Universidad
Carlos III de Madrid

Grado en Ingeniería Informática

Curso 2020/2021

Teoría Avanzada de la Computación

Optimización combinatoria

Autores:

Iván Miguélez García	*100383387*
Alba Reinders Sánchez	*100383444*
Alejandro Valverde Mahou	*100383383*

Índice

1. Introducción

Explicar por encima el TSP
Objetivo y estructura de la práctica
Decir que el código se lleva a cabo en C++ y por qué

2. Algoritmo de búsqueda iterativa

El algoritmo que se plantea es DFS (*Depth First Search*). Este algoritmo consiste en expandir desde el nodo raíz, que representa la posición inicial, todos sus nodos hijos. A continuación, se expanden todos los nodos del primer hijo. Esto se repite por el mismo camino hasta que termina y vuelve al origen. Una vez termina, va iterando hacia atrás y realiza el mismo proceso con todos sus nodos hermanos.

Para la implementación de este algoritmo se ha hecho uso de una pila de nodos expandidos, y cada vez que un nodo generaba nuevos hijos, se añadían a la pila. Cada camino se completa cuando no es capaz de generar más hijos porque ya ha visitado todos los nodos.

Este algoritmo, al aplicarse en el problema del TSP, genera todos los ciclos Hamiltonianos posibles comenzando en un mismo punto y, por tanto, es completo. También se puede asegurar su optimalidad ya que, al generar todos los caminos, y ser todos del mismo tamaño, tan solo hay que almacenar el camino que genera mejores resultados.

Al tener que expandir todos los nodos, el tiempo de computo aumenta considerablemente según el tamaño del problema crece, es decir, cuando se añaden más ciudades el número de nodos a generar crece muy rápido. Esto hace que se busque algún método para reducir el número de nodos a generar. Esta mejora consiste en realizar una poda del árbol de búsqueda cuando el coste acumulado del camino es mayor que el costed del mejor camino ya encontrado.

2.1. Coste Computacional DFS básico

2.1.1. Estudio Analítico

Usando n como el número de ciudades del problema, el número de ciclos Hamiltonianos que se pueden realizar es $(n-1)!$ porque tanto la primera como la última ciudad son la misma y son fijas. En esta implementación no existe un peor caso, ya que es necesario expandir todos los nodos en todos los casos, independientemente de si el mejor camino es el primero o el último.

2.1.2. Estudio Empírico

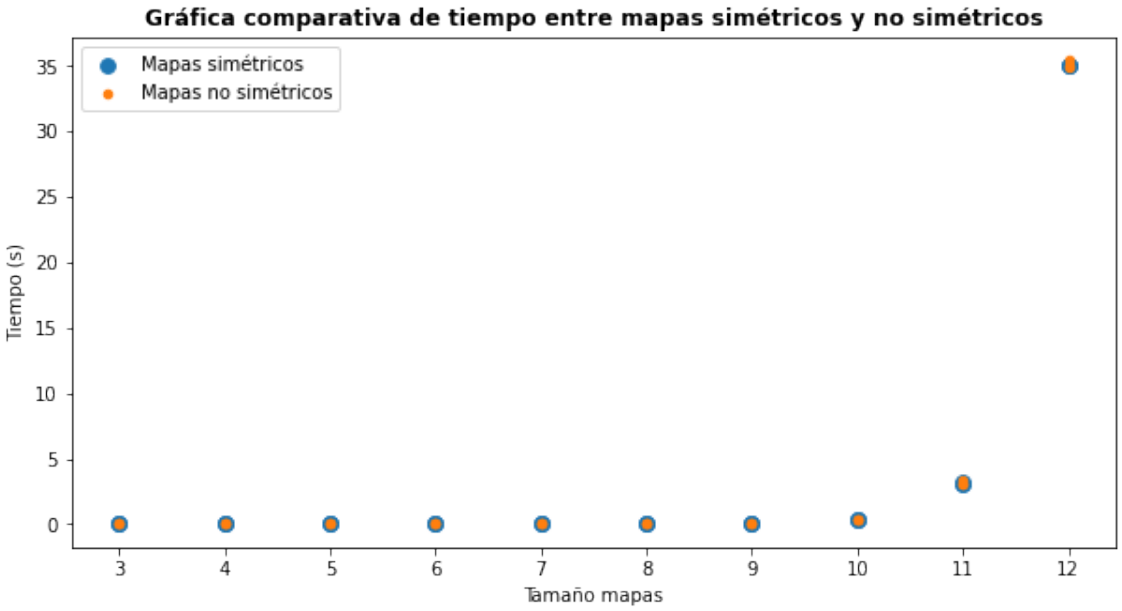

Figura 1: Comparativa de tiempo DFS básico

2.1.3. Estudio Combinado

2.2. Coste Computacional DFS con poda

2.2.1. Estudio Analítico

2.2.2. Estudio Empírico

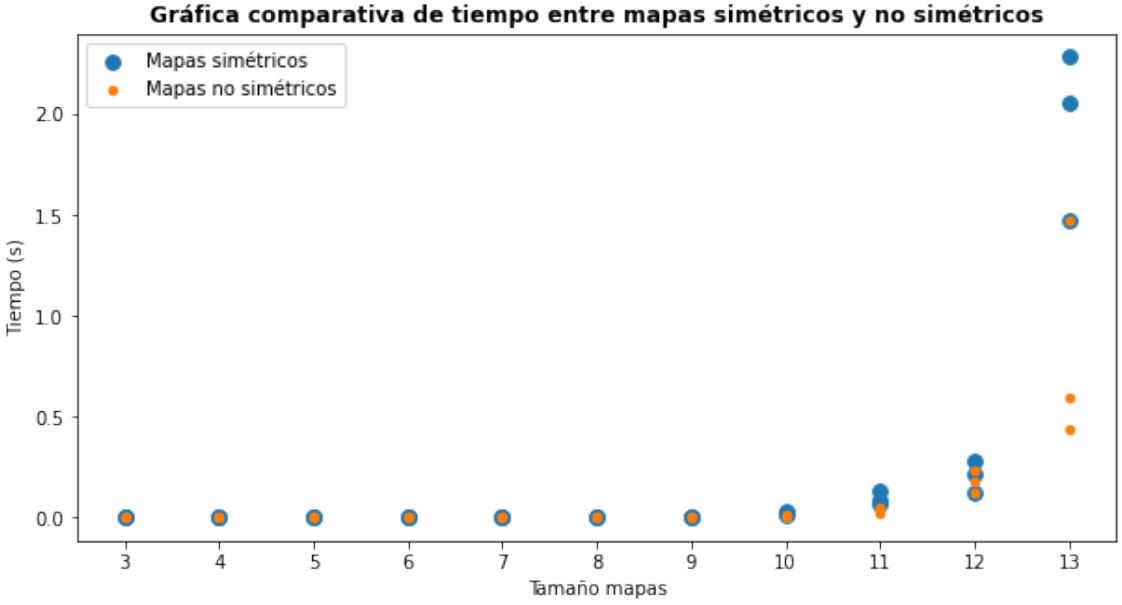

Figura 2: Comparativa de tiempo DFS con poda

2.2.3. Estudio Combinado

3. Algoritmo de búsqueda local

Búsqueda local. Partiendo de la solución de un algoritmo greedy [1], aplicar el operador 2-opt [2]. En este caso, no se podrá garantizar una solución óptima, pero se podrá estudiar qué calidad puede proporcionar el método (en % de desviación respecto al recorrido óptimo) en un tiempo razonable.

3.1. Coste Computacional

3.1.1. Estudio Analítico

3.1.2. Estudio Empírico

3.1.3. Estudio Combinado

4. Conclusión