Relatório sobre a implementação do algoritmo colonia de formigas

Victor Luiz Bernardes[1]

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

victor.bernardes99@edu.udesc.br

Resumo. *Neste presente artigo serão apresentados os resultados da análise do algorítimo inspirado em colonia de formigas, bem como implementação e desafios para gerar os parâmetros para cada situação e tipo de dado utilizado.*

1. Introdução

Nos últimos tempos, tem havido uma tendência notável de utilização de algoritmos bioinspirados para agrupar grandes conjuntos de dados. Modelos e algoritmos de *clustering* baseados em inteligência de enxame ganharam força devido às suas múltiplas vantagens. Esses benefícios incluem a capacidade de auto-organização, adaptabilidade a conjuntos de dados dinâmicos, robustez no tratamento de dados ruidosos, independência do conhecimento prévio e uma abordagem descentralizada.

Neste presente trabalho utilizaremos do algoritmo de agrupamento inspirado em colonia de formigas para agrupar dados heterogêneos e homogêneos, com diferentes abordagens referentes a implementação das fórmulas para cada situação.

2. Metodologia de Desenvolvimento

Para elaboração dos algoritmos foi utilizada a linguagem de programação Go[1] junto com a biblioteca ebiten[2] para mostrar as iterações do algoritmo de forma visual, facilitando o *debug* da aplicação e análise do comportamento das formigas. A linguagem Go foi escolhida por se tratar de uma linguagem com sintaxe simples, porem competente para trabalhar com paralelismo.

Em todos os casos foram utilizadas funções com probabilidade de Pegar (*Pick*) ou Largar (*Drop*), sendo implementadas de maneiras diferentes a depender do tipo de dado e como foi realizado o cálculo de similaridade entre os eles.

O ambiente é uma matriz quadrada no qual os itens (ou dados) são dispostos de maneira aleatória inicialmente. Também é criado uma matriz quadrada com o mesmo tamanho do ambiente para armazenar o estado das formigas. Cada formiga itera de maneira individual em sua "thread virtual", ou seja, para controlar o acesso à matriz de estados das formigas e ao ambiente foi necessário utilizar RWmutex, visto que neste caso temos muitas leituras para calcular similaridade e poucas escritas na matriz. Usar RWmutex evita que o sistema inteiro fique parado esperando todos lerem, apenas quando é necessária uma escrita é onde todos devem esperar terminar o processamento.

[1] https://go.dev/
[2] https://github.com/hajimehoshi/ebiten

$$f(i) = \begin{cases} \max\left(0.0, \frac{1}{\sigma^2}\sum_{j \in L}(1 - \frac{\delta(i,j)}{\alpha})\right) & if \quad \left(1 - \frac{\delta(i,j)}{\alpha}\right) > 0 \\ 0 & otherwise \end{cases}$$

Figure 1. Fórmula para similaridade [Sadeghi et al. 2008]

2.1. Homogêneo

Para calcular a probabilidade de *pick* e *drop* foram utilizadas as seguintes fórmulas:

$$P_{pick} = (1 - (numVizinhosComItem/qtdVizinhos))^2 \tag{1}$$

$$P_{Drop} = (numVizinhosComItem/qtdVizinhos)^2 \tag{2}$$

Sendo a variável $numVizinhosComItem$ o número de vizinhos que possuem items e $qtdVizinhos$ o total de células vizinhas excluindo a célula atual, ou seja, em um *grid* 3×3, teremos 8 vizinhos.

Como neste caso os items eram apenas valores booleanos não foi necessário fazer uma função de similaridade entre os dados vistos que todos possuem o mesmo comportamento, 0 para *false* ou 1 para *true*.

Nas fórmulas 1 e 2, ao elevar ao quadrado ambas as fórmulas, os items são agrupados de maneira a não deixar espaços entre eles quando ocorre o agrupamento.

2.2. Heterogêneo

Como os tipos de dados diferem, é necessário calcular a distância entre os dados usando algum método como Hamming, Minkowski, Manhattan ou distância Euclidiana. Neste trabalho usaremos a distância Euclidiana.

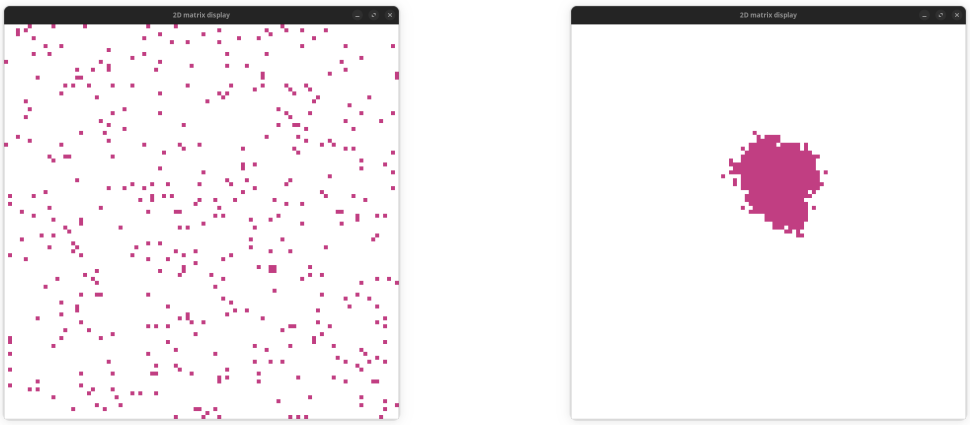
Foi utilizada a fórmula de similaridade da figura 1 para ser usado futuramente na formula de *pick* e *drop*.

Algumas variáveis da fórmula são descritas como:

- α: Depende da distância entre os dados que estamos usando;
- $\delta(i, j)$: Distancia entre i e seu vizinho j, neste caso distância euclidiana;
- σ: Numero de vizinhos ao redor.

Nas fórmulas de pegar e largar são usados valores chamados de K_1 para a função de pegar e K_2 para a função de largar. Esses valores são definidos empiricamente a partir de testes realizados com a base de dados. Ao contrário da fórmula citada por [Sadeghi et al. 2008], neste caso estamos usando a fórmula elevada ao cubo, pois o resultado foi um agrupamento sem espaços entre os dados.

$$P_{pick} = \begin{cases} 0 & \text{if } f(i) \geq K_1 \\ (\frac{K_1}{K_1 + f(i)})^3 & \text{if } f(i) < K_1 \end{cases} \tag{3}$$



(a) Geração dos items na *grid*

(b) Agrupamento após n iterações

Figure 2. Visualização do agrupamento homogêneo

$$P_{drop} = \begin{cases} 1 & \text{if } f(i) \geq K_2 \\ (\frac{f(i)}{(f(i)+K_2)})^3 & \text{if } f(i) < K_2 \end{cases} \tag{4}$$

3. Experimentos

3.1. Agrupamento homogêneo

Na figura 2 podemos observar o comportamento do sistemas com as seguintes variáveis:

- Tamanho da matriz = 100x100
- Numero de formigas = 20
- Numero de items = 400
- Raio de visão da formiga = 1
- Número de iterações = 10.000.000

Para fins de análise foi feito um teste com o raio da formiga = 5. A figura 3 mostra que os items ficaram ainda meio espaçados entre eles, isso é uma consequência do grande número de vizinhos que cada célula agora tem, no caso com raio 5 teremos uma visão de 5×5 células ao redor.

3.2. Agrupamento heterogêneo

Neste experimento, os items agora darão lugar aos dados. Estes dados que por sua vez tem duas dimensões (x,y) porem ainda são homogêneos visto que são do mesmo tipo float64. Com os dados também é recebido uma *label* que será usada para identificar os grupos visualmente.

Na figura 4, podemos observar como os dados foram gerados, usando python3 e a biblioteca matplotlib para obter o grafico do posicionamento de ambos os casos. Podemos observar que a figura B, com 4 grupos, é algo bem-comportado, ficando cada grupo em um quadrante separado. Isso já nos dá ideia de como o comportamento do experimento precisa ser conduzido. Já o grupo com 15, é algo bem aleatório.

Para gerar a figura 5 foi necessário usar a seguinte configuração:

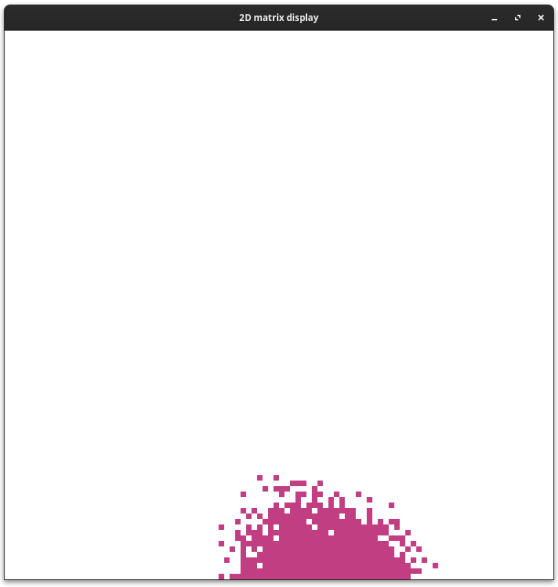
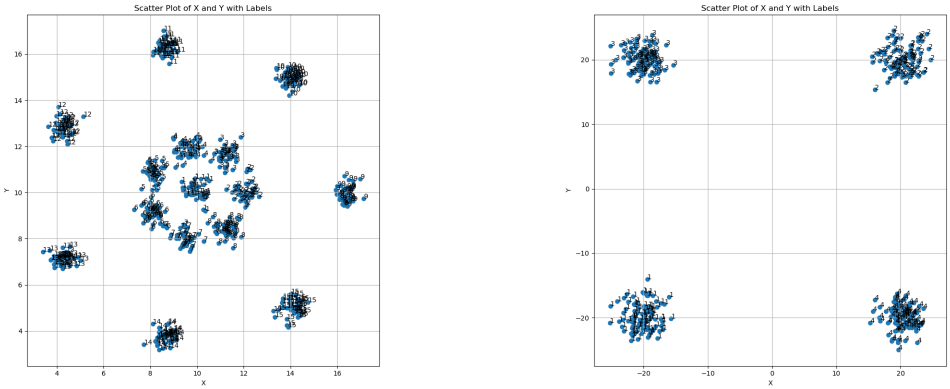
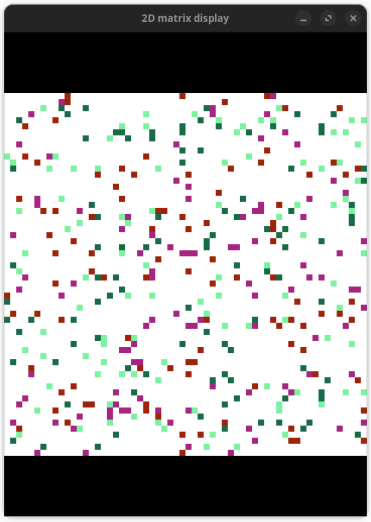


Figure 3. Testes com raio de visão da formiga = 5

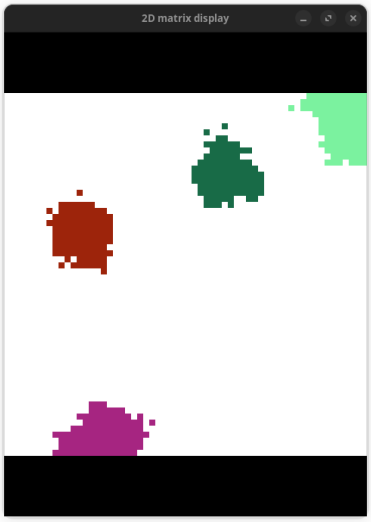


(a) 15 Grupos (b) 4 Grupos

Figure 4. Visualização do *dataset* usado no agrupamento



(a) Geração dos items na *grid*



(b) Agrupamento após n iterações

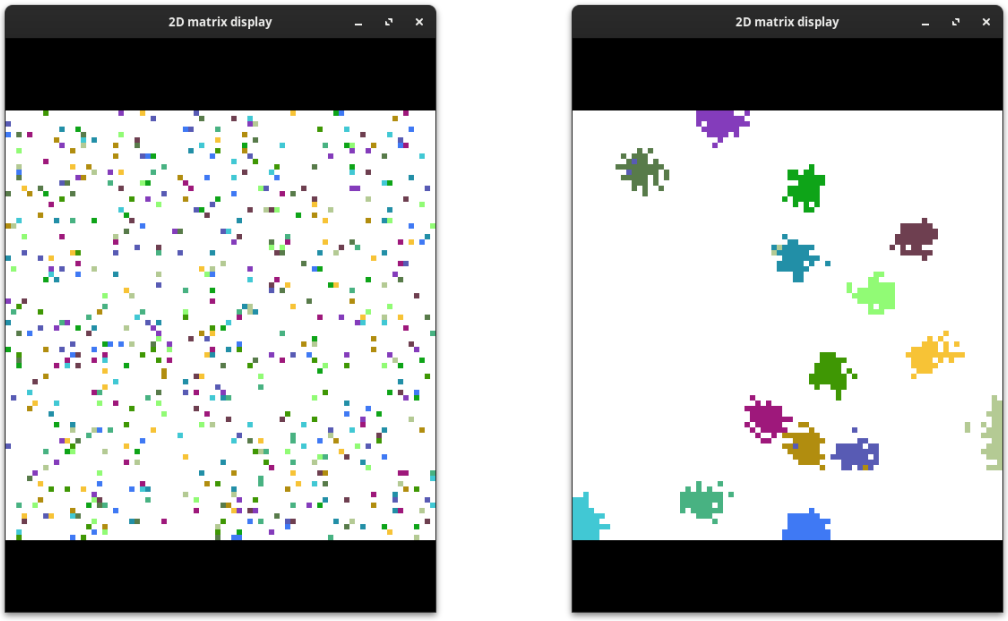
Figure 5. Visualização do agrupamento para 4 grupos

- Tamanho da matriz = 60x60;
- Numero de formigas = 20;
- Numero de items = 400;
- Raio de visão da formiga = 1;
- Número de iterações = 10.000.000;
- α: 20, isso baseado na distância de items do mesmo grupo, ou seja, pode variar com 18 até 24 em alguns casos, mas é uma média que ficou boa nos testes realizados;
- K_1 = 0.4
- K_2 = 0.6

Para K_1 e K_2 foram realizados testes baseados na fórmula de *pick* e *drop* sabendo que a formula de similaridade influencia se ele ficará com 100% de chance de pegar ou não.

Igualmente, para gerar a figura 6 foi necessária uma configuração diferente dos 4 grupos:

- Tamanho da matriz = 80x80 (tamanho da matriz foi definido empiricamente após testes);
- Numero de formigas = 20;
- Numero de items = 600;
- Raio de visão da formiga = 1;
- Número de iterações = 10.000.000;
- α: 1.25 (bem diferente dos 4 grupos, isso ocorre devido à distância entre os dados do mesmo grupo ser bem menor)
- K_1 = 0.4;
- K_2 = 0.3;



(a) Geração dos items na *grid* (b) Agrupamento após n iterações

Figure 6. Visualização do agrupamento para 15 grupos.

Neste caso, para K_1 e K_2 também foram realizados testes empíricos para determinar os valores que poderiam ser atribuídos conforme amostragem visual que obtemos.

No experimento realizado por [Sadeghi et al. 2008] foram usados valores $K_1 = 0.15$ e $K_2 = 0.15$ com $\alpha = 0.5$, mas por que não podemos simplesmente usar esses mesmo valores na aplicação? Simples, base de dados diferentes. O autor em questão não deixou publico a base de dados, então usar essas mesmas constantes não fará sentido.

O presente trabalho pode ser encontrado

4. Análise dos experimentos

Após realizados diversos testes e ajustes nas variáveis de configuração do projeto, podemos definir que parte importante de se usar esse tipo de algoritmo é definir bem o conjunto de dados que serão usados. Após definição dos dados, será pensando em uma estratégia de agrupamento a depender de quantas variáveis possuímos, se é homogêneo ou heterogêneo.

Utilizar-se do PEAS (desempenho, ambiente, atuadores e sensores) para elaborar essa etapa inicial de obter os requisitos e principais objetivos do sistema para entender de que forma podemos abordar o problema. Quanto mais informações obtivemos a partir da análise PEAS melhor será o resultado, visto que o escopo estará bem definido.

Após definido os requisitos do sistema, implementar será apenas uma questão de escolha de ferramentas e métodos, como o caso do cálculo de distância usado para agrupar os grupos de 4 e 15 dados. Neste caso, escolhemos distancia euclidiana por se tratar de uma fórmula conhecida e neste cenário gerou resultados satisfatórios.

5. Conclusão e trabalhos futuros

A utilização de agrupamentos de dados possui uma importância grande para auxiliar a tomada de decisão, desde bolsa de valores até em uma empresa. A utilização da linguagem GO foi extremamente competente para controlar as formigas paralelamente e controlar o acesso aos recursos do sistema eficientemente, utilizando todo o CPU em todos os testes realizados com baixo consumo de memória.

Após realizados os testes e definições de parâmetros de forma empírica, podemos notar que as técnicas aqui utilizadas podem ser levadas para outros campos de conhecimento. Um exemplo da utilização desdas técnicas em uma abordagem futuras seria o agrupamento de ações baseados em coeficientes já pré-definidos como EBITA, dividas da empresa, lucro líquido e bruto entre outros coeficientes, agrupando as ações por nível de risco.

Para trabalhos futuros seria necessário testar o algoritmo com diferentes formulas de distância, visto que só trabalhamos neste artigo com distância euclidiana e pesquisar métodos para definir o tamanho da *grid* a partir do tamanho de items. Também seria interessante realizar um *benchmark* utilizando GPU para observar os tempos de agrupamento.

References

Sadeghi, Z., Teshnehlab, M., and Pedram, M. M. (2008). K-ants clustering - a new strategy based on ant clustering.