INSTITUTO POLITÉCNICO NACIONAL

Centro de Investigación en Computación

ASIGNATURA:

Metaheurísticas

Actividad #5:

Guía Taller Laboratorio 2

PROFESORA:

Dra. Yenny Villuendas Rey

PRESENTA:

Juan René Hernández Sánchez

Adriana Montserrat García Carrillo

1. Introducción

Los procedimientos metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas complejos de optimización. Las metaheurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos.

Con el método de ascensión de colinas la estrategia es repetidamente expandir un nodo, inspeccionar sus sucesores recién generados, y seleccionar y expandir el mejor entre los sucesores sin mantener referencias a los padres. Cuando se llega a un nodo muerto no hay forma de hacer retroceso (salvo generar otro nodo raíz).

En el siguiente trabajo se profundizará sobre los temas de métodos heurísticos, algoritmo de ascensión de colinas, operadores, ventajas y desventajas y aplicaciones.

2. Desarrollo

Asignatura: Metaheurísticas
Actividad N_o. 5
Guía Taller N_o. 2
Título: Solución de problemas mediante Ascensión de Colinas.
Contenido:

- Métodos heurísticos de solución de problemas.
- Ascensión de Colinas.
- Ascensión de Colinas con mutación aleatoria.

Objetivo: Implementar algoritmos de Ascensión de Colinas, en lenguajes de alto nivel, para la solución de problemas de la profesión.

3. Enuncie las ventajas y desventajas de la Ascensión de Colinas

Ventajas:

- Útil para resolver problemas de optimización y encontrar el mejor estado.
- El algoritmo no mantiene un árbol de búsqueda, por lo que la estructura de datos del nodo sólo necesita registrar el estado y su evaluación.(Norving, 1995)

Desventajas:
Puede presentar los siguientes inconvenientes:

- *Máximos locales*: Una vez en un máximo local, el algoritmo se detendrá (aunque la solución no sea la óptima). (Norving, 1995)
- *Meseta:* una meseta es una zona del espacio de estados donde la función de evaluación es esencialmente plana. La búsqueda realizará un recorrido aleatorio.
- *Crestas:* una cresta puede tener lados muy inclinados, de modo que la búsqueda alcanza la cima de la cresta con facilidad, pero la cima puede tener una pendiente muy suave hacia un pico. A menos que haya operadores que se muevan directamente a lo largo de la cima de la cresta, la búsqueda puede oscilar de un lado a otro, avanzando poco. (Norving, 1995)

4. Detalle el pseudocódigo del algoritmo Random mutation hill-climbing (RMHC)

Algorithm 1 Heurística Random mutation hill-climbing (RMHC)

Input: Un Estado Aleatorio del problema
Output: Un óptimo local

```
1: iterations ← S                ▷ Siendo S ∈ ℕ Un número determinado de Evaluaciones
2: bestEvaluated ← random string
3: bestFitness ← ComputeFitness(bestEvaluated)
4: length ← bestEvaluated.lenght()
5: repeat
6:     locus ← Rand(0, length)
7:     mutatedHilltop ← Mutate(bestEvaluated, locus)
8:     mutatedFitness ← ComputeFitness(mutatedHilltop)
9:     if mutatedFitness ≥ bestFitness then
10:        bestEvaluated := mutatedHilltop
11:        bestFitness := mutatedFitness
12:    end if
13:    iterations := iterations − 1
14: until iterations ≠ 0
15: return bestEvaluated
```

(S., 1993)

5. Realice la modelación matemática necesaria para la solución, mediante RMHC, de los problemas siguientes:

5.1. Problema del viajero vendedor (Travel Salesman Problem)

- Estado inicial: ciclo hamiltoniano.
- Estado final: Un ciclo $x\prime$ **s.a.** $x\prime = \{Min(x), (x\prime)\}$ donde $x_i \neq x_i\prime$, $0 \leq i \leq n$.
- Test objetivo:
  $\sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} x_{ij}$
  s.a.
  $\sum_{j=1}^{n} x_{ij} = 1,\ i = 1, ..., n$
  $\sum_{i=1}^{n} x_{ij} = 1,\ j = 1, ..., n$
  $x_{ij} \in \{\ 0, 1\},\ i = 1, ..., n,\ j = 1, ..., n$
- Acciones posibles (operadores):

  $f : x \mapsto x\prime$ **s.a.** $x\prime$ es una permutación de $i, j = 1, ..., n$ donde $x_i, x_j \neq x_i\prime, x_j\prime$.

6. Dado el Problema del viajero vendedor, proponga las estructuras de datos necesarias para su implementación.

- cadenas
- Listas
- Tuplas

7. Diseñe la interfaz de usuario para la solución del problema planteado

```
from itertools import permutations
import random

def fitness(permutacion,distancias):
    costo=0
    origen=0
    for j in permutacion:
        costo += distancias[origen][j]
        origen=j
    return costo

def permutar(posicion,ciudades):
    res=[]
    #TODO Funcion de permutacion
    return res

N = input('Ingresa el numero de evaluaciones: ')
N = int(N)

#n se refiere al numero de ciudades
n = 4

ciudades=[0,1,2,3]
distancias = [[0,16,85,12],[16,0,10,50],[85,10,0,31],[12,50,31,0]]

fitness_actual = fitness(ciudades,distancias)
eval = 1

while eval < N:
    locus = random.randint(0, n-1)
    nuevo_recorrido = permutar(locus, ciudades)
    nueva_fitness = fitness(nuevo_recorrido, distancias)
    eval += 1

    if nueva_fitness >= fitness_actual:
        ciudades = nuevo_recorrido
        fitness_actual = nueva_fitness


print("Recorrido Final: " + ciudades +
      " con fitness igual a: " + fitness_actual + "\n")
```

8. Conclusiones

En este trabajo se pudo observar que los algoritmos metaheurísticos constituyen ideas generales que permiten un margen de maniobra muy amplio a la hora de ser aplicados. Esta gran versatilidad es la que los hace muy atractivos, ya que es posible adaptarlos a casi cualquier problema de optimización. De igual manera, se logró aprender las ventajas y desventajas de los algoritmos de ascensión de colinas. Descubriendo que son una poderosa herramienta para la resolución de problemas en diversas áreas.

Referencias

Norving, R. . (1995). *Artificial intelligence – a modern approach.* PRENTICE HALL. (Third ed.)

S., M. M. H. J. H. . F. (1993). Relative building-block fitness and the building block hypothesis. *D. Whitley, Foundations of Genetic Algorithms*, *2*(5), 109–126.