Tarea 5 - Problemas Conceptuales - Sección 5

Mauricio Urrego (202211641)

1 de diciembre del 2023

1 Definiciones

1.1 ¿Qué es un problema de optimización?

Un problema de optimización es aquel en el que se busca encontrar la mejor solución posible entre un conjunto de soluciones factibles. La mejor solución se define mediante una función objetivo, que asigna un valor a cada solución factible. El objetivo es encontrar la solución que maximice o minimice el valor de la función objetivo, dependiendo del problema específico [3].

1.2 Concepto de algoritmo de aproximación

Un algoritmo de aproximación es un tipo específico de algoritmo diseñado para resolver problemas de optimización. Se diferencia de un algoritmo de optimización exacto en que no garantiza encontrar la solución óptima. En cambio, proporciona una solución que se garantiza que está dentro de un factor determinado de la solución óptima [1].

1.3 ¿Cómo se define el factor de aproximación de un algoritmo?

El factor de aproximación de un algoritmo de aproximación es una medida de cuán cerca está la solución producida por el algoritmo de la solución óptima. Normalmente se define como la relación entre el valor de la función objetivo de la solución producida por el algoritmo y el valor de la función objetivo de la solución óptima [2].

2 Tipos de Aproximación

2.1 Diferencia entre algoritmos de aproximación y algoritmos de optimización exacta

1. **Algoritmos de aproximación:** No garantizan encontrar la solución óptima, pero proporcionan soluciones con la garantía de estar dentro de un factor determinado de la solución óptima. Suelen ejecutarse en tiempo polinómico, lo que los hace adecuados para problemas a gran escala.

2. **Algoritmos de optimización exactos:** Buscan encontrar la solución óptima. Aunque garantizan la búsqueda de la mejor solución, suelen tener una complejidad temporal exponencial, lo que los hace imprácticos para problemas a gran escala [1].

2.2 ¿En qué se diferencian los problemas de aproximación polinomial de los problemas NP-completos?

1. **Problemas de aproximación polinomial:** Se pueden resolver mediante algoritmos de aproximación que se ejecutan en tiempo polinómico respecto al tamaño de la entrada. Además, estos problemas suelen tener un factor de aproximación independiente del tamaño de la entrada.

2. **Problemas NP-completos:** Son una clase de problemas conocidos por ser difíciles de resolver con exactitud. Aunque se pueden resolver mediante búsqueda exhaustiva en tiempo exponencial, no existe ningún algoritmo eficiente conocido para encontrar la solución óptima. Sin embargo, muchos problemas NP-completos admiten algoritmos de aproximación eficientes [2].

3 Factores de Aproximación

3.1 ¿Cómo se calcula el factor de aproximación de un algoritmo? [2]

El factor de aproximación se calcula dividiendo el valor de la función objetivo de la solución producida por el algoritmo de aproximación por el valor de la función objetivo de la solución óptima.

1. **Para un problema de minimización:** Factor de aproximación = Valor de la función objetivo de la solución de aproximación / Valor de la función objetivo de la solución óptima

2. **Para un problema de maximización:** Factor de aproximación = Valor de la función objetivo de la solución óptima / Valor de la función objetivo de la solución de aproximación

3.2 Términos "factor de aproximación garantizado" y "factor de aproximación esperado

1. **Factor de aproximación garantizado:** proporciona un límite en el peor de los casos para el rendimiento del algoritmo. Esto significa que el algoritmo siempre encontrará una solución que esté dentro de un cierto factor de la solución óptima, independientemente del problema de entrada.

2. **Factor de aproximación esperado:** proporciona un límite de caso promedio sobre el rendimiento del algoritmo. Esto significa que el algoritmo encontrará una solución que esté dentro de un cierto factor de la solución óptima en promedio, pero puede funcionar peor en algunas entradas específicas.

4 Problemas prácticos

4.1 Clique

La solución de este problema se compone de dos partes: hallar el grafo $T(G)$ de clausura transitiva y posteriormente calcular el tamaño de la clique más grande en $T(G)$. Para la primera parte se hizo uso del algoritmo DFS, consiguiendo una complejidad de $O(V^3)$ en el peor de los casos. Para esta parte llamamos recusivamente a la función DFS para cada nodo del gráfico para marcar sus vértices alcanzables. Finalmente, para la segunda parte del algoritmo, haciendo uso de recursión, calculamos todos los subconjuntos posibles de los vértices y llamamos a la función *is_clique* para verificar si el conjunto de vértices dado forma una clique o no. La complejidad temporal de este algoritmo es de $O(2^V \cdot V^2)$. Siendo esta también la complejidad total del algoritmo.

4.2 *Unidirectional TSP*

El Problema del Viajante (TSP) es un problema clásico de optimización en el que el objetivo es encontrar la gira más corta posible que visite un conjunto de ciudades y regrese a la ciudad de origen. El enfoque de programación dinámica (el cual fue utilizado), para resolver el TSP implica descomponer el problema en subproblemas más pequeños y resolverlos de manera recursiva, almacenando las soluciones en una tabla para evitar cálculos redundantes. La complejidad de esta solución de programación dinámica es $O(2^V \cdot V^2)$, donde n es el número de ciudades. Esta complejidad surge de la necesidad de considerar todos los subconjuntos posibles de ciudades en cada paso del algoritmo de programación dinámica, lo que conduce a un crecimiento exponencial en el número de subproblemas. A pesar de su eficacia para instancias pequeñas o moderadamente grandes, claramente el enfoque de programación dinámica se vuelve computacionalmente inviable para instancias grandes de TSP debido a su complejidad temporal exponencial.

References

[1] URL: https://www.geeksforgeeks.org/approximation-algorithms/.

[2] URL: https://cs229.stanford.edu/notes-spring2019/cs229-notes2.pdf.

[3] Thomas H. Cormen et al. *Introduction to algorithms*. Second edition. Cambridge, MA, U.S.A.: The MIT Press, 2009.