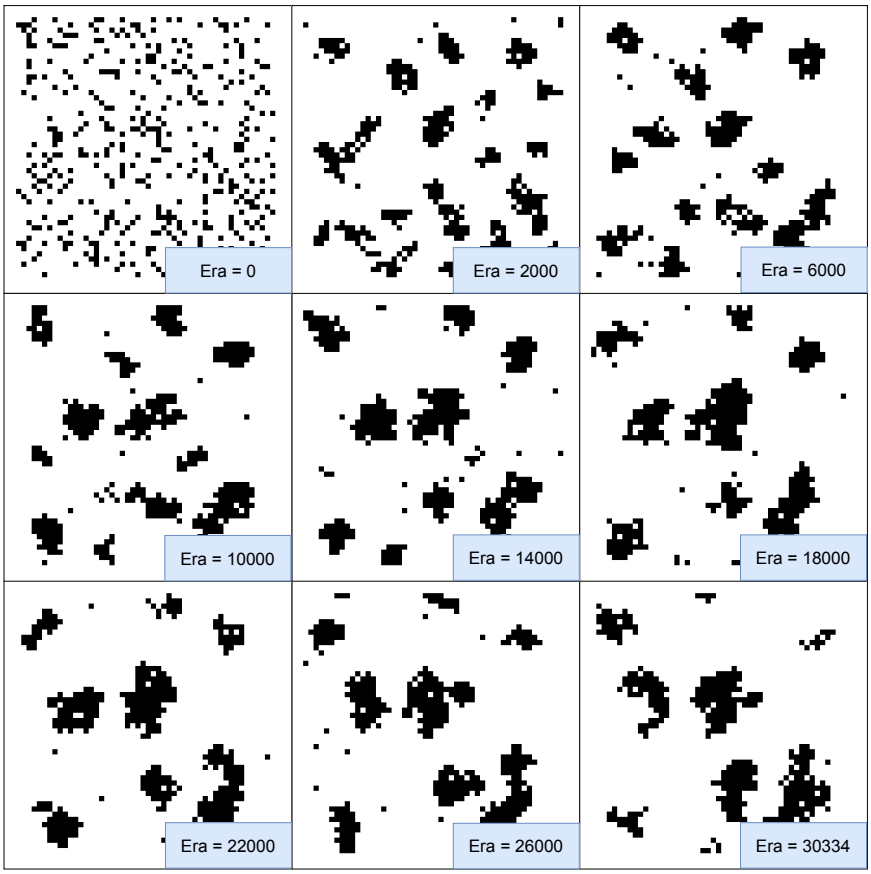

Figura 6. Raio 5 e c = 0.7

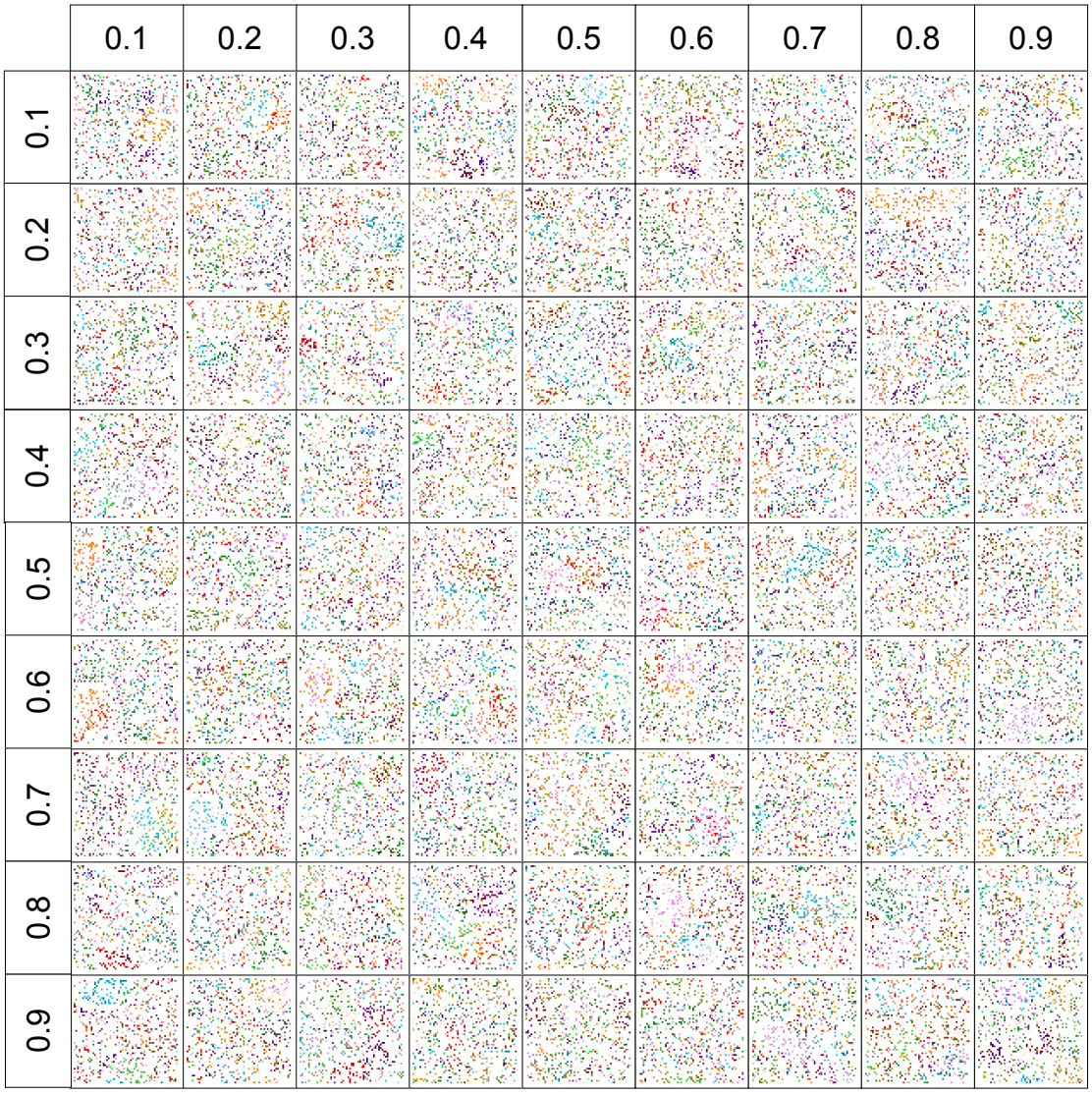

Figura 7. Execução do algoritmo de *clustering* por 30 mil iterações para a base de dados A, com os valores k1 (colunas) e k2 (linhas) variando de 0.1 à 0.9 com raio 1.

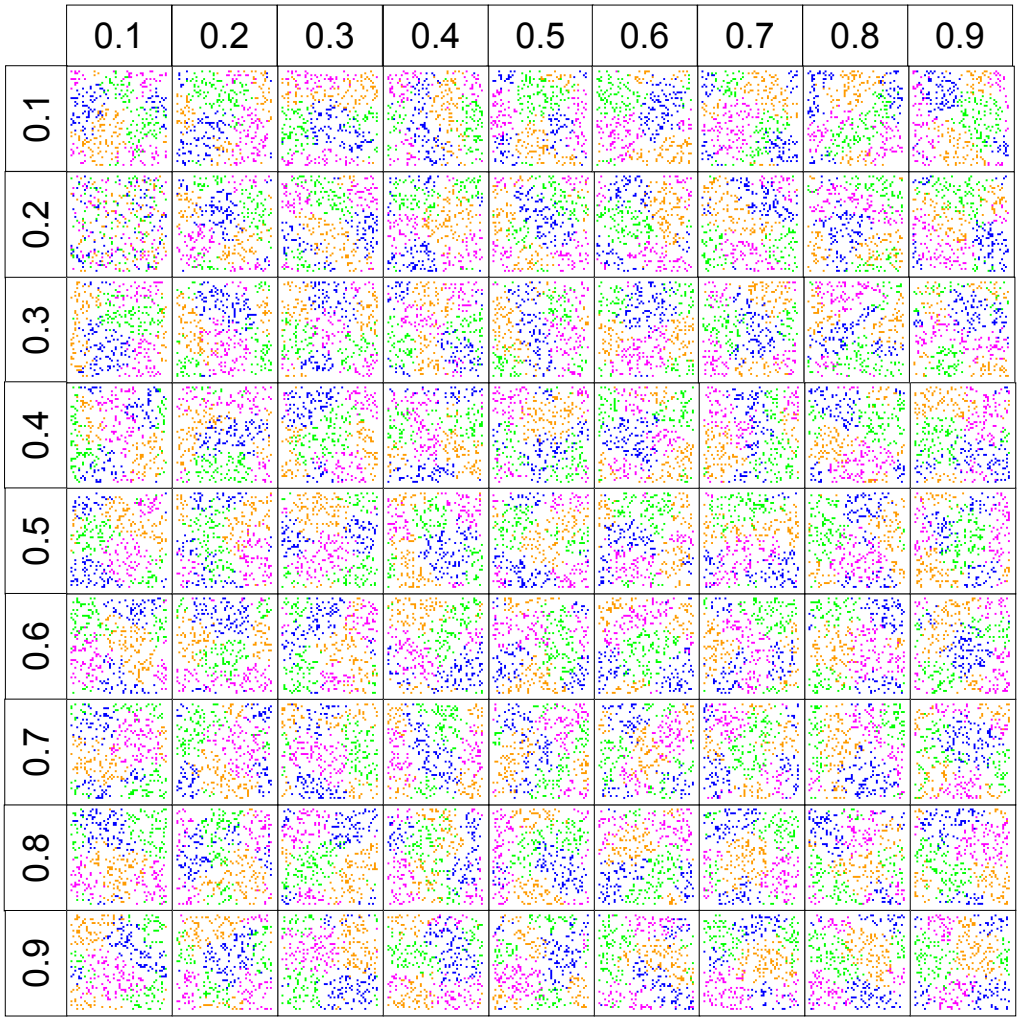

Figura 8. Execução do algoritmo de *clustering* por 30 mil iterações para a base de dados B, com os valores k1 (colunas) e k2 (linhas) variando de 0.1 à 0.9 com raio 1.