TALLER N°2

El problema del coloreo de vértices

usach

Taller de programación 2-2024

Fecha: 11/12/2024

Autor: Vicente Aninat

TALLER N°2

El problema del coloreo de vértices

Explicación breve del algoritmo

En base a los algoritmos DSatur y Branch and Bound. Para resolver el problema del coloreo de vértices primero se define el grafo que se planea resolver, una vez teniéndolo se toma el vértice no coloreado con mayor grado de saturación para pintarlo y el grafo resultante se vuelve a calcular de manera recursiva. De esta forma se calculan todos los posibles caminos de solución. Caminos que no siempre son terminados, ya que existen criterios de cuándo vale la pena calcular un estado.

Al momento de que se detecte que todos los vértices han sido coloreados, se actualizan los criterios de cálculo y se conserva aquel estado final con menor cantidad de colores diferentes.

Heurísticas o técnicas utilizadas

Para la resolución del problema se utilizan principalmente los algoritmos Greedy, DSatur, Backtracking y Branch and Bound. Los primeros tres sirven para apoyar a la implementación del algoritmo Branch and Bound, ya que Greedy y DSatur permiten tener un acercamiento inicial para conocer el menor número de colores del problema.

Cuando se llama al Branch and Bound, el principal algoritmo de resolución del programa, este hace uso de cálculos preliminares hechos por DSatur para conocer la cota superior de colores del problema, también se establece la heurística de que si el número actual de colores supera a la cota superior, este estado no se calcula y tampoco todos sus derivados.

La cota superior pasa por un proceso de actualización en el que si durante la ejecución se encuentra un estado final con un número de colores menor a la cota superior, la cota superior adopta el número de colores del estado final, modificando la cota en tiempo de ejecución, mejorando su exactitud y acotando los estados a calcular en favor del tiempo de ejecución.

La cota inferior se actualiza con un llamado DSatur sobre cada estado nuevo, de esta forma se actualiza esta cota inferior y se reduce la cantidad de estados a operar junto con el tiempo de ejecución.

Funcionamiento del programa

A continuación se presenta una muestra del funcionamiento del código

- Si el grafo del estado está completamente coloreado

        - Si no hay mejor estado o si el número de colores del estado es menor a la cota superior
            - El estado actual se vuelve el mejor
            - La cota superior se vuelve el número de colores del estado actual
        - Se retorna el número de colores del mejor estado
- Si no
    - Si el mayor entre la cota inferior y el número de colores del estado actual es menor a la cota superior
        - Se selecciona el vértice con mayor grado de saturación
        - Se colorea con todos los vértices posibles
        - Se calcula la cota inferior para el nuevo estado con el algoritmo DSatur
        - Para cada vértice coloreado en el estado derivado, se llama al algoritmo para el nuevo estado creado
        - Si el vértice no pudo colorearse
            - Se añade un nuevo color con el que pintar
            - Se llama al algoritmo para el nuevo estado
            - Se calcula la cota inferior para el nuevo estado con el algoritmo DSatur
            - Se llama nuevamente al algoritmo para el huevo estado.
    - Se retorna el número de colores del mejor estado

El algoritmo calcula todos los caminos posibles a excepción de aquellos que no se consideran confiables para la determinación del mejor resultado.

Aspectos de implementación y eficiencia

Los principales elementos de la implementación del algoritmo Branch and Bound con DSatur utilizado se detallan a continuación:

- La utilización preferencial de los vertices con mayor grado de saturación, permite encontrar cotas inferiores más temprano en la ejecución, lo que permite reducir el tiempo de espera antes de empezar a acortar el conjunto de estados que deben calcularse.
- La condición de poda implementada en relación al número de colores del estado, la cota inferior y la cota superior permite una poda rigurosa de los estados, esto significa un recorte importante en la utilización de memoria y el tiempo de ejecución.
- Para encontrar los nuevos valores de la cota inferior se utiliza un algoritmo DSatur en vez del Branch and Bound en el sub grafo colorado del nuevo estado, esto con el fin de acortar tiempos de ejecución.

Todos los aspectos anteriormente mencionados contribuyen en un tiempo de ejecución total que va desde los microsegundos hasta 20 segundos en los grafos más grandes probados.

Ejecución del código

Para compilar, ejecutar ‘make’ por línea de comandos sobre la carpeta de código fuente del programa (también es posible escribir ‘make clean’ para deshacer los archivos de objeto creados en la ejecución). Luego, escribir ‘./main’ por línea de comandos.

Al momento de ejecutar, se solicitará que escriba el nombre del archivo de grafos a calcular, para esto se debe colocar en línea de comandos el nombre de cualquiera de los archivos sin la extensión .txt. Al hacer esto el programa calculará la solución por medio de los algoritmos Greedy, DSatur y Branch and Bound, calculando los tiempos de cada uno y se detendrá luego de hacerlo.

El grafo de ejemplo proporcionado para la evaluación, se encuentra cargado como “ejemploTDP”.

Bibliografía

Furini, F., Gabrel, V., & Ternier, I.-C. (s. f.). *An improved DSATUR-based branch and bound for the vertex coloring problem*. Université Paris Dauphine.

Lewis, R. (2023, 4 abril). *DSatur Algorithm for Graph Coloring*. GeeksForGeeks. Recuperado 12 de diciembre de 2024, de https://www.geeksforgeeks.org/dsatur-algorithm-for-graph-coloring/