ESCOM

Problema de la Coloración Mínima

Diego Castillo Reyes[a], **Marthon Leobardo Yañez Martinez**[a], **Aldo Escamilla Resendiz**[a] **y Muñoz González Eduardo**[a]

[a]*Investigadores en formación, ESCOM, IPN*

Dra. Miriam Pescador Rojas

Resumen—En este trabajo se presenta el problema de la coloración mínima, el cual es un problema NP-completo. Se propone una solución basada en algoritmos genéticos para encontrar la coloración mínima de un grafo.

Keywords—*Coloración Mínima, Genéticos, Algoritmos, Grafos*

1. Introducción

El problema de la coloración mínima es un problema NP-completo, el cual consiste en asignar un color a cada vértice de un grafo de tal manera que dos vértices adyacentes no tengan el mismo color. El objetivo es encontrar la coloración mínima, es decir, la menor cantidad de colores posibles para colorear el grafo. Este problema es de gran importancia en la teoría de grafos, ya que tiene aplicaciones en la asignación de horarios, asignación de frecuencias, asignación de canales, entre otros. En este trabajo se propone una solución basada en PSO, un algoritmo de optimización basado en la inteligencia de enjambre, para encontrar la coloración mínima de un grafo.

2. Antecedentes

En una pequeña ciudad de Rusia, Königsberg (Actualmente Kaliningrado, Rusia), existen siete puentes que conectan cuatro islas. Un matemático llamado Leonhard Euler (1707-1783) se preguntó si era posible recorrer todos los puentes una sola vez y regresar al punto de partida. Euler demostró en 1736 que no era posible en su publicación *Solutio problematis ad geometriam situs pertinentis* [1], y para ello utilizó un grafo para representar las islas y los puentes como se ve en la Figura 1.

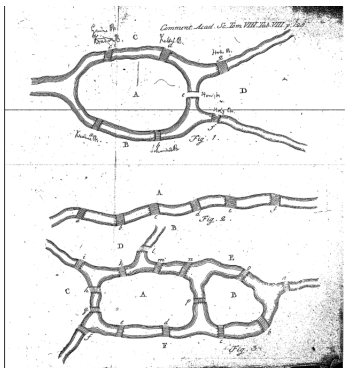

Figura 1. Grafo de Königsberg

Él demostró utilizando cada puente como una arista y cada isla como un vértice, que no era posible recorrer todos los puentes una sola vez y regresar al punto de partida. Siendo este el primer problema de teoría de grafos, el cual es un área de las matemáticas que estudia las relaciones entre los vértices y las aristas de un grafo. Un grafo es un conjunto de vértices y aristas, donde las aristas son pares no ordenados de vértices. Posteriormente, en 1852, Francis Guthrie planteó el problema de los cuatro colores, el cual consiste en colorear un mapa de tal manera que dos regiones adyacentes no tengan el mismo color. Este problema fue resuelto en 1976 por Kenneth Appel y Wolfgang Haken utilizando una computadora, demostrando que cuatro colores son suficientes para colorear cualquier mapa [2]. A partir de este problema, se han planteado diversos problemas de coloración, entre ellos el problema de la coloración mínima, el cual es un problema NP-completo.

3. Metodología

Para poder resolver el problema de la coloración mínima, se propone una solución basada en algoritmos genéticos. El algoritmo genético elegido fue el de PSO (Particle Swarm Optimization), el cual es un algoritmo de optimización basado en la inteligencia de enjambre. El algoritmo PSO se basa en el comportamiento social de las partículas, las cuales se mueven en un espacio de búsqueda en busca de la mejor solución. Las partículas tienen mejoras cognitivas y mejoras sociales, las cuales les permiten moverse en el espacio de búsqueda. En el caso del problema de la coloración mínima, las partículas representan una coloración de un grafo, y el objetivo es encontrar la coloración mínima. La representación de las partículas es un vector de enteros que representa los colores de los vértices del grafo. El algoritmo PSO se encarga de mover las partículas en el espacio de búsqueda, de tal manera que se encuentre la coloración mínima del grafo. Para evaluar la calidad de una coloración, se utiliza la función objetivo, la cual es el número de colores utilizados en la coloración.

4. Propuesta de solución

```python
import numpy as np
import networkx as nx
import random
import matplotlib.pyplot as plt
import itertools

# Funcion de fitness
def fitness(solution, graph):
    conflicts = 0
    for edge in graph.edges:
        if solution[edge[0]] == solution[edge[1]]:
            conflicts += 1
    num_colors = len(set(solution))
    return num_colors + conflicts * 1000

def ANOVA(num_particles, w, c1, c2, num_combinations=5):
    combinaciones = list(itertools.product(num_particles, w, c1, c2))
    random.shuffle(combinaciones)  # Mezcla las combinaciones
    return combinaciones[:num_combinations]  # Devuelve las primeras 'num_combinations' combinaciones

# PSO parameters
num_particles = [30, 50, 100]
num_nodos = 10 # Numero de nodos
max_iter = 100

# Parametros de PSO
w = [0.3, 0.5, 0.8]  # Inercia
c1 = [1.0, 1.5, 2.0]  # Constante cognitiva
c2 = [1.0, 1.5, 2.0]  # Constante social

# Probabilidad de conexion
p = 0.5

# Crear el grafo
graph = nx.erdos_renyi_graph(num_nodos, p)

# Asegurarse de que el grafo no sea totalmente conexo
while nx.is_connected(graph):
    graph = nx.erdos_renyi_graph(num_nodos, p)
```

```python

# Dibujar el grafo
plt.figure(figsize=(8, 6))
nx.draw(graph, with_labels=True, node_color='skyblue', node_size=500, edge_color='gray')
plt.show()

# Generar las combinaciones de parametros
parametros = ANOVA(num_particles, w, c1, c2, num_combinations=5)

for i, (num_particles, w, c1, c2) in enumerate(parametros):
    print(f"Experimento {i + 1}: num_particles={num_particles}, w={w}, c1={c1}, c2={c2}")

    # Inicializacion de particulas
    particles = [np.random.randint(0, num_nodos, num_nodos) for _ in range(num_particles)]
    velocities = [np.random.randint(-1, 2, num_nodos) for _ in range(num_particles)]
    pbest_positions = particles.copy()
    pbest_scores = [fitness(p, graph) for p in particles]
    gbest_position = pbest_positions[np.argmin(pbest_scores)]
    gbest_score = min(pbest_scores)

    # PSO main loop
    for iteration in range(max_iter):
        for j in range(num_particles):
            velocities[j] = (w * velocities[j]
                             + c1 * random.random() * (pbest_positions[j] - particles[j])
                             + c2 * random.random() * (gbest_position - particles[j]))

            # Actualiza la posicion de las particulas
            particles[j] = np.clip(particles[j] + velocities[j], 0, num_nodos - 1).astype(int)

            # Evaluar nueva posicion
            current_fitness = fitness(particles[j], graph)

            # Actualizar pbest
            if current_fitness < pbest_scores[j]:
                pbest_positions[j] = particles[j]
                pbest_scores[j] = current_fitness

            # Actualizar gbest
            if current_fitness < gbest_score:
                gbest_position = particles[j]
                gbest_score = current_fitness

        print(f"Iteracion {iteration + 1}/{max_iter}, Mejor Fitness: {gbest_score}")

    # Resultado final
    print("Mejor solucion encontrada:", gbest_position)
    print("Numero de colores utilizados:", len(set(gbest_position)))
    print("\n\n")

    # Dibujar el grafo final coloreado para cada experimento
    color_map = [f"C{color}" for color in gbest_position]  # Asigna un color a cada nodo segun la solucipn optima
    plt.figure(figsize=(8, 6))
    nx.draw(graph, with_labels=True, node_color=color_map, node_size=500, edge_color='gray')
    plt.show()
```

Código 1. Algoritmo PSO para el problema de la coloración mínima

5. Resultados

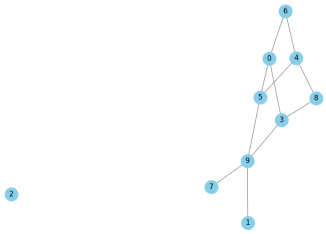

Figura 2. Grafo de 10 nodos

En la Figura 2 se muestra un grafo de 10 nodos que no es totalmente conexo, el cual se utilizó para evaluar el algoritmo PSO propuesto.

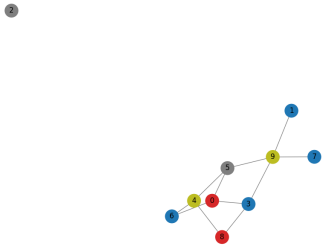

Figura 3. Coloración del grafo de 10 nodos

En la Figura 3 se muestra el resultado de la coloración del grafo de 10 nodos utilizando el algoritmo PSO propuesto. Como se puede observar, se utilizan 4 colores para colorear el grafo.
En la salida de la terminal obtuvimos lo siguiente:

```
Iteracion 1/100, Mejor Fitness: 6
Iteracion 2/100, Mejor Fitness: 5
Iteracion 3/100, Mejor Fitness: 5
Iteracion 4/100, Mejor Fitness: 5
Iteracion 5/100, Mejor Fitness: 5
Iteracion 6/100, Mejor Fitness: 5
Iteracion 7/100, Mejor Fitness: 4
Iteracion 8/100, Mejor Fitness: 4
Iteracion 9/100, Mejor Fitness: 4
Iteracion 10/100, Mejor Fitness: 4
Iteracion 11/100, Mejor Fitness: 4
Iteracion 12/100, Mejor Fitness: 4
Iteracion 13/100, Mejor Fitness: 4
Iteracion 14/100, Mejor Fitness: 4
Iteracion 15/100, Mejor Fitness: 4
Iteracion 16/100, Mejor Fitness: 4
Iteracion 17/100, Mejor Fitness: 4
Iteracion 18/100, Mejor Fitness: 4
Iteracion 19/100, Mejor Fitness: 4
Iteracion 20/100, Mejor Fitness: 4
Iteracion 21/100, Mejor Fitness: 4
Iteracion 22/100, Mejor Fitness: 4
Iteracion 23/100, Mejor Fitness: 4
Iteracion 24/100, Mejor Fitness: 4
Iteracion 25/100, Mejor Fitness: 4
Iteracion 26/100, Mejor Fitness: 4
Iteracion 27/100, Mejor Fitness: 4
Iteracion 28/100, Mejor Fitness: 4
Iteracion 29/100, Mejor Fitness: 4
Iteracion 30/100, Mejor Fitness: 4
Iteracion 31/100, Mejor Fitness: 4
Iteracion 32/100, Mejor Fitness: 4
Iteracion 33/100, Mejor Fitness: 4
Iteracion 34/100, Mejor Fitness: 4
Iteracion 35/100, Mejor Fitness: 4
Iteracion 36/100, Mejor Fitness: 4
Iteracion 37/100, Mejor Fitness: 4
Iteracion 38/100, Mejor Fitness: 4
Iteracion 39/100, Mejor Fitness: 4
```

```
Iteracion 40/100, Mejor Fitness: 4
Iteracion 41/100, Mejor Fitness: 4
Iteracion 42/100, Mejor Fitness: 4
Iteracion 43/100, Mejor Fitness: 4
Iteracion 44/100, Mejor Fitness: 4
Iteracion 45/100, Mejor Fitness: 4
Iteracion 46/100, Mejor Fitness: 4
Iteracion 47/100, Mejor Fitness: 4
Iteracion 48/100, Mejor Fitness: 4
Iteracion 49/100, Mejor Fitness: 4
Iteracion 50/100, Mejor Fitness: 4
Iteracion 51/100, Mejor Fitness: 4
Iteracion 52/100, Mejor Fitness: 4
Iteracion 53/100, Mejor Fitness: 4
Iteracion 54/100, Mejor Fitness: 4
Iteracion 55/100, Mejor Fitness: 4
Iteracion 56/100, Mejor Fitness: 4
Iteracion 57/100, Mejor Fitness: 4
Iteracion 58/100, Mejor Fitness: 4
Iteracion 59/100, Mejor Fitness: 4
Iteracion 60/100, Mejor Fitness: 4
Iteracion 61/100, Mejor Fitness: 4
Iteracion 62/100, Mejor Fitness: 4
Iteracion 63/100, Mejor Fitness: 4
Iteracion 64/100, Mejor Fitness: 4
Iteracion 65/100, Mejor Fitness: 4
Iteracion 66/100, Mejor Fitness: 4
Iteracion 67/100, Mejor Fitness: 4
Iteracion 68/100, Mejor Fitness: 4
Iteracion 69/100, Mejor Fitness: 4
Iteracion 70/100, Mejor Fitness: 4
Iteracion 71/100, Mejor Fitness: 4
Iteracion 72/100, Mejor Fitness: 4
Iteracion 73/100, Mejor Fitness: 4
Iteracion 74/100, Mejor Fitness: 4
Iteracion 75/100, Mejor Fitness: 4
Iteracion 76/100, Mejor Fitness: 4
Iteracion 77/100, Mejor Fitness: 4
Iteracion 78/100, Mejor Fitness: 4
Iteracion 79/100, Mejor Fitness: 4
Iteracion 80/100, Mejor Fitness: 4
Iteracion 81/100, Mejor Fitness: 4
Iteracion 82/100, Mejor Fitness: 4
Iteracion 83/100, Mejor Fitness: 4
Iteracion 84/100, Mejor Fitness: 4
Iteracion 85/100, Mejor Fitness: 4
Iteracion 86/100, Mejor Fitness: 4
Iteracion 87/100, Mejor Fitness: 4
Iteracion 88/100, Mejor Fitness: 4
Iteracion 89/100, Mejor Fitness: 4
Iteracion 90/100, Mejor Fitness: 4
Iteracion 91/100, Mejor Fitness: 4
Iteracion 92/100, Mejor Fitness: 4
Iteracion 93/100, Mejor Fitness: 4
Iteracion 94/100, Mejor Fitness: 4
Iteracion 95/100, Mejor Fitness: 4
Iteracion 96/100, Mejor Fitness: 4
Iteracion 97/100, Mejor Fitness: 4
Iteracion 98/100, Mejor Fitness: 4
Iteracion 99/100, Mejor Fitness: 4
Iteracion 100/100, Mejor Fitness: 4
Mejor solucion encontrada: [3 0 7 0 8 7 0 0 3 8]
Numero de colores utilizados: 4
```

Código 2. Salida del algoritmo PSO

En esta salida podemos apreciar la rapidez con la que converge el algoritmo de PSO para encontrar la coloración mínima del grafo de 10 nodos.

6. Conclusiones

En conclusión, la coloración minima de grafos es un problema fundamental en la teoría de grafos y gracias a sus amplias aplicaciones como la asignación de horarios, asignación de frecuencias, asignación de canales, entre otros. Mediante la solución usando PSO hemos logrado solucionar el problema pasando de un tiempo NP a un tiempo polinomial.

Referencias

[1] L. Euler, «Solutio problematis ad geometriam situs pertinentis,» *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, vol. 8, págs. 128-140, 1736.

[2] K. Appel y W. Haken, *Every Planar Map Is Four Colorable* (Contemporary Mathematics). Providence, RI: American Mathematical Society, 1976, vol. 98.