Facultad de Ingeniería de Sistemas e Informática
E.A.P de Ingeniería de Sistemas e Informática

TRABAJO GRUPAL

GRUPO N°3

Curso: Investigación Operativa
Docente: Cubas Becerra, Richard Javier

Integrantes:

Aldana Chipana, Mauricio	[22200164]
Castillo Reupo, John	[22200117]
Escribas Alan, Daniel	[22200057]
Espíritu Unsihuay, Erika Milagros	[22200170]
Valdiviezo Goicochea, Wisner	[22200217]

Lima, Perú
2024

Contents

1 Introducción al Templado Simulado

El templado simulado es un algoritmo de optimización inspirado en el proceso de enfriamiento de metales. Este método es utilizado para encontrar soluciones aproximadas a problemas de optimización complejos.

2 Funcionamiento del Algoritmo

El algoritmo de templado simulado se basa en la analogía con el enfriamiento de metales. A continuación se presenta un modelo matemático que describe su funcionamiento:

$$P(E) = \exp\left(-\frac{\Delta E}{kT}\right) \tag{1}$$

Donde:

- $P(E)$ es la probabilidad de aceptar una solución peor.
- ΔE es el cambio en la energía (o costo) de la solución.
- k es una constante de Boltzmann.
- T es la temperatura.

El algoritmo comienza con una temperatura alta que se va reduciendo gradualmente. En cada iteración, se generan soluciones vecinas y se aceptan o rechazan basándose en la probabilidad $P(E)$.

3 Parámetros del Algoritmo

Los parámetros clave del algoritmo de templado simulado incluyen:

- Temperatura inicial (T_0)
- Tasa de enfriamiento (α)
- Número de iteraciones por temperatura

Estos parámetros deben ser ajustados cuidadosamente para obtener un buen rendimiento del algoritmo.

4 Ventajas y Limitaciones

4.1 Ventajas

- Capacidad para escapar de óptimos locales.
- Flexibilidad para ser aplicado a una amplia variedad de problemas.

4.2 Limitaciones

- Sensibilidad a la elección de parámetros.
- Puede ser computacionalmente costoso para problemas muy grandes.

5 Caso Aplicativo

Para ilustrar el uso del templado simulado, consideremos un problema de optimización en la investigación operativa, como la optimización de rutas de transporte. El objetivo es encontrar rutas eficientes que minimicen el costo total de transporte, considerando restricciones como la capacidad de los vehículos y las ventanas de tiempo de entrega.

El problema de optimización de rutas de transporte (Vehicle Routing Problem, VRP) es un problema clásico en la investigación operativa. En este problema, un conjunto de vehículos debe recoger y entregar bienes a un conjunto de clientes, minimizando el costo total de transporte. Las restricciones incluyen la capacidad de los vehículos y las ventanas de tiempo en las que las entregas deben realizarse.

El algoritmo de templado simulado puede ser utilizado para encontrar soluciones aproximadas a este problema. A continuación, se presenta una implementación en Python del algoritmo de templado simulado para resolver el VRP.

Implementación en Python

```
import math
import random

#Funcion de costo (distancia total de la ruta)
def calculate_cost(route, distance_matrix):
    cost = 0
    for i in range(len(route) - 1):
        cost += distance_matrix[route[i]][route[i + 1]]
    cost += distance_matrix[route[-1]][route[0]]  # Regresar al
        punto de inicio
    return cost

# Generar una solucion vecina
def generate_neighbor(route):
    new_route = route[:]
    i, j = random.sample(range(len(route)), 2)
    new_route[i], new_route[j] = new_route[j], new_route[i]
    return new_route

# Algoritmo de templado simulado
def simulated_annealing(distance_matrix, initial_temp, cooling_rate
    , num_iterations):
    current_route = list(range(len(distance_matrix)))
    random.shuffle(current_route)
    current_cost = calculate_cost(current_route, distance_matrix)
    best_route = current_route[:]
    best_cost = current_cost
    temperature = initial_temp
```

```
27
28      for _ in range(num_iterations):
29          new_route = generate_neighbor(current_route)
30          new_cost = calculate_cost(new_route, distance_matrix)
31          delta_cost = new_cost - current_cost
32
33          if delta_cost < 0 or random.random() < math.exp(-delta_cost
                / temperature):
34              current_route = new_route
35              current_cost = new_cost
36
37              if current_cost < best_cost:
38                  best_route = current_route
39                  best_cost = current_cost
40
41          temperature *= cooling_rate
42
43      return best_route, best_cost
44
45  # Ejemplo de uso
46  distance_matrix = [
47      [0, 2, 9, 10],
48      [1, 0, 6, 4],
49      [15, 7, 0, 8],
50      [6, 3, 12, 0]
51  ]
52
53  initial_temp = 1000
54  cooling_rate = 0.99
55  num_iterations = 10000
56
57  best_route, best_cost = simulated_annealing(distance_matrix,
        initial_temp, cooling_rate, num_iterations)
58  print("Mejor ruta:", best_route)
59  print("Costo de la mejor ruta:", best_cost)
```

Este código proporciona una base para implementar el algoritmo de templado simulado en Python. Puedes adaptarlo y expandirlo según tus necesidades específicas.

5.1 El Problema del Viajante de Comercio (Traveling Salesman Problem, TSP)

¿En qué consiste?

El problema del Viajante de Comercio es un clásico problema de optimización combinatoria que se define así:

Dado un conjunto de n ciudades y las distancias entre cada par de ciudades, el objetivo es encontrar la ruta más corta que:

- Visite cada ciudad exactamente una vez.
- Regrese a la ciudad de origen.

Ejemplo

Imagina que un vendedor necesita visitar 5 ciudades:

- Ciudades: A, B, C, D, E.
- Distancias: Una tabla o un mapa que muestra la distancia entre cada par de ciudades.

El viajante debe encontrar la ruta óptima que minimice la distancia total, visitando todas las ciudades y regresando al punto de partida.

Complejidad del Problema

El problema del Viajante es NP-completo, lo que significa que no existe (hasta ahora) un algoritmo eficiente que resuelva todas las instancias del problema en tiempo polinómico. La cantidad de posibles rutas crece factorialmente ($n!$) con el número de ciudades:

- Para 5 ciudades: $5! = 120$ rutas posibles.
- Para 10 ciudades: $10! = 3,628,800$ rutas posibles.

Por lo tanto, para un gran número de ciudades, no es factible calcular exhaustivamente todas las rutas posibles.

Resolviendo el Problema con Templado Simulado

El Templado Simulado (Simulated Annealing, SA) es una técnica heurística que encuentra soluciones cercanas a la óptima sin necesidad de explorar todas las rutas posibles. Es ideal para problemas como el TSP debido a su capacidad de escapar de mínimos locales durante la optimización.

Cómo funciona el Templado Simulado para el TSP

- **Representación de la Ruta:** Una ruta se representa como una permutación de las ciudades, por ejemplo:
  - Ruta inicial: $A \to B \to C \to D \to E \to A$.
- **Función de Energía:** La energía es el costo total de la ruta, es decir, la suma de las distancias entre las ciudades en el orden actual.
  - Costo de la ruta = Distancia($A \to B$) + Distancia($B \to C$) + ... + Distancia($E \to A$).
- **Generación de Vecinos:** Un vecino se genera modificando ligeramente la ruta actual:
  - Intercambiando dos ciudades.
  - Revirtiendo el orden de una subsección de la ruta.
- **Aceptación de Vecinos:** Si el vecino tiene un costo menor ($\Delta E < 0$), se acepta como la nueva ruta. Si el vecino tiene un costo mayor ($\Delta E > 0$), se acepta con una probabilidad que depende de la diferencia de costo (ΔE) y la temperatura (T):

$$P = e^{-\Delta E/T} \tag{2}$$

  Esto permite al algoritmo explorar soluciones subóptimas inicialmente para evitar quedar atrapado en mínimos locales.

- **Reducción de la Temperatura:** La temperatura (T) se reduce gradualmente después de cada iteración (enfriamiento).
- **Finalización:** El algoritmo termina cuando la temperatura alcanza un valor mínimo (T_{min}), devolviendo la mejor ruta encontrada.