Análisis de Algoritmos
Proyecto

Carlos Alexis Barrios Bello
zs23000636@estudiantes.uv.mx

Maestría en Inteligencia Artificial

IIIA Instituto de Investigaciones en Inteligencia Artificial
Universidad Veracruzana
Campus Sur, Calle Paseo Lote II, Sección 2a, No 112
Nuevo Xalapa, Xalapa, Ver., México 91097

4 de enero de 2024

1. Introducción

A lo largo del tiempo, se han desarrollado nuevos algoritmos que pueda cumplir con su meta y es encontrar una solución óptima a un problema, en este caso se hará uso del algoritmo del recocido simulado para hallar la mejor ruta en el trayecto de atravesar todas las capitales de la República Mexicana y CDMX, es decir, aplicar este algoritmo al problema del viajero, un problema muy conocido en IA por su sencillez y facilidad de explicar en muchos algoritmos.

2. Marco Teórico

2.1. Simulado Recocido

Este algoritmo es muy conocido por estar optimizado de manera aleatoria. En este algoritmo, se aumenta la exploración a una solución global, no tanto una local.
Si etimología nace de la metalurgia, en donde se caliente un metal a muy alta temperatura y luego se deja enfriar, así repitiendo este proceso para que el metal esté más fuerte. De manera análoga, pasa lo mismo con este

algoritmo, de manera aleatoria se intenta llegar a una solución óptima, y para llegar a ella se tiene que alejar para buscar todas las respuestas posibles (como nuevas alternativas) y después acercarse a la respuesta óptima.
Este algoritmo sigue lo siguiente Russell and Norvig (2022):

Algorithm 1 *SimulatedAnnealing*

function SIMULATED-ANNEALING(*problem*, *schedule*) **return** a solution state.
    current ← problem.INITIAL.
    for $t = 1$ **to** ∞ **do**
        T ← schedule(t)
        if $T = 0$ **then return** current
            next ← a randomly selected sucessor of current
            ΔE ← VALUE(current) - VALUE(next)
            if $\Delta E > 0$ **then** current ← next
            elsecurrent ← next only with probability $e^{-\Delta E/T}$
            end if
        end if
    end for
end function

2.2. Problema del viajero

Este consiste en encontrar el camino más corto a través de un conjunto de N ciudades, donde cada ciudad es visitada exactamente una vez y se regresa a la ciudad inicial.
Este problema se puede definir como un circuito Hamiltoniano, según Ríos (2013), donde éste es un ciclo que pasa por todos los vértices del grafo exactamente una vez. Se puede definir como una secuencia de $n + 1$ vértices adyacentes $\nu_{i0}, \nu_{i1}, ..., \nu_{in-1}$ donde el primer vértice de la secuencia es el mismo que el último, y todos los otros $n - 1$ vértices son distintos.
Para solucionar este problema, Ríos (2013) sugiere usar el enfoque de genera y prueba, el cuál es un método de solución a problemas combinatorios. Sugiere generar cada uno de los elementos del dominio de un problema, seleccionar aquellos que satisfacen las restricciones del problema, y encontrar un elemento deseado. Y para obtener todos los viajes, se puede generando todas las permutaciones de las $n-1$ ciudades intermedias, luego calcular las longitudes de los viajes y encontrar el viaje más corto (como lo sugiere el recocido simulado).

3. Actividad

En esta sección se encargó lo siguiente:

A partir de las coordenadas de las ciudades de México, en cada ejecución del algoritmo:

1. Dar el valor del número de ciudades que se van a utilizar (N).
2. Hacer una selección aleatoria de N ciudades de México.
3. Resolver el problema del viajero para ese conjunto de ciudades empleando el algoritmo de recocido simulado.
4. Para N pequeña (menor que 10) comparar la solución obtenida por recocido simulado (ds) con la solución óptima (d^*) que brinda el algoritmo de fuerza bruta que genera las $N!$ posibilidades. La comparación se realizará en términos del error de la distancia ($e = d^* - ds$). Mostrar también si los recorridos obtenidos son distintos. Comparar así mismo en tiempo de ejecución ΔT.
5. Probar con al menos 3 valores distintos de N, y donde uno de esos valores sea la totalidad de las ciudades capitales de México.
6. Mostrar de manera gráfica la solución final obtenida en cada caso.

4. Resultados

Para lograr todos los puntos de la actividad, se hizo uso del lenguaje de programación *python*.

1. Se empieza importando todos los módulos necesarios para ejecutar el código:

```
import numpy as np
import itertools
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import time
```

```
from scipy.spatial.distance import euclidean
```

2. Ahora se sigue definiendo las distancias, el total de distancia y el algoritmo de fuerza bruta:

```
def distance(city1, city2):
    return euclidean(city1, city2)

def total_distance(path, cities):
    dist = 0
    for i in range(len(path) - 1):
        dist += distance(cities[path[i]], cities[path[i + 1]])
    # Distancia de regreso a la ciudad inicial
    dist += distance(cities[path[-1]], cities[path[0]])
    return dist

def brute_force(cities):
    # Generar todas las posibles combinaciones de ciudades
    all_paths = list(itertools.permutations(range(len(cities))))

    min_distance = float('inf')
    best_path = None

    for path in all_paths:
        current_distance = total_distance(path, cities)
        if current_distance < min_distance:
            min_distance = current_distance
            best_path = path

    return best_path, min_distance
```

3. Se define el simulado recocido:

```
def simulated_annealing(cities, initial_temperature=1000,
cooling_rate=0.003, num_iterations=10000):
    num_cities = len(cities)
    current_path = np.random.permutation(num_cities)
    current_distance = total_distance(current_path, cities)

    best_path = np.copy(current_path)
```

```
    best_distance = current_distance

    temperature = initial_temperature

    for _ in range(num_iterations):
        new_path = np.copy(current_path)
        # Intercambiar dos ciudades aleatorias
        idx1, idx2 = np.random.choice(num_cities, 2, replace=False)
        new_path[idx1], new_path[idx2] = new_path[idx2], new_path[idx1]

        new_distance = total_distance(new_path, cities)

        # Decidir si se acepta el nuevo camino
        if new_distance < current_distance or np.random.rand()
        < np.exp((current_distance - new_distance) / temperature):
            current_path = np.copy(new_path)
            current_distance = new_distance

        # Actualizar la mejor solución
        if current_distance < best_distance:
            best_path = np.copy(current_path)
            best_distance = current_distance

        # Enfriar el sistema
        temperature *= 1 - cooling_rate

    return best_path, best_distance
```

4. Y por último se define las coordenadas de cada capital de los estados de la República Mexicana, estas coordenadas fueron sacadas de geodatos y otros se corroboraron con Wikipedia:

```
cities_mexico = np.array([
[19.4326, -99.1332], #CDMX
[21.8853, -102.2916], #Aguascalientes
[24.1426, -110.3128], #La Paz
[19.8454, -90.523], #Campeche
[25.6866, -100.3161], #Saltillo
[19.2433, -103.725], #Colima
[16.7573, -93.1292], #Tuxtla Gutiérrez
```

```
[28.6353, -106.0889], #Chihuahua
[24.0277, -104.6532], #Durango
[21.1619, -101.6299], #Guanajuato
[17.5509, -99.505], #Chilpancigo
[20.1011, -98.7591], #Pachuca
[20.6597, -103.3496], #Guadalajara
[19.2826, -99.655], #Toluca
[21.5006, -104.876], #Tepic
[25.6866, -100.3161], #Monterrey
[17.0732, -96.7266], #Oaxaca
[19.0414, -98.2063], #Puebla
[20.5888, -100.3899], #Querétaro
[18.502, -88.3054], #Chetumal
[22.1566, -100.9851], #San Luis Potosí
[24.805, -107.396], #Culiacán
[29.0892, -110.961], #Hermosillo
[17.9895, -92.9288], #Villahermosa
[23.7369, -99.141], #Ciudad Victoria
[19.3189, -98.2375], #Tlaxcala
[19.5138, -96.9102], #Xalapa
[20.967, -89.6237], #Mérida
[22.768, -102.5814], #Zacatecas
[32.6245, -115.4523], #Mexicali
[19.7007, -99.2216], #Morelia corrección
[18.9261,-99.2307] #Cuernavaca correción
])
```

Por último se procede a mostrar los resultados como lo pide la actividad, el primero es la elección aleatoria den $N = 3$ (ciudades), en éstas se mostrará las ciudades escogidas de manera aleatoria, sus distancias recorridas por fuerza bruta y el recocido simulado, si hay diferencia entre el recorrido y sus tiempos de ejecución:

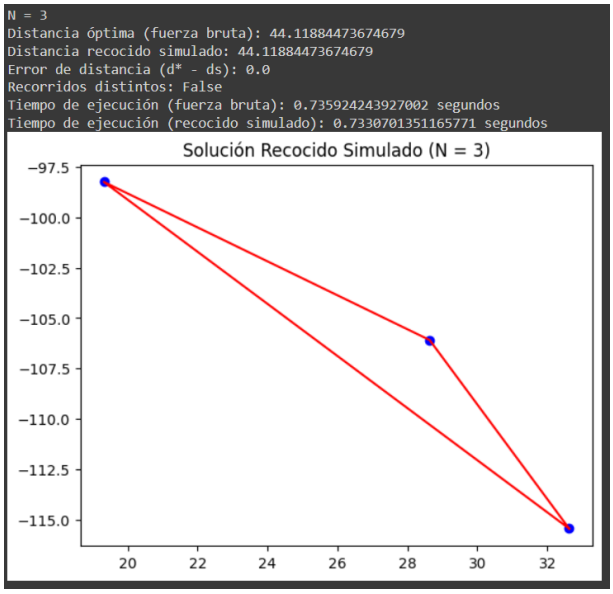

Figura 1: Gráfica para escoger tres ciudades de manera aleatoria.

Sigue la gráfica con $N = 5$:

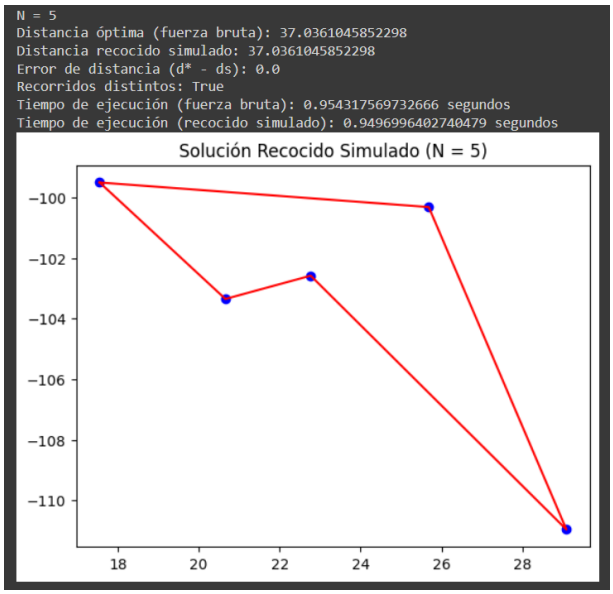

Figura 2: Gráfica para escoger cinco ciudades de manera aleatoria.

Ahora la gráfica con $N = 8$:

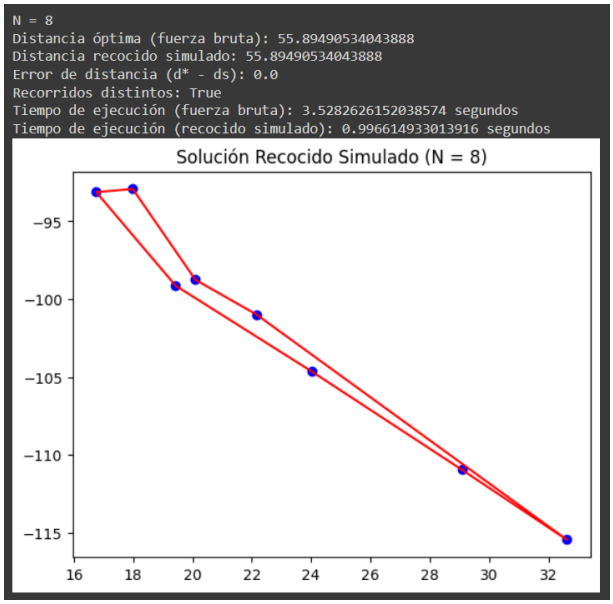

Figura 3: Gráfica para escoger ocho ciudades de manera aleatoria.

Y por último, la gráfica con $N = 11$:

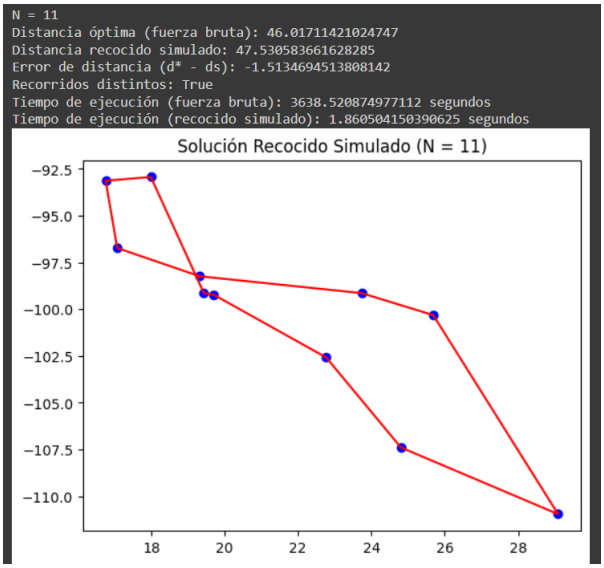

Figura 4: Gráfica para escoger once ciudades de manera aleatoria.

Como última gráfica, se hizo el plot de las 31 capitales y la CDMX, sin embargo tuvo un problema, y es que como agarra las ciudades de manera aleatoria, cambia la trayectoria óptima, dando resultados no tan óptimos, pero sí se pudo obtener la ruta óptima y será la última imagen:

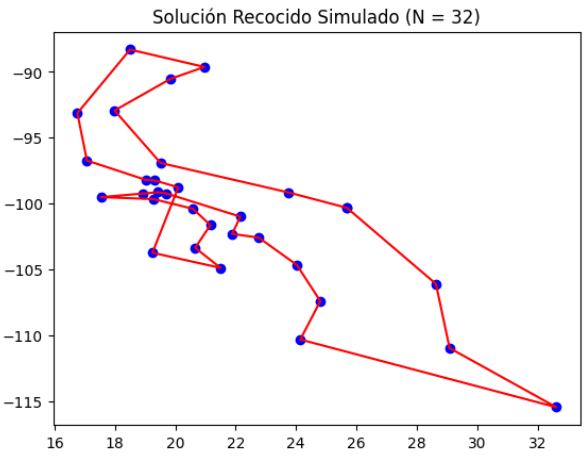

Figura 5: Gráfica para escoger todas las ciudades de manera aleatoria, primer intento.

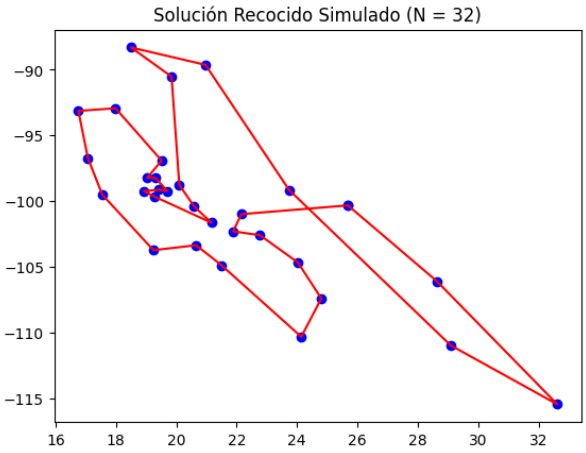

Figura 6: Gráfica para escoger todas las ciudades de manera aleatoria, segundo intento.

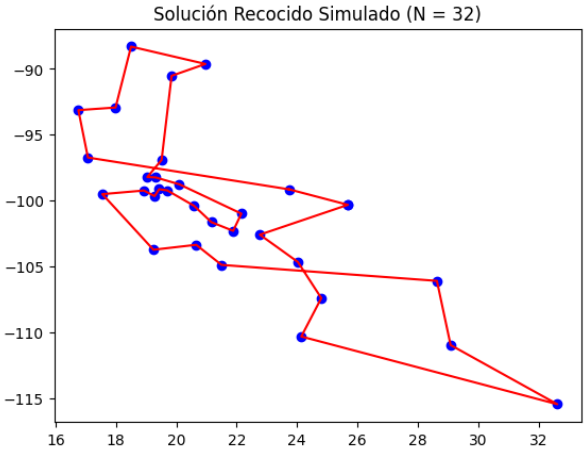

Figura 7: Gráfica para escoger todas las ciudades de manera aleatoria, tercer intento.

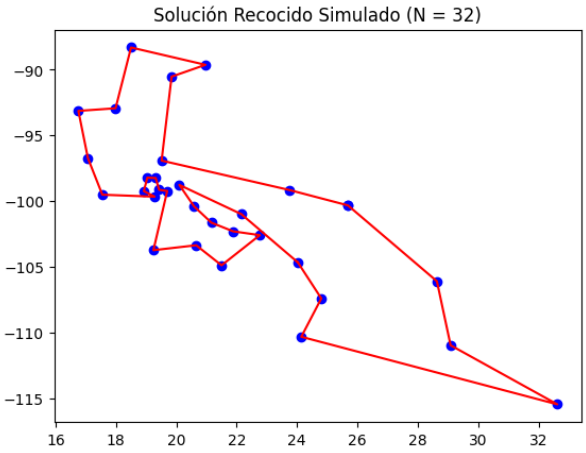

Figura 8: Gráfica para escoger todas las ciudades de manera aleatoria, ruta óptima (casi).

Esta última gráfica nos arroja todas las coordenadas de las capitales de la República Mexicana, pero están un poco extrañas y esto es debido a sus coordenadas negativas. Arroja la ubicaciones de cabeza, también considerando la curvatura del planeta, lo ideal sería una geodésica para expresar esto y no líneas rectas. Siendo la República esta:

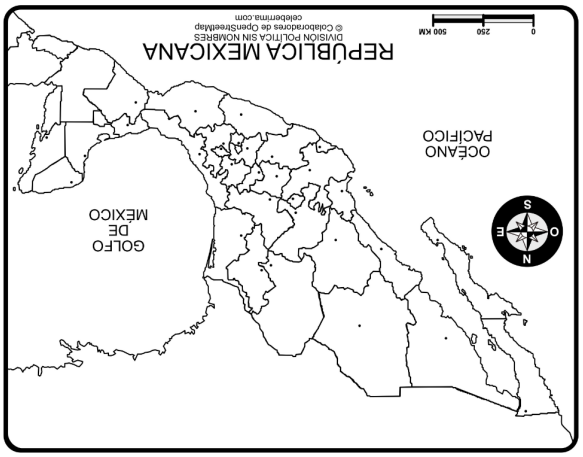

Figura 9: República Mexicana volteada $-180°$

5. Conclusiones

A lo largo de este proyecto se pudo conocer las generalidades del algoritmo "Recocido Simulado" el cual está optimizado para encontrar una solución global de manera aleatoria, también se conoció las generalidades del problema del viajero.
Se cubrió cada uno de los puntos propuestos por la actividad, dando un resultado esperado, pues, se esperaba que el recocido simulado fuera mejor que el método de fuerza bruta para encontrar la mejor ruta para cruzar cada una de las capitales de la República Mexicana, esto se puede observar mejor en la figura 4 donde hay una diferencia abrumadora para encontrar la mejor ruta entre estos dos métodos. Por ello, no se pudo sacar las mismas generalidades para N mayores a 11, el tiempo de ejecución crecía de una manera abrupta, se hubiera llevados bastantes días para encontrar esa solución en el método de fuerza bruta, riesgo que no se quiso tomar por el bien del equipo de cómputo.

Referencias

Russell, S. and Norvig, P. (2022). *Artificial Intelligence: A Modern Approach*. Fourth edition.

Ríos, Homero V., M. F. M. C. V. R. (2013). *Análisis de Algoritmos*. Textos Universitarios, Universidad Veracruzana, México, primera edition.