

# Practica 5. Optimización por Colonia de Hormigas

## Algoritmos Bioinspirados

Diego Castillo Reyes  
Marthon Leobardo Yañez Martinez  
Aldo Escamilla Resendiz

6 de junio de 2024

# Índice general

## 1. Introducción

El Algoritmo de Optimización por Colonia de Hormigas (ACO, por sus siglas en inglés) es una técnica meta-heurística inspirada en el comportamiento de las hormigas en la naturaleza. Este algoritmo, propuesto inicialmente por Marco Dorigo en la década de 1990, simula la forma en que las hormigas encuentran caminos óptimos entre su colonia y una fuente de alimento.

En la naturaleza, las hormigas depositan feromonas en su camino al buscar comida, creando un rastro químico que guía a otras hormigas. Con el tiempo, los caminos con más feromonas se vuelven más atractivos, y las hormigas tienden a seguir estos senderos más prometedores, lo que eventualmente conduce a la convergencia en el camino más corto.

El ACO utiliza este principio para resolver problemas de optimización combinatoria. En este algoritmo, una colonia de hormigas artificiales construye soluciones al problema, y durante este proceso, cada hormiga deposita feromonas en el entorno, influyendo en las decisiones de las hormigas subsiguientes. Los caminos que resultan en mejores soluciones reciben más feromonas, aumentando la probabilidad de ser seleccionados en iteraciones futuras.

## 2. Problema de Asignación Cuadrática

```
1  '''
2  Resolver el problema de asignacion
3  cuadratica usando ACO.
4  '''
5  import numpy as np
6  import itertools
7
8  def selecciona_siguiente_ciudad(actual, no_visitadas, feromona, distancia, alpha, beta, epsilon=1e
   -10):
9      feromona_actual = feromona[actual][no_visitadas] ** alpha
10     visibilidad = (1.0 / (distancia[actual][no_visitadas] + epsilon)) ** beta
11     probabilidades = feromona_actual * visibilidad
12
13     # Chequear y manejar valores infinitos o NaN en probabilidades
14     if np.any(np.isinf(probabilidades)) or np.any(np.isnan(probabilidades)):
15         probabilidades = np.nan_to_num(probabilidades, nan=epsilon, posinf=epsilon, neginf=epsilon
   )
16
17     suma_probabilidades = sum(probabilidades)
18
19     # Chequear si la suma de probabilidades es cero o infinito para evitar NaNs
20     if suma_probabilidades == 0 or np.isinf(suma_probabilidades) or np.isnan(suma_probabilidades):
21         probabilidades = np.ones_like(probabilidades) / len(probabilidades)
22     else:
23         probabilidades /= suma_probabilidades
24
25     return np.random.choice(no_visitadas, 1, p=probabilidades)[0]
26
27 def construye_camino(inicio, n_ciudades, feromona, distancia, alpha, beta):
28     camino = [inicio]
29     no_visitadas = list(range(n_ciudades))
30     no_visitadas.remove(inicio)
```

```

31
32     actual = inicio
33     while no_visitadas:
34         siguiente = selecciona_siguiente_ciudad(actual, no_visitadas, feromona, distancia, alpha,
35         beta)
36         camino.append(siguiente)
37         no_visitadas.remove(siguiente)
38         actual = siguiente
39
40     return camino
41
42 def computa_costo(camino, distancia, flujo):
43     costo = 0
44     for i in range(len(camino)):
45         for j in range(len(camino)):
46             costo += flujo[i][j] * distancia[camino[i]][camino[j]]
47     return costo
48
49 def actualiza_feromona(feromona, caminos, costos, evaporacion, q):
50     feromona *= (1.0 - evaporacion)
51     for camino, costo in zip(caminos, costos):
52         for i in range(len(camino) - 1):
53             feromona[camino[i]][camino[i+1]] += q / costo
54             feromona[camino[-1]][camino[0]] += q / costo
55
56 def aco(distancia, flujo, n_hormigas, n_iteraciones, evaporacion, alpha=1, beta=5, q=1):
57     n_ciudades = distancia.shape[0]
58     feromona = np.ones(distancia.shape) / n_ciudades
59
60     mejor_camino = None
61     mejor_costo = float('inf')
62
63     for _ in range(n_iteraciones):
64         caminos = [construye_camino(np.random.choice(n_ciudades), n_ciudades, feromona, distancia,
65         alpha, beta) for _ in range(n_hormigas)]
66         costos = [computa_costo(camino, distancia, flujo) for camino in caminos]
67         actualiza_feromona(feromona, caminos, costos, evaporacion, q)
68         min_costo = min(costos)
69         if min_costo < mejor_costo:
70             mejor_costo = min_costo
71             mejor_camino = caminos[costos.index(min_costo)]
72
73     return mejor_camino, mejor_costo
74
75 def leer_matrices(file_path):
76     with open(file_path, 'r') as file:
77         lines = file.readlines()
78
79     matrices = []
80     current_matrix = []
81     for line in lines:
82         stripped_line = line.strip()
83         if stripped_line:
84             current_matrix.append(list(map(int, stripped_line.split())))
85         else:
86             if current_matrix:
87                 matrices.append(np.array(current_matrix))
88                 current_matrix = []
89     if current_matrix:
90         matrices.append(np.array(current_matrix))
91
92     return matrices
93
94 def ANOVA(distancia, flujo, n_iteraciones, evaporacion, alphas, betas, n_hormigas):
95     resultados = []
96     combinaciones = list(itertools.product(n_hormigas, alphas, betas))
97
98     for n_h, alpha, beta in combinaciones:
99         mejor_camino, mejor_costo = aco(distancia, flujo, n_h, n_iteraciones, evaporacion, alpha,
100         beta)
101         resultados.append((n_h, alpha, beta, mejor_camino, mejor_costo))

```

```

99
100     return resultados
101
102 # Usar las matrices leídas
103 # ! Cambiar el path al archivo
104 # file_path = 'Algoritmos Bioinspirados/Practica 5/matricesProblemaQAP/matricesProblemaQAP/tai15.
    dat'
105 file_path = 'matricesProblemaQAP/matricesProblemaQAP/tai30.dat'
106 matrices = leer_matrices(file_path)
107 distancia = matrices[1]
108 flujo = matrices[2]
109
110 # Definir los parámetros
111 n_hormigas = [10, 25, 50, 100]
112 alphas = [50, 100, 200]
113 betas = [2, 10]
114 n_iteraciones = 100
115 evaporacion = 0.5
116
117 # Ejecutar experimentos
118 resultados = ANOVA(distancia, flujo, n_iteraciones, evaporacion, alphas, betas, n_hormigas)
119
120 # Imprimir resultados
121 for i, (n_h, alpha, beta, mejor_camino, costo) in enumerate(resultados):
122     print(f"{i + 1}. Número de hormigas: {n_h}, alpha: {alpha}, beta: {beta}, Mejor camino: {
        mejor_camino}, Costo: {costo}")

```

```

PowerShell
DiegoDEMON Practica 5 main >_ pwsh python AC0.py
1. Número de hormigas: 10, alpha: 50, beta: 2, Mejor camino: [2, 3, 1, 0, 4], Costo: 62
2. Número de hormigas: 10, alpha: 50, beta: 10, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
3. Número de hormigas: 10, alpha: 100, beta: 2, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
4. Número de hormigas: 10, alpha: 100, beta: 10, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
5. Número de hormigas: 10, alpha: 200, beta: 2, Mejor camino: [2, 3, 1, 0, 4], Costo: 62
6. Número de hormigas: 10, alpha: 200, beta: 10, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
7. Número de hormigas: 25, alpha: 50, beta: 2, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
8. Número de hormigas: 25, alpha: 50, beta: 10, Mejor camino: [2, 3, 1, 0, 4], Costo: 62
9. Número de hormigas: 25, alpha: 100, beta: 2, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
10. Número de hormigas: 25, alpha: 100, beta: 10, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
11. Número de hormigas: 25, alpha: 200, beta: 2, Mejor camino: [2, 3, 1, 0, 4], Costo: 62
12. Número de hormigas: 25, alpha: 200, beta: 10, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
13. Número de hormigas: 50, alpha: 50, beta: 2, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
14. Número de hormigas: 50, alpha: 50, beta: 10, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
15. Número de hormigas: 50, alpha: 100, beta: 2, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
16. Número de hormigas: 50, alpha: 100, beta: 10, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
17. Número de hormigas: 50, alpha: 200, beta: 2, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
18. Número de hormigas: 50, alpha: 200, beta: 10, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
19. Número de hormigas: 100, alpha: 50, beta: 2, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
20. Número de hormigas: 100, alpha: 50, beta: 10, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
21. Número de hormigas: 100, alpha: 100, beta: 2, Mejor camino: [2, 3, 1, 0, 4], Costo: 62
22. Número de hormigas: 100, alpha: 100, beta: 10, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
23. Número de hormigas: 100, alpha: 200, beta: 2, Mejor camino: [0, 3, 1, 4, 2], Costo: 60
24. Número de hormigas: 100, alpha: 200, beta: 10, Mejor camino: [3, 2, 4, 0, 1], Costo: 62
DiegoDEMON Practica 5 main >_ pwsh |

```

Figura 1: Pruebas de regresión

```
PowerShell
1. Número de hormigas: 10, alpha: 50, beta: 2, Mejor camino: [3, 1, 6, 8, 11, 9, 0, 4, 2, 5, 7, 10], Costo: 276220
2. Número de hormigas: 10, alpha: 50, beta: 10, Mejor camino: [4, 6, 1, 10, 11, 8, 0, 9, 3, 7, 5, 2], Costo: 285030
3. Número de hormigas: 10, alpha: 100, beta: 2, Mejor camino: [3, 4, 5, 1, 6, 0, 8, 10, 2, 9, 11, 7], Costo: 261538
4. Número de hormigas: 10, alpha: 100, beta: 10, Mejor camino: [6, 8, 10, 9, 3, 7, 5, 0, 2, 11, 4, 1], Costo: 268724
5. Número de hormigas: 10, alpha: 200, beta: 2, Mejor camino: [1, 2, 6, 3, 7, 5, 11, 0, 4, 9, 8, 10], Costo: 257430
6. Número de hormigas: 10, alpha: 200, beta: 10, Mejor camino: [3, 4, 9, 5, 2, 6, 0, 10, 7, 11, 8, 1], Costo: 262076
7. Número de hormigas: 25, alpha: 50, beta: 2, Mejor camino: [3, 1, 6, 5, 2, 8, 11, 0, 9, 10, 4, 7], Costo: 276834
8. Número de hormigas: 25, alpha: 50, beta: 10, Mejor camino: [1, 6, 0, 11, 8, 2, 5, 7, 3, 10, 4, 9], Costo: 285490
9. Número de hormigas: 25, alpha: 100, beta: 2, Mejor camino: [7, 0, 4, 5, 10, 9, 2, 6, 11, 8, 1, 3], Costo: 258888
10. Número de hormigas: 25, alpha: 100, beta: 10, Mejor camino: [7, 8, 0, 10, 1, 6, 4, 9, 5, 11, 2, 3], Costo: 254928
11. Número de hormigas: 25, alpha: 200, beta: 2, Mejor camino: [10, 2, 11, 1, 6, 4, 7, 0, 3, 9, 5, 8], Costo: 264590
12. Número de hormigas: 25, alpha: 200, beta: 10, Mejor camino: [9, 5, 11, 1, 6, 7, 0, 4, 3, 10, 8, 2], Costo: 258672
13. Número de hormigas: 50, alpha: 50, beta: 2, Mejor camino: [7, 2, 5, 11, 8, 6, 1, 10, 3, 9, 0, 4], Costo: 272696
14. Número de hormigas: 50, alpha: 50, beta: 10, Mejor camino: [4, 2, 5, 11, 8, 6, 1, 10, 3, 9, 0, 7], Costo: 288172
15. Número de hormigas: 50, alpha: 100, beta: 2, Mejor camino: [3, 0, 1, 4, 5, 7, 6, 2, 9, 11, 8, 10], Costo: 254192
16. Número de hormigas: 50, alpha: 100, beta: 10, Mejor camino: [7, 0, 8, 1, 3, 9, 2, 5, 6, 4, 11, 10], Costo: 258896
17. Número de hormigas: 50, alpha: 200, beta: 2, Mejor camino: [3, 8, 1, 4, 7, 5, 2, 9, 6, 0, 11, 10], Costo: 247028
18. Número de hormigas: 50, alpha: 200, beta: 10, Mejor camino: [3, 11, 10, 7, 9, 0, 8, 4, 2, 5, 6, 1], Costo: 257900
19. Número de hormigas: 100, alpha: 50, beta: 2, Mejor camino: [8, 11, 0, 9, 3, 10, 1, 6, 4, 2, 5, 7], Costo: 266502
20. Número de hormigas: 100, alpha: 50, beta: 10, Mejor camino: [9, 0, 11, 8, 2, 5, 7, 3, 10, 1, 6, 4], Costo: 280354
21. Número de hormigas: 100, alpha: 100, beta: 2, Mejor camino: [8, 6, 2, 7, 4, 3, 9, 11, 1, 0, 10, 5], Costo: 257372
22. Número de hormigas: 100, alpha: 100, beta: 10, Mejor camino: [7, 9, 3, 6, 8, 2, 4, 10, 0, 1, 11, 5], Costo: 260692
23. Número de hormigas: 100, alpha: 200, beta: 2, Mejor camino: [3, 8, 1, 7, 5, 11, 6, 9, 4, 0, 2, 10], Costo: 259102
24. Número de hormigas: 100, alpha: 200, beta: 10, Mejor camino: [3, 8, 11, 4, 7, 5, 2, 9, 6, 1, 0, 10], Costo: 251410
DiegoDEMON > Practica 5 > main > .\pwhsh
```

Figura 2: Pruebas de regresión

```
PowerShell
1. Número de hormigas: 10, alpha: 50, beta: 2, Mejor camino: [1, 11, 3, 2, 13, 12, 6, 7, 4, 5, 8, 0, 9, 10, 14], Costo: 444856
2. Número de hormigas: 10, alpha: 50, beta: 10, Mejor camino: [13, 12, 8, 5, 14, 1, 3, 11, 0, 9, 6, 7, 4, 10, 2], Costo: 449958
3. Número de hormigas: 10, alpha: 100, beta: 2, Mejor camino: [11, 1, 3, 12, 8, 5, 14, 6, 7, 10, 4, 13, 9, 0, 2], Costo: 435068
4. Número de hormigas: 10, alpha: 100, beta: 10, Mejor camino: [14, 0, 13, 6, 4, 12, 1, 5, 7, 8, 11, 2, 9, 10, 3], Costo: 429336
5. Número de hormigas: 10, alpha: 200, beta: 2, Mejor camino: [0, 5, 10, 1, 7, 3, 11, 2, 6, 12, 13, 4, 14, 9, 8], Costo: 437720
6. Número de hormigas: 10, alpha: 200, beta: 10, Mejor camino: [5, 12, 2, 0, 6, 11, 1, 3, 13, 4, 7, 14, 8, 10, 9], Costo: 439522
7. Número de hormigas: 25, alpha: 50, beta: 2, Mejor camino: [12, 8, 5, 14, 1, 11, 6, 7, 4, 13, 9, 0, 10, 2, 3], Costo: 444252
8. Número de hormigas: 25, alpha: 50, beta: 10, Mejor camino: [13, 12, 8, 5, 14, 1, 3, 11, 0, 9, 6, 7, 4, 10, 2], Costo: 449958
9. Número de hormigas: 25, alpha: 100, beta: 2, Mejor camino: [11, 1, 3, 4, 5, 8, 12, 2, 10, 7, 6, 13, 9, 0, 14], Costo: 432860
10. Número de hormigas: 25, alpha: 100, beta: 10, Mejor camino: [11, 0, 5, 14, 12, 6, 1, 13, 8, 7, 10, 3, 4, 2, 9], Costo: 429770
11. Número de hormigas: 25, alpha: 200, beta: 2, Mejor camino: [12, 3, 6, 11, 5, 8, 13, 7, 0, 4, 9, 14, 1, 2, 10], Costo: 442624
12. Número de hormigas: 25, alpha: 200, beta: 10, Mejor camino: [3, 13, 7, 6, 14, 4, 8, 2, 10, 12, 5, 11, 0, 1, 9], Costo: 429152
13. Número de hormigas: 50, alpha: 50, beta: 2, Mejor camino: [5, 8, 12, 1, 3, 6, 7, 4, 13, 0, 9, 2, 14, 11, 10], Costo: 431848
14. Número de hormigas: 50, alpha: 50, beta: 10, Mejor camino: [2, 12, 8, 5, 14, 1, 3, 9, 0, 11, 6, 7, 4, 10, 13], Costo: 448974
15. Número de hormigas: 50, alpha: 100, beta: 2, Mejor camino: [3, 14, 8, 5, 0, 9, 1, 11, 12, 2, 6, 7, 10, 13, 4], Costo: 433988
16. Número de hormigas: 50, alpha: 100, beta: 10, Mejor camino: [6, 4, 8, 0, 10, 11, 3, 7, 14, 2, 5, 12, 13, 1, 9], Costo: 437000
17. Número de hormigas: 50, alpha: 200, beta: 2, Mejor camino: [8, 6, 7, 4, 13, 14, 3, 11, 10, 9, 1, 5, 2, 0, 12], Costo: 427902
18. Número de hormigas: 50, alpha: 200, beta: 10, Mejor camino: [9, 6, 11, 2, 12, 0, 7, 8, 3, 10, 1, 4, 14, 5, 13], Costo: 433926
19. Número de hormigas: 100, alpha: 50, beta: 2, Mejor camino: [11, 1, 3, 4, 5, 8, 12, 2, 10, 7, 6, 0, 9, 13, 14], Costo: 432598
20. Número de hormigas: 100, alpha: 50, beta: 10, Mejor camino: [14, 5, 8, 12, 13, 6, 7, 1, 3, 11, 0, 9, 2, 4, 10], Costo: 448290
21. Número de hormigas: 100, alpha: 100, beta: 2, Mejor camino: [11, 5, 0, 12, 13, 3, 1, 14, 8, 10, 4, 6, 9, 7, 2], Costo: 421446
22. Número de hormigas: 100, alpha: 100, beta: 10, Mejor camino: [8, 11, 6, 1, 10, 12, 13, 14, 0, 2, 9, 4, 7, 3, 5], Costo: 430164
23. Número de hormigas: 100, alpha: 200, beta: 2, Mejor camino: [8, 11, 5, 2, 10, 4, 0, 12, 7, 3, 1, 6, 9, 14, 13], Costo: 426164
24. Número de hormigas: 100, alpha: 200, beta: 10, Mejor camino: [5, 12, 13, 1, 6, 4, 14, 2, 7, 3, 9, 0, 11, 10, 8], Costo: 433846
DiegoDEMON > Practica 5 > main > .\pwhsh
```

Figura 3: Pruebas de regresión

```
PowerShell
1. Número de hormigas: 10, alpha: 50, beta: 2, Mejor camino: [20, 0, 24, 16, 10, 18, 7, 3, 13, 12, 28, 1, 17, 25, 9, 26, 22, 5, 21, 27, 8, 2, 15, 29, 11, 14, 6, 19, 23, 4], Costo: 2126814
2. Número de hormigas: 10, alpha: 50, beta: 10, Mejor camino: [24, 0, 16, 8, 2, 15, 29, 11, 16, 13, 12, 28, 23, 19, 17, 21, 5, 22, 26, 9, 25, 20, 18, 6, 27, 7, 5, 18, 1, 4], Costo: 211399
3. Número de hormigas: 10, alpha: 100, beta: 2, Mejor camino: [0, 17, 11, 5, 24, 27, 13, 8, 7, 22, 25, 29, 23, 12, 16, 15, 4, 26, 19, 28, 6, 9, 20, 14, 21, 10, 1, 2, 18, 3], Costo: 207385
4. Número de hormigas: 10, alpha: 100, beta: 10, Mejor camino: [16, 17, 7, 2, 29, 21, 9, 5, 8, 4, 22, 11, 20, 27, 15, 6, 25, 14, 13, 26, 10, 24, 12, 18, 19, 3, 28, 1, 0, 23], Costo: 20861
5. Número de hormigas: 10, alpha: 200, beta: 2, Mejor camino: [25, 13, 29, 11, 17, 18, 1, 24, 14, 22, 19, 12, 21, 3, 7, 2, 15, 23, 26, 20, 27, 5, 8, 16, 20, 18, 6, 4, 0, 9], Costo: 204684
6. Número de hormigas: 10, alpha: 200, beta: 10, Mejor camino: [24, 0, 9, 16, 5, 26, 11, 22, 13, 6, 28, 25, 2, 23, 10, 7, 4, 3, 17, 1, 18, 21, 20, 19, 15, 27, 12, 8, 14, 29], Costo: 20857
7. Número de hormigas: 25, alpha: 50, beta: 2, Mejor camino: [11, 2, 15, 29, 22, 5, 21, 20, 0, 16, 24, 9, 26, 18, 7, 3, 6, 19, 17, 14, 23, 1, 25, 4, 27, 8, 12, 28, 13, 10], Costo: 2189176
8. Número de hormigas: 25, alpha: 50, beta: 10, Mejor camino: [26, 9, 25, 1, 6, 10, 15, 29, 11, 18, 27, 7, 3, 13, 12, 28, 14, 2, 8, 16, 24, 0, 20, 17, 19, 23, 4, 22, 5, 21], Costo: 211312
9. Número de hormigas: 25, alpha: 100, beta: 2, Mejor camino: [15, 12, 6, 1, 28, 10, 4, 14, 7, 18, 23, 27, 20, 3, 0, 26, 8, 16, 9, 24, 25, 17, 5, 11, 22, 29, 21, 19, 13, 2], Costo: 209335
10. Número de hormigas: 25, alpha: 100, beta: 10, Mejor camino: [29, 27, 22, 21, 15, 16, 14, 4, 20, 0, 25, 13, 5, 10, 12, 28, 9, 8, 19, 3, 1, 17, 18, 23, 7, 24, 26, 2, 6, 11], Costo: 2076
11. Número de hormigas: 25, alpha: 200, beta: 2, Mejor camino: [8, 5, 2, 24, 3, 27, 9, 20, 23, 11, 17, 7, 15, 21, 28, 22, 12, 25, 1, 13, 4, 14, 19, 10, 6, 16, 26, 0, 29, 18], Costo: 20843
12. Número de hormigas: 25, alpha: 200, beta: 10, Mejor camino: [10, 28, 6, 25, 0, 15, 8, 12, 9, 13, 17, 29, 3, 20, 5, 2, 27, 7, 1, 21, 11, 22, 4, 18, 20, 10, 19, 23, 24, 14], Costo: 2089
13. Número de hormigas: 50, alpha: 50, beta: 2, Mejor camino: [18, 19, 23, 28, 12, 13, 14, 2, 8, 16, 24, 0, 20, 25, 9, 26, 22, 5, 21, 27, 7, 3, 6, 29, 11, 15, 10, 17, 1, 4], Costo: 211761
14. Número de hormigas: 50, alpha: 50, beta: 10, Mejor camino: [6, 0, 16, 24, 15, 29, 11, 5, 22, 26, 9, 25, 20, 13, 12, 28, 14, 2, 0, 27, 7, 3, 17, 19, 23, 4, 1, 18, 10, 21], Costo: 21007
15. Número de hormigas: 50, alpha: 100, beta: 2, Mejor camino: [16, 26, 23, 27, 15, 4, 1, 9, 2, 5, 25, 14, 6, 8, 11, 10, 7, 17, 13, 19, 3, 18, 20, 24, 21, 26, 0, 12, 29, 22], Costo: 20889
16. Número de hormigas: 50, alpha: 100, beta: 10, Mejor camino: [2, 20, 22, 18, 15, 25, 13, 19, 24, 7, 8, 5, 6, 17, 26, 28, 12, 3, 23, 16, 9, 29, 27, 14, 10, 1, 21, 0, 11, 4], Costo: 2069
17. Número de hormigas: 50, alpha: 200, beta: 2, Mejor camino: [3, 4, 11, 13, 27, 19, 10, 25, 20, 23, 5, 26, 17, 12, 14, 29, 22, 2, 7, 9, 15, 8, 28, 18, 24, 0, 21, 6, 16, 1], Costo: 20682
18. Número de hormigas: 50, alpha: 200, beta: 10, Mejor camino: [15, 27, 3, 14, 5, 25, 16, 19, 6, 12, 18, 7, 1, 13, 28, 9, 4, 8, 17, 24, 22, 0, 11, 23, 26, 10, 20, 2, 21, 29], Costo: 2093
19. Número de hormigas: 100, alpha: 50, beta: 2, Mejor camino: [6, 0, 24, 16, 19, 17, 21, 5, 22, 26, 9, 25, 4, 23, 8, 2, 15, 29, 7, 3, 20, 13, 14, 28, 12, 10, 27, 11, 1, 18], Costo: 21177
20. Número de hormigas: 100, alpha: 50, beta: 10, Mejor camino: [24, 0, 16, 8, 2, 15, 29, 11, 14, 13, 12, 28, 1, 23, 19, 17, 25, 9, 26, 22, 5, 21, 27, 7, 3, 6, 10, 20, 18, 4], Costo: 2082
21. Número de hormigas: 100, alpha: 100, beta: 2, Mejor camino: [10, 26, 6, 21, 17, 5, 12, 11, 13, 22, 28, 27, 14, 2, 4, 24, 0, 7, 19, 8, 3, 9, 1, 23, 16, 20, 18, 15, 29, 25], Costo: 2072
22. Número de hormigas: 100, alpha: 100, beta: 10, Mejor camino: [23, 16, 18, 20, 13, 12, 4, 11, 22, 9, 6, 0, 29, 2, 3, 21, 7, 8, 25, 5, 26, 10, 1, 27, 17, 14, 19, 15, 28, 24], Costo: 207
23. Número de hormigas: 100, alpha: 200, beta: 2, Mejor camino: [2, 19, 14, 12, 10, 4, 18, 20, 8, 15, 6, 28, 0, 22, 13, 11, 9, 7, 24, 3, 16, 17, 5, 25, 29, 1, 21, 23, 27, 26], Costo: 2065
24. Número de hormigas: 100, alpha: 200, beta: 10, Mejor camino: [24, 22, 12, 21, 1, 3, 4, 16, 23, 11, 28, 9, 26, 8, 13, 10, 19, 6, 25, 0, 15, 18, 14, 2, 29, 17, 27, 20, 5, 7], Costo: 208
DiegoDEMON > Practica 5 > main > .\pwhsh
```

Figura 4: Pruebas de regresión

### 3. Lectura

Problema de Optimización Combinatoria Resuelto por ACO

El Algoritmo de Optimización por Colonia de Hormigas (ACO) resuelve problemas de optimización combinatoria convirtiéndolos en problemas de encontrar el camino más corto en un grafo ponderado. Formalmente, esto se describe como:

- **Grafo Ponderado**  $G = (V, E)$ : donde  $V$  es el conjunto de vértices y  $E$  es el conjunto de aristas.
- **Función de Costo**  $c : E \rightarrow R^+$ : Asigna un costo positivo a cada arista.
- **Soluciones**  $S$ : Conjuntos de caminos en el grafo.
- **Función Objetivo**  $f : S \rightarrow R$ : Evaluación del costo de cada solución.

Las hormigas artificiales construyen soluciones explorando el grafo, guiadas por niveles de feromonas que reflejan la calidad de soluciones previas.

Cuadro Comparativo de Versiones de ACO

Versión ACO	Descripción
<b>Ant System (AS)</b>	Primera versión básica de ACO. Las hormigas depositan feromonas proporcionalmente a la calidad de la solución encontrada.
<b>Ant Colony System (ACS)</b>	Introduce la actualización local de feromonas y un mecanismo de exploración más agresivo para evitar la convergencia prematura.
<b>Max-Min Ant System (MMAS)</b>	Establece límites máximo y mínimo a los niveles de feromonas, ayudando a prevenir la convergencia en soluciones subóptimas.
<b>Rank-Based Ant System (RAS)</b>	Asigna diferentes cantidades de feromonas según el ranking de las soluciones, favoreciendo las mejores soluciones encontradas.
<b>Elitist Ant System (EAS)</b>	Similar a AS, pero con un sesgo adicional hacia las mejores soluciones almacenadas en una memoria global.
<b>Hybrid Ant System (HAS)</b>	Combina ACO con otros métodos de optimización como el recorrido simulado o algoritmos genéticos para mejorar la exploración y explotación del espacio de soluciones.

Cuadro 1: Comparación de versiones de ACO

El último congreso registrado es el "ANTS 2022 - International Conference on Swarm Intelligence". Los tópicos abordados incluyeron:

- Algoritmos bioinspirados
- Optimización multiobjetivo
- Aplicaciones industriales de ACO
- Modelado y análisis teórico de ACO

Las pláticas cubrieron avances recientes en teoría de feromonas, aplicaciones prácticas y nuevas variantes híbridas del algoritmo.

Aplicaciones Relevantes

Tipo de Aplicación	Descripción
<b>Mono-objetivo</b>	Resolución del problema del viajero, optimización de rutas en redes de telecomunicaciones, y diseño de circuitos electrónicos.
<b>Multiobjetivo</b>	Optimización de problemas de programación de la producción, planificación de horarios en universidades, y diseño de redes robustas.

Cuadro 2: Aplicaciones Relevantes de ACO

## 4. Conclusiones

Escamilla Resendiz Aldo: El algoritmo de optimización por colonia de hormigas es una técnica metaheurística que ha demostrado ser efectiva en la resolución de problemas de optimización combinatoria. A través de la simulación del comportamiento de las hormigas en la naturaleza, el ACO es capaz de encontrar soluciones de alta calidad en un tiempo razonable. Las diferentes versiones de ACO ofrecen una variedad de enfoques para abordar problemas específicos, y su flexibilidad permite adaptarse a una amplia gama de aplicaciones. En el contexto de la configuración de parámetros, ANOVA ayuda a identificar cuáles parámetros tienen un efecto significativo en el rendimiento del sistema o proceso bajo estudio. Esto se realiza dividiendo la variabilidad total de los datos en componentes atribuibles a diferentes fuentes de variación.

Yañez Martinez Marthon Leobardo: El ACO es una técnica de optimización poderosa que ha demostrado ser efectiva en una variedad de aplicaciones. Al simular el comportamiento de las hormigas en la naturaleza, el ACO es capaz de encontrar soluciones de alta calidad para problemas de optimización combinatoria. Las diferentes versiones de ACO ofrecen una variedad de enfoques para abordar problemas específicos, y su flexibilidad permite adaptarse a una amplia gama de aplicaciones.

Castillo Reyes Diego: El algoritmo de optimización por colonia de hormigas es una técnica metaheurística que ha demostrado ser efectiva en la resolución de problemas de optimización combinatoria. A través de la simulación del comportamiento de las hormigas en la naturaleza, el ACO es capaz de encontrar soluciones de alta calidad en un tiempo razonable. Las diferentes versiones de ACO ofrecen una variedad de enfoques para abordar problemas específicos, y su flexibilidad permite adaptarse a una amplia gama de aplicaciones.