

***PROGRAMA 3 – Algoritmo de SALESMAN***

**Asignatura:** INTELIGENCIA ARTIFICIAL

**Profesor:** D. Sc. Gerardo García Gil

2022-B

***López Arellano Ricardo David***

***Universidad de Guadalajara (CUCEI)***

**Presentación**

El clásico problema del viajante de comercio se puede enunciar como: ’si un viajante parte de una ciudad y las distancias a otras ciudades son conocidas, ¿cuál es la ruta ´optima que debe elegir para visitar todas las ciudades y volver a la ciudad de partida?’. El estudio de este problema de programación entera es el objeto de este trabajo. Comenzaremos con una introducción histórica mostrando las inquietudes que hicieron nacer el problema del viajante a mediados del siglo XX. A continuación, veremos la gran dificultad del problema, hallar un circuito Hamiltoniano. En el ´ultimo capítulo estudiaremos diferentes métodos de resolución basados en la ramificación y la acotación. Por ´ultimo, veremos algún ejemplo a modo de aplicación del problema.

**Introducción**

En este documento hablaremos sobre el algoritmo de Salesman el cual es un [algoritmo](https://es.wikipedia.org/wiki/Algoritmo) de la [teoría de grafos](https://es.wikipedia.org/wiki/Teor%C3%ADa_de_grafos) para encontrar un [mínimo](https://es.wikipedia.org/wiki/%C3%81rbol_recubridor_m%C3%ADnimo) en un grafo conexo y ponderado.

En esta investigación se van a diseñar algoritmos heurísticos y metaheurísticos para resolver el Problema del Agente Viajero. Para ello se ha estructurado en quince apartados, en los tres primeros se establece el área de conocimiento y línea de investigación, el tema delimitado y la introducción que describe el cuerpo de esta memoria.

**Antecedentes**

El Problema del Agente Viajero (Traveling Salesman Problem, TSP) es aún un problema abierto en el área de conocimiento de la Programación Matemática. Desde los años cincuenta ha despertado mucho interés dentro de la comunidad científica por su forma sencilla de enunciarse y por su extrema dificultad en resolverse, aún teniendo a mano el desarrollo tecnológico con computadoras cada día más veloces y un sinnúmero de ideas para tratarlo, desde los aportes teóricos, hasta métodos de aproximación como Simulated Annealing, Greedy, Inteligencia Artificial, Simulación heurística, entre otros.

**Desarrollo**

**1.** Ingrese: MatrizDistancia, longitudTabú. MaxIter, CiclosDiver, ListaTabú.

**2.** Generar la solución inicial(permutación natural) y la distancia d(x).

**3**. Mientras Iter ≤ MaxIter.

**3.1.** Actualice MatrizTabu.

**3.2.** Menor distancia ← d(x).

**3.3.** Generar las n − 1 permutaciones.

**3.4.** Determinar la distancia de cada permutación.

**3.5.** Clasifique los movimientos en prohibidos y no prohibidos.

**3.6.** Determinar la menor distancia de los movimientos no prohibidos y seleccione la permutación.

**3.7.** Si todos los movimientos son prohibidos, aplique aspiración. Sino pare.

**3.8.** Actualice d(x).

**3.9.** Si d(x) < Op(x), entonces actualice Op(x). Pase a la instrucción 3.1.

**3.10.** Si en k iteraciones no se actualiza Op(x), entonces diversifique. Pase a la instrucción 3.1.

**4**. El óptimo encontrado es Op(x).

**5.** Fin.

**Implementación**

//RICARDO DAVID LOPEZ ARELLANO

#include<stdio.h>

#include<conio.h>

#include<iostream>

using namespace std;

int c = 0,cost = 999;

int graph[4][4] = { {0, 10, 17, 15},

{20, 0, 19, 18},

{50, 44, 0, 25},

{45, 40, 20, 0}

};

void swap (int \*x, int \*y)

{

int temp;

temp = \*x;

\*x = \*y;

\*y = temp;

}

void copy\_array(int \*a, int n)

{

int i, sum = 0;

for(i = 0; i <= n; i++)

{

sum += graph[a[i % 4]][a[(i + 1) % 4]];

}

if (cost > sum)

{

cost = sum;

}

}

void permute(int \*a, int i, int n)

{

int j, k;

if (i == n)

{

copy\_array(a, n);

}

else

{

for (j = i; j <= n; j++)

{

swap((a + i), (a + j));

permute(a, i + 1, n);

swap((a + i), (a + j));

}

}

}

int main()

{

int i, j;

int a[] = {0, 1, 2, 3};

permute(a, 0, 3);

cout<<"\n\t < ALGORITMO DE SALESMAN >"<<endl<<endl;

cout<<"La matriz a resolver es la siguiente: \n";

cout<<" PINTURAS: B| A| N| R|\n";

cout<<" BLANCO: 0| 10| 17| 15|\n";

cout<<" AMARILLO: 20| 0| 19| 18|\n";

cout<<" NEGRO: 50| 44| 0| 25|\n";

cout<<" ROJO: 45| 40| 20| 0|\n"<<endl;

cout<<"La ruta mas corta es: "<<cost<<endl;

system("pause");

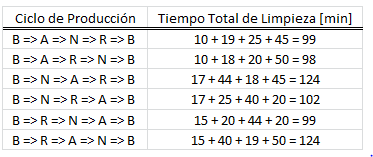
}

**Experimentos**

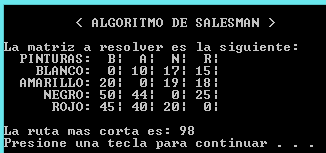
Para lograr programar el algoritmo de Salesman tuve que intentar hacerlo desde en Python, java y termine haciéndolo en c++ ya que siento que ahí soy más fuerte y podía realizarlo de una mejor manera.

**Resultados**

Las corridas posibles son las siguientes:



En mi programa dio el siguiente resultado:



**Conclusión**

El problema del Agente Viajero (TSP) es un problema cuya solución ha sido estudiada desde los inicios de la Inteligencia Artificial considerando que su aplicación puede ser en cualquier área de estudio cuyos problemas reflejen una situación donde se tienen diferentes puntos a visitar con un costo considerado en el enlace entre dichos puntos (costo: recursos empleados como distancia, tiempo, monto económico, etc.). Cada autor ha propuesto soluciones para ciertas instancias de TSP, cada uno con una perspectiva diferente empleando técnicas que no son repetibles pero que, en determinado momento, se pueden emplear para dar lugar a nuevas soluciones; de las técnicas empleadas la más común es el uso de redes neuronales dada su similitud, donde cada neurona es un nodo a visitar y las relaciones entre neuronas es el vector que representa el costo.

**Referencias**

1. Parajón Guevara, A. (2017, agosto). TRAVELING SALESMAN PROBLEM (TSP). FACULTAD REGIONAL MULTIDISCIPLINARIA DE CHONTALES FAREM-CHONTALES. Recuperado 12 de octubre de 2022, de https://repositorio.unan.edu.ni/8779/1/11129.pdf