Viajante del comercio

19 de enero de 2023

Analisis y Diseño de Algoritmos

Herrera Guadarrama Juan Pablo

0.NP Problema del viajante de comercio

Problema del viajante de comercio (TSP): dado un conjunto de ciudades y distancias entre cada par de ciudades, el problema es encontrar la ruta más corta posible que visite cada ciudad exactamente una vez y regrese al punto de partida.

Formulación de la programación lineal en enteros

$$
\begin{aligned}
&\min \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij} x_{ij} \\
&0 \leq x_{ij} \leq 1 && i, j = 0, \ldots n \\
&x_{ij} \text{ integer} && i, j = 0, \ldots n \\
&\sum_{i=0, i \neq j}^{n} x_{ij} = 1 && j = 0, \ldots, n \\
&\sum_{j=0, j \neq i}^{n} x_{ij} = 1 && i = 0, \ldots, n \\
&u_i - u_j + n x_{ij} \leq n - 1 && 1 \leq i \neq j \leq n.
\end{aligned}
$$

1.Análisis y diseño

El tiempo de ejecución es un factor polinómico de orden el Factorial del número de ciudades, esta solución es impracticable para dado solamente 20 ciudades. Una de las mejores aplicaciones de la Programación dinámica es el algoritmo Held–Karp que resuelve el problema en $O(n^2 2^n)$

Ver más explicación en 3.

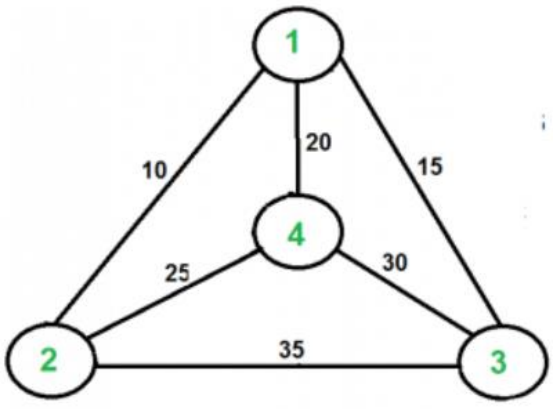
2.Código fuente

```
// C++ HerreraGuadarrama
#include <bits/stdc++.h>
using namespace std;
#define V 4

int travllingSalesmanProblem(int graph[][V], int s);

int main()
{
// representación matricial del gráfico
      int graph[][V] = { { 0, 10, 15, 20 },
                         { 10, 0, 35, 25 },
                         { 15, 35, 0, 30 },
                         { 20, 25, 30, 0 }
                         };
      int s = 0;
      cout << travllingSalesmanProblem(graph, s) << endl;
```

```
	return 0;
}

int travllingSalesmanProblem(int graph[][V], int s)
{

// almacena todos los vértices excepto el vértice de origen
	vector<int> vertex;
	for (int i = 0; i < V; i++)
		if (i != s)
			vertex.push_back(i);

// almacenar peso mínimo Ciclo hamiltoniano.
	int min_path = INT_MAX;
	do {

		// almacenar el peso de la ruta actual (costo)
		int current_pathweight = 0;
		// calcula el peso de la ruta actual
		int k = s;
		for (int i = 0; i < vertex.size(); i++)
		{
			current_pathweight += graph[k][vertex[i]];
			k = vertex[i];
			printf("\nPeso actual %d nodo = %d\n",current_pathweight,k);
		}
		current_pathweight += graph[k][s];

		// actualizar el minimo
		min_path = min(min_path, current_pathweight);

	} while (next_permutation(vertex.begin(), vertex.end()));

	return min_path;
}
```

3. Evidencias

0. Este es problema planteado para comprobar la elaboración del Código

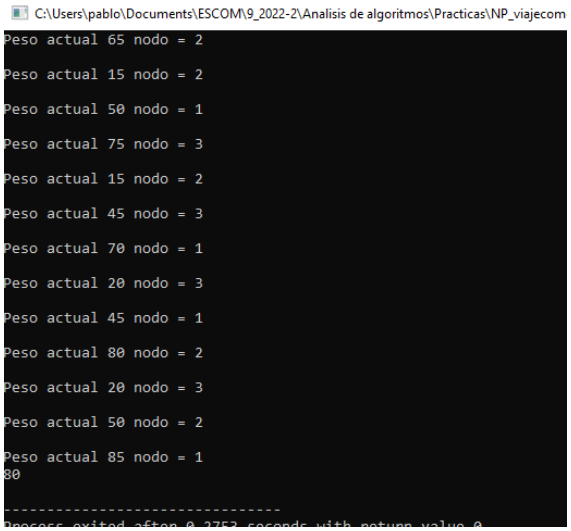


1. Se considera la ciudad 1 como el punto inicial y final. Dado que la ruta es cíclica, podemos considerar cualquier punto como punto de partida.

2. Genera todo (n-1) permutaciones de ciudades.

3. Calcule el costo de cada permutación y realice un seguimiento de la permutación de costo mínimo.

4. Devolver la permutación con coste mínimo.en este caso seria 80.



4. Referencias

El problema del viajero. (2022). sedici. Recuperado 13 de enero de 2023, de http://sedici.unlp.edu.ar/bitstream/handle/10915/4059/1__El_problema_del_viajante_de_comercio.pdf?sequence=4&isAllowed=y

colaboradores de Wikipedia. (2022, 15 octubre). *Problema del viajante*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Problema_del_viajante