Computação Evolutiva aplicada á detecção de fraudes em sistema de pagamentos de plano de saúde

Bruno M. Pires[1]

[1]Departamento de Ciências da Computação
Universidade do Estado de Santa Catarina (UDESC)
Caixa Postal 15.064 – 88.035-901 – Joinville – SC – Brasil

bruno.pires@edu.udesc.br

Abstract. *In this report, we provide a detailed analysis of the practical implementation of an ant-inspired clustering algorithm. Clustering plays a pivotal role in data organization across various domains, including artificial intelligence. This algorithm offers a distinctive and effective approach, harnessing its ability to efficiently group data while considering complex features and multidimensional variables.*

Resumo. *Neste relatório, apresentamos uma análise detalhada da aplicação prática de um algoritmo de clusterização bio-inspirado em formigas. A clusterização desempenha um papel fundamental na organização de dados em várias áreas, incluindo a inteligência artificial, e este algoritmo oferece uma abordagem única e eficaz, explorando a capacidade do algoritmo de agrupar dados de maneira eficiente, considerando características complexas e variáveis multidimensionais.*

1. Introdução

Dentro de Inteligência artificial diversas áreas foram desenvolvidas levando como base alguns comportamentos que acontecem na natureza, a final de contas, se estes comportamentos resolvem alguns problemas dentro de suas realidades, porque não tentar utilizar a mesma lógica para resolver problemas que inicialmente não possuem muita ligação natural?

Um algoritmo de clusterização baseado em formigas é uma técnica de agrupamento de dados que se inspira no comportamento das formigas para resolver problemas de clusterização. Esse tipo de algoritmo faz parte da categoria de algoritmos bioinspirados, que buscam soluções para problemas complexos modelando o comportamento de organismos ou sistemas naturais.

Durante estudos, notou-se que quando formigas são postas em um ambiente com diversas formigas mortas ao redor, o comportamento padrão delas é realizar o agrupamento dos corpos, este comportamento é também observado dentro dos formigueiros, e tem como objetivo manter a higiene da colônia. Extrapolando isso para uma visão computacional, é possível imaginar aplicações desse comportamento específico para resolver problemas reais. O principal problema que pode ser atacado pela abordagem, é o agrupamento de dados e mineração de dados.

Basicamente, a mineração de dados é uma área de inteligência artificial que tem por objetivo encontrar grupos naturais de dados em grandes conjuntos de dados, ajudando

na descoberta de padrões e muitas vezes auxiliando empresas na tomada de decisões estratégicas.

O artigo tem como objetivo relatar o que se foi observado aplicando esta técnica em 3 simples casos:

- Itens: Aqui as formigas agrupam itens simples, ou seja, dados homogêneos, sem distinção entre um item e outro
- Grupo de 4 dados: Aqui as formigas estarão em um ambiente com 4 grupos de dados (itens) distintos
- Grupo de 15 dados: Aqui as formigas estarão em um ambiente com 15 grupos de dados (itens) distintos

O relatório estará estruturado em 4 seções, esta dedicada a Introdução, uma dedicada a explicar o desenvolvimento do trabalho, uma dedicada a apresentação dos resultados e por último, a conclusão e trabalhos futuros.

2. Desenvolvimento

Assim como no desenvolvimento de software, inicialmente foi necessário realizar uma espécie de ”modelagem” do problema, pra enteder melhor suas características. De forma básica, a tabela mostrada abaixo for preenchida, utilizando os conceitos também mostrados abaixo:

- Agentes: O agente é a entidade que toma decisões e interage com o ambiente.
- Observável: Refere-se à capacidade do agente de observar o estado atual do ambiente.
- Determinístico: Indica se o comportamento do agente é completamente previsível com base no estado atual e nas ações tomadas.
- Episódico: Indica se o agente toma decisões em episódios ou etapas discretas, onde as ações tomadas em um episódio não afetam diretamente os episódios subsequentes.
- Estático: Refere-se a se o ambiente muda ao longo do tempo.
- Discreto: Indica se o espaço de ações ou estados do agente é discreto, o que significa que existem um número finito e bem-definido de opções.
- Multiagente: Indica se existem múltiplos agentes interagindo no mesmo ambiente

Agente	Observável	Determinístico	Episódico	Estático	Discreto	Multiagente
Formigas	Parcialmente	Não	Não	Não	Sim	Sim

Table 1. Modelagem das característica do ambiente

Entendendo as características do ambiente, é possível iniciar a codificação. Basicamente a área de atuação das formigas é representada por uma matriz NxN, chamada de board, que simula uma esfera, ou seja, quando uma formiga atravessa uma borda, ela aparece no lado oposto da tela. O sistema mantém uma estrutura de lista onde cada posição representa um objeto do tipo ”Formiga” e elas iteram sobre o nosso board cada uma de forma aleatória, é importante salientar que cada formiga possui uma visão, que é a quantidade de itens que ela enxerga ao seu redor. No caso dos testes realizados, todas elas tinham raio igual a um, ou seja, exergavam os 8 slots de posição ao seu redor.

Para facilitar os testes, foi desenvolvida uma ferramenta em python utilizando uma biblioteca de interface gráfica que gera HTML chamada ”streamlit” juntamente com uma biblioteca para aplicações gráficas chamada pygame. A interface em HTML é responsável pelas parametrizações do sistema (Número de formigas, tamanho do board...) e por chamar o pygame que por sua vez executa a clusterização em si. A figura 1 representa um dos menus da ferramenta.

Figure 1. Tela Inicial Ferramenta de aplicação de Algoritmo Bio-Inspirado

Descrevendo um possível pseudo-código da aplicação, inicialmente temos a alocação do espaço de busca (board), logo após, a disposição estocástica dos itens ou dados por todo board, seguido pela disposição também aleatória das formigas. Quando tudo estiver alocado, um loop é responsável pelo agrupamento, fazendo com que as formigas andem pelo board de forma completamente aleatória, não podendo ocupar posições onde exista outra formiga, e não podendo voltar para a posição que ocupou 1 iteração

anterior, em cada posição que uma formiga para, uma ação deve ser tomada levando em consideração o estado da formiga, se ela está vazia ou carregando algo, e o estado da posição em si, se contém ou não um item. Essa ação chamada de ”pick” e ”drop” é calculada com base em fórmulas de probabilidade, e no problema que está sendo atacado.

Com finalidade de deixar mais pálpavel em termos de código, a figura 2 representa a codificação da classe formiga, visto que todo código foi implementado utilizando-se conceitos de orientação a objetos. O exemplo a esquerda representa a codificação da formiga que trabalha com itens simples, e da direita a formiga que trabalha com dados. E a figura 3 apresenta o método responsável pelo movimento das formigas, igual em ambas as classes, onde é randomizado o próximo passo da formiga, dentro de uma lista de possibilidades, visto que a posição imediatamente anterior não deve ser revisitada.

```python
import random
import pygame
class Ant():
    def __init__ (self, row, column):
        self.vision = 1
        self.row = row
        self.column = column
        self.payload = " "
        self.last_position = None

    def move(self, boardDimension):···
    def pintaPos(self, screen, cor, pos):···
```

```python
import random
import pygame
from dataAnts.dataType import *

class ScrappyAnt():
    def __init__ (self, row, column):
        self.vision = 1
        self.row = row
        self.column = column
        self.payload = None
        self.last_position = None

    def move(self, boardDimension):···

    def pintaPos(self, screen, cor, pos):···

    def setPayload(self, payloadFromBoard):···
```

Figure 2. Codificação da classe Formiga, Equerda para itens, Direita para dados

```python
def move(self, boardDimension):
    possibilities = [(0, 1), (0, -1), (1, 0), (-1, 0)]

    # Remove a posição atual do conjunto de posições visitadas
    if self.last_position:
        possibilities.remove((self.last_position[0] - self.row, self.last_position[1] - self.column))

    # Remove as posições inválidas das possibilidades
    valid_possibilities = [(dr, dc) for dr, dc in possibilities
                           if 0 <= self.row + dr < boardDimension and 0 <= self.column + dc < boardDimension]

    if valid_possibilities:
        direction = random.choice(valid_possibilities)
        next_row = self.row + direction[0]
        next_col = self.column + direction[1]

        # Atualiza a posição anterior
        self.last_position = (self.row, self.column)

        self.row = next_row
        self.column = next_col
```

Figure 3. Codificação do método move

3. Aplicação em Itens simples

Como mencionado, o primeiro cenário de aplicação e teste do algoritmo foi em itens homogêneos, aqui temos uma situação bastante simples, onde em uma matriz de espaços

vazios, algumas strings ”1” são espalhadas de forma aleatória, e as formigas são responsáveis por agrupar estes itens.

A definição matemática para decisão da formiga depende dos estados do agente, que podem ser os seguintes:

- Formiga Vazia em posição cheia: aqui a Probabilidade da formiga pegar o item da posição é dada pela fórmula:

$$P_{\text{pick}} = \left(1 - \frac{\text{qtdItens}}{\text{spaces}}\right)^2$$

  Onde ”qtdItens” representa a quantidade de itens ao redor da formiga, respeitando o raio de visão. E ”spaces” representa a quantidade de casas que a formiga enxerga, respeitando o raio de visão.
- Formiga Carregando em posição vazia: aqui a probabilidade da formiga largar o item que carrega é dada pela fórmula:

$$P_{\text{drop}} = \left(\frac{qtdItens}{spaces}\right)^2$$

- Observação: Para os casos onde a Formiga está carregando algo em uma posição cheia, e onde a Formiga não carrega nada em uma posição vazia, nenhuma ação pode ser tomada.

Sabendo das probabilidades, e tendo entendido o funcionamento do código, foi realizado um teste com as seguintes especificações:

- Quantidade de itens: 1000 itens
- Quantidade de formigas: 15 formigas
- Tamanho do Board: 60x60
- Quantidade de iterações: 1.000.000

A figura 4 representa o estado do board no início do algoritmo, enquanto a figura 5 representa o estado do board após as 1.000.000 iterações. Percebe-se que as formigas formaram dois clusters de itens, tudo isso sem qualquer relação entre as formigas, todas agindo por conta própria, provavelmente em uma situação de longo prazo com mais iterações ainda, ambos os clusters convergiriam para um único grande conglomerado de itens. Os itens nos cantos da imagem parecem separados, mas é importante lembrar da característica infinita do board, simulando uma esfera, ou seja, para a formiga as posições extremas do board são consecutivas.

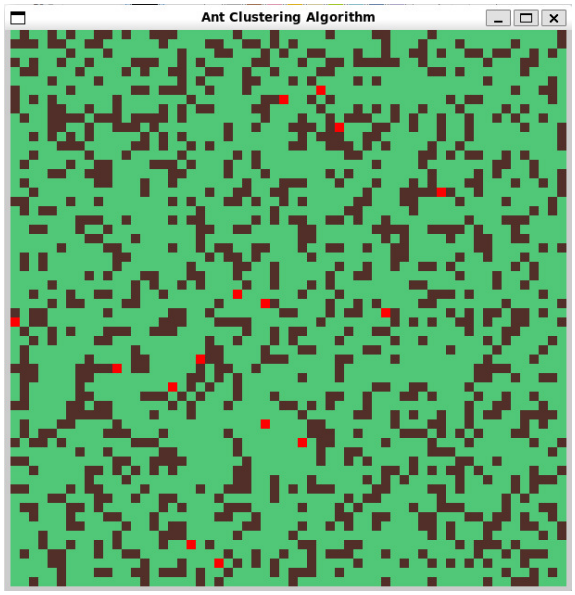

Figure 4. Início da clusterização

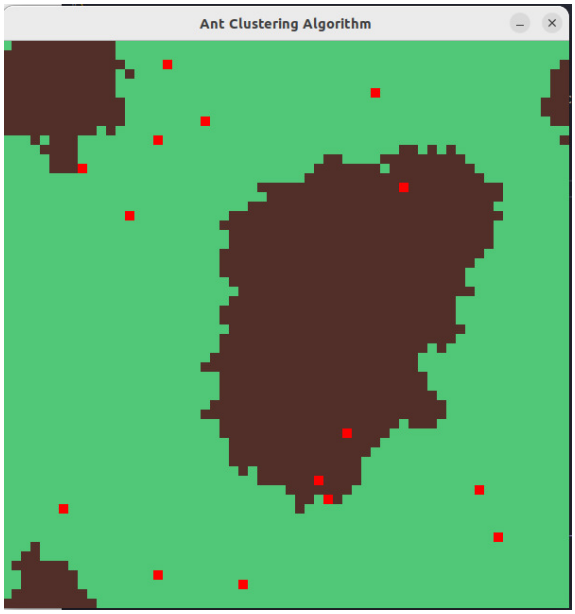

Figure 5. Final da clusterização

4. Aplicação em ambiente com dados heterogêneos

Aqui, temos um cenário mais complexo, foi utilizado dois datasets, um com 400 dados de 4 tipos diferentes, e outro de 600 dados de 15 tipos diferentes. A estrutura do dado em ambos os casos é idêntica, cada dado possui um valor referente a uma coordenada X, e uma coordenada Y, bem como um rótulo que o diferencia dos demais, e de forma prática foi apenas utilizado para pintá-los na interface gráfica. O funcionamento do algoritmo é exatamente igual ao caso anterior, mudando somente a lógica de pegar ou largar um item, que neste caso é utilizada com auxílio de uma função de similaridade, representada pela

figura 6, onde d(i,j) é a distancia euclidiana entre o item e um vizinho, α é um fator que define a escala para dissimilaridade definido empiricamente, e s é a dimensão do grid de visão da formiga:

$$f(i) = \begin{cases} \frac{1}{s^2} \sum_j (1 - d(i,j)/\alpha) & \text{if } f(i) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Figure 6. Função de similaridade

Para cada caso do agente, temos as seguintes probabilidades:

- Formiga Vazia em posição cheia: aqui a Probabilidade da formiga pegar o item da posição é dada pela seguinte fórmula, sendo k1 empiricamente definido:

$$P_{\text{pick}(x_i)} = \left(\frac{k_1}{k_1 + f(x_i)} \right)^3$$

  Onde "qtdItens" representa a quantidade de itens ao redor da formiga, respeitando o raio de visão. E "spaces" representa a quantidade de casas que a formiga enxerga, respeitando o raio de visão.
- Formiga Carregando em posição vazia: aqui a probabilidade da formiga largar o item que carrega é dada pela seguinte fórmula, sendo k2 empiricamente definido:

$$P_{\text{drop}(x_i)} = \left(\frac{f(x_i)}{k_2 + f(x_i)} \right)^3$$

- Observação: Para os casos onde a Formiga está carregando algo em uma posição cheia, e onde a Formiga não carrega nada em uma posição vazia, nenhuma ação pode ser tomada.

4.1. 4 Grupos

Para execução com 4 diferentes grupos de dados, os seguinte parâmetros foram utilizados:

- Quantidade de itens: 400 itens divididos em 4 grupos distintos
- Quantidade de formigas: 20 formigas
- Tamanho do Board: 55x55
- Quantidade de iterações: 1.500.000
- $k1 = 0.3$, $k2 = 0.6$, $\alpha = 25$

A figura 7 representa o estado do board no início do algoritmo, enquanto a figura 8 representa o estado do board após as 1.500.000 iterações. Percebe-se que as formigas formaram um cluster de itens para cada dado, representado pelas diferentes cores. Algo interessante de se notar, é que diferente do caso anterior, os clusters agora estão mais dispersos, por mais que seja possível notar o agrupamento e distinção dos dados, ainda existem muitas lacunas nas placas. Provalvelmente é algo que pode ser solucionado com um ajuste mais fino dos parâmetros usados, mas no geral o resultado está bastante alinhado com o que era esperado.

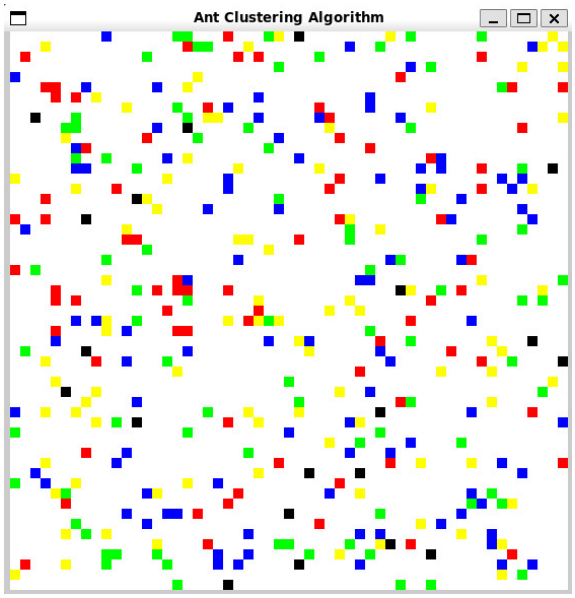

Figure 7. Início da clusterização

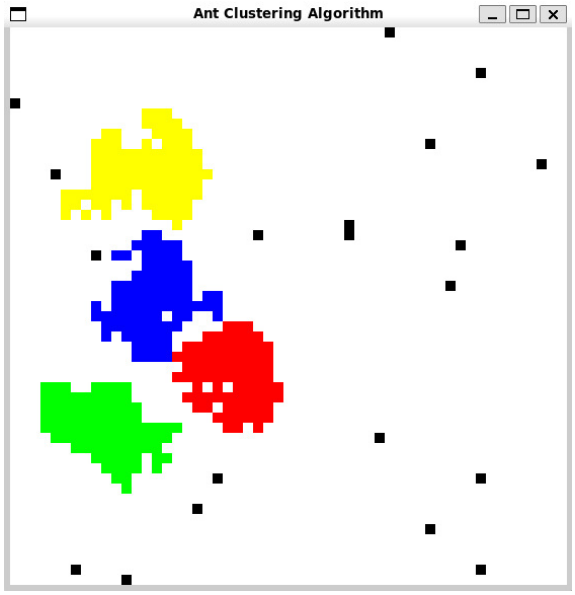

Figure 8. Final da clusterização

4.2. 15 Grupos

Para execução com 15 diferentes grupos de dados, os seguinte parâmetros foram utilizados:

- Quantidade de itens: 600 itens divididos em 15 grupos distintos
- Quantidade de formigas: 20 formigas
- Tamanho do Board: 70x70
- Quantidade de iterações: 3.500.000
- $k1 = 0.15$, $k2 = 0.35$, $\alpha = 2$

A figura 9 representa o estado do board no início do algoritmo, enquanto a figura 10 representa o estado do board após as 3.500.000 iterações. Percebe-se que as formigas formaram um cluster de itens para cada dado, representado pelas diferentes cores. Neste caso, algo interessante de se notar é que devido ao ajuste de parâmetros, alguns dados estão agrupados com dados diferentes. Provavelmente com tendência de longo prazo, esses problemas de incompatibilidade ou ruído deveriam ser ajustados com o passar da iterações.

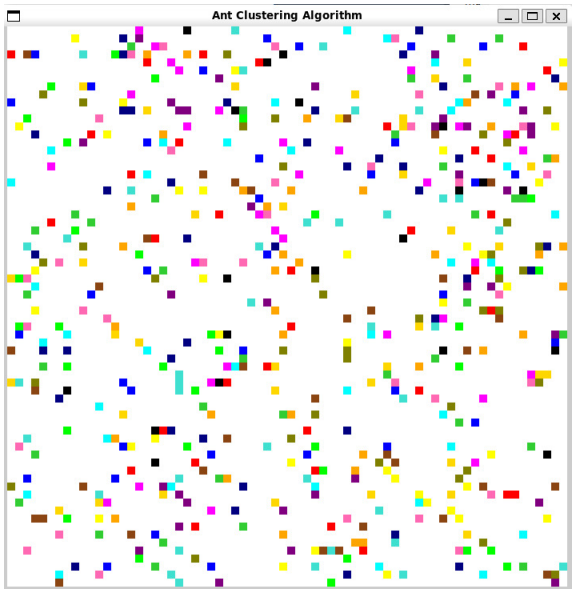

Figure 9. Início da clusterização

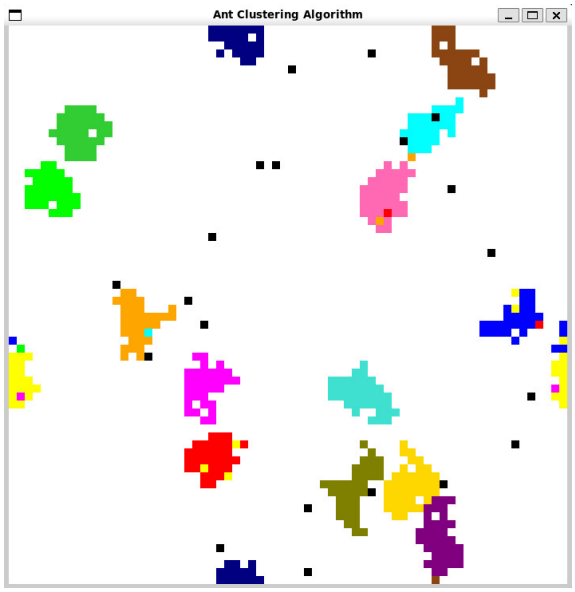

Figure 10. Final da clusterização

5. Conclusão e Trabalhos Futuros

A ferramenta desenvolvida tinha como objetivo aplicar um algoritmo bio-inspirado em diferentes casos, e estudar o comportamento do mesmo. Esse objetivo foi alcançado com sucesso e permitiu a consolidação de diversos conceitos aprendidos em sala de aula. Para acesso a ferramenta, um e-mail pode ser enviado ao autor solicitando acesso ao repositório no github que contêm todos os fontes.

Alguns problemas foram encontrados, principalmente nos cenários de dados heterogêneos, onde alguns clusters possuem um ruído, ou seja, dados que não deveriam estar agrupados ali, isso poderia ser tratado em trabalhos futuros, bem como a implementação de uma lógica de paralelismo, tratando cada formiga como uma thread e cuidando dos possíveis casos de deadlock.