**Data Structure**

Catedrático: Fernandojosé Boiton Tello

Catedrático auxiliar: Luis Angel Tórtola Tejeda

Sharon Anesveth A. Maatens (20190339)

**Longest Path**

En teoría de grafos, el problema del camino más largo es, dado un grafo, encontrar un camino simple de longitud máxima. Una ruta es simple si no contiene ningún vértice más de una vez. La longitud de una ruta se puede definir como el número o el acumulado peso de sus bordes. LP es NP-completo

1. **A Streaming Algorithm for the Undirected Longest Path Problem**

Algoritmo de transmisión para el problema de ruta más largo en grafos no dirigidos. el grafo de entrada se proporciona como una secuencia de bordes y la RAM se limita a solo un número lineal de bordes en un tiempo (lineal en el número de vértices n).

Funciona en dos fases. En la primera fase, información global en el grafo se recopila en forma de un número constante de árboles de expansión (arboles conectados) T1,. . . , Tτ. Esto es posible en el modelo de transmisión ya que, en términos generales, para un árbol de expansión podemos ‘tomar bordes como vienen ". Se puede construir un árbol de expansión en una sola pasada; sin embargo, utiliza múltiples pases y se limita el grado máximo durante los primeros pases para favorecer las estructuras de camino y evitar grupos de bordes. Los experimentos indican claramente que esta limitación de grado es esencial para la calidad de la solución. Los árboles de expansión se ajustan a la RAM, ya que consideramos τ como constante (de hecho tendremos τ = 1 o τ = 2 en los experimentos). Después de la construcción de τ árboles, se fusionan en un grafo U tomando la unión de sus bordes. Entonces usamos algoritmos estándar para determinar un long path P en U, aislar P y finalmente agregar suficiente bordes alrededor de P para obtener un árbol T.

Luego, en la segunda fase, realizamos más pases durante los cuales probamos si el intercambio de bordes individuales de T puede mejorar el longest path en él. (Se puede encontrar un longest path en un árbol realizando DFS dos veces [10]; la longitud de un longest path en un árbol es su diámetro).

El principal desafío en la segunda fase es determinar rápidamente qué bordes deben intercambiarse. Este algortimo muestra que esta decisión se puede tomar en tiempo lineal, por lo tanto, se obtiene un procesamiento por borde tiempo de O (n).

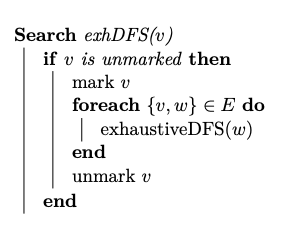
1. **Parallel Optimal Longest Path Search**

Esta sección presenta un algoritmo llamado "Ruta más larga por programación dinámica" (LPDP). El algoritmo resuelve el problema de ruta más larga (LP) para grafos ponderados no dirigidos. LPDP se basa en los principios de la programación dinámica.

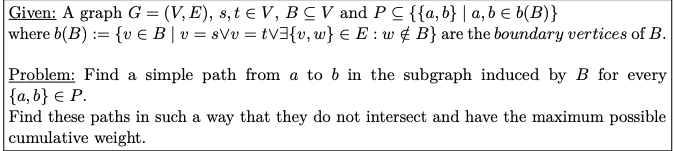
Una forma sencilla de resolver el problema del camino más largo es la búsqueda exhaustiva de profundidad primero. En la búsqueda regular de profundidad primero (DFS), un vértice tiene dos estados: marcado y sin marcar. Inicialmente, todos los vértices no están marcados. La búsqueda comienza llamando al procedimiento DFS con un determinado vértice como parámetro. Este vértice se llama raíz. El vértice actual (el parámetro de la llamada DFS actual) se marca y luego el procedimiento DFS se ejecuta recursivamente en cada vértice no marcado alcanzable por un borde desde el vértice actual. El vértice actual es llamado el padre de estos vértices. Una vez que finalizan las llamadas recursivas DFS, retrocedemos al vértice padre.

La búsqueda finaliza una vez que DFS retrocede desde el vértice raíz. De esa manera, cada ruta simple en el grafo a partir de la raíz se explora el vértice.

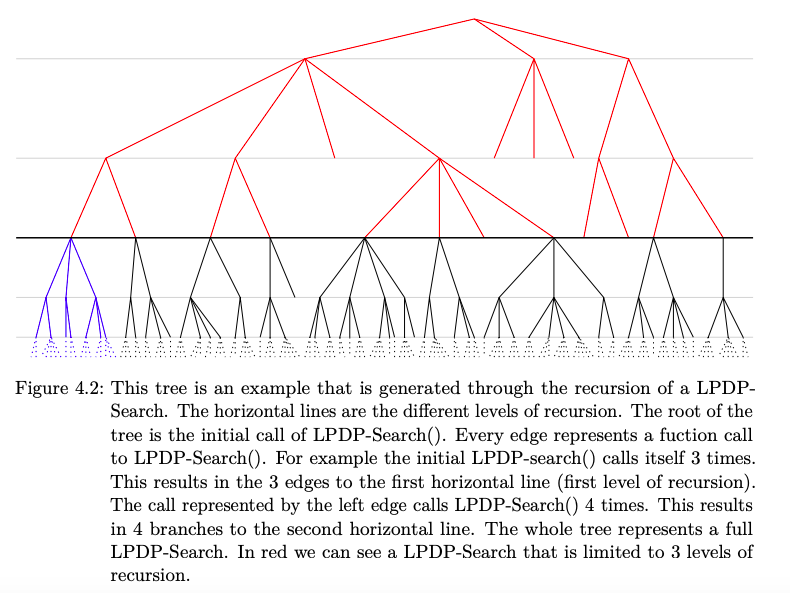
El problema de LP se puede resolver con DFS exhaustivo utilizando el inicio vértice como la raíz. Durante la búsqueda, la longitud de la ruta actual se almacena y compara a la mejor solución anterior cada vez que se alcanza el vértice objetivo. Si la longitud actual es mayor que la de la mejor solución, se actualiza en consecuencia. Cuando la búsqueda se ha encontrado una ruta con una longitud máxima de s a t. Si almacenamos la longitud de la ruta más larga para cada vértice (no solo el vértice objetivo), entonces las rutas largas simples desde s hasta cualquier otro vértice se pueden calcular simultáneamente.



La programación dinámica requiere que podamos dividir LP en subproblemas. A fin de que hacer esto primero generalizamos el problema:



Para hacer esto simultaneamente a traves de la “paralelizacion” (inspirado en el "Cubo y conquista" enfoque para la solución SAT presentado por Heule et al.) la fórmula SAT se divide en muchas subformulas. Estas subformulas pueden resolverse en paralelo. Podemos hacer lo mismo para LPDP dividiendo el espacio de búsqueda de LPDP-Search en muchas ramas disjuntas. Hacemos esto ejecutando LPDP-Search desde cada límite vértice con una profundidad de recursión limitada. Cada vez que la búsqueda alcanza un cierto nivel de recursividad la búsqueda almacena su contexto actual en una lista y vuelve a la recursividad anterior nivel.



**Shortest Path**

En la teoría de grafos, el problema del camino más corto es el problema que consiste en encontrar un camino entre dos vértices (o nodos) de tal manera que la suma de los pesos de las aristas que lo constituyen es mínima.

1. **SLF algorithm: The Small Label First algorithm (SLF)**
2. **A Generalized Threshold Algorithm for the Shortest Path Problem with Time Windows**
3. **Dijkstra’s Algorithm with Arc-Flags (Fast Point-to-Point Shortest Path Computations with Arc-Flags)**

En el problema de la ruta más corta punto a punto (P2P) se tiene que encontrar una ruta más corta entre dos nodos especificados en un grafo de entrada. Un algoritmo estándar para este problema es el desarrollado por Dijkstra (1959) que se ejecuta en tiempo O (m + n log n) (Fredman y Tarjan, 1987). Sin embargo,en la práctica, las técnicas de aceleración pueden reducir el tiempo de ejecución y a menudo resultan en una ejecución sublineal respecto al tiempo. Dependen crucialmente del hecho de que el algoritmo de Dijkstra establece etiquetas y que puede ser terminado cuando se liquida el nodo de destino. Por lo tanto, el algoritmo no necesariamente busca todo el grafo.

Si permitimos un paso de preprocesamiento, el tiempo de ejecución se puede reducir aún más con el siguiente insight: considere, para cada arco a, el conjunto Sa de nodos a los que se puede llegar por una ruta más corta que comienza con a. Es fácil verificar que el algoritmo de Dijkstra se puede restringir al subgrafo con esos arcos para cuál es el nodo de cola t en Sa. Sin embargo, almacenar todos los conjuntos Sa requiere O (n^2) espacio, lo cual es prohibitivo para grafos grandes (n >> 1M). Por lo tanto, se utiliza una partición del conjunto de nodos V en p (: = | R |) regiones para una aproximación del conjunto Sa. Formalmente, usaremos una función

r : V → {1,. . . , p}

que asigna a cada nodo el número de su región. Ahora usaremos un vector de bandera fa: {1,. . . , p} →{true, false} con p entradas, cada una de las cuales corresponde a una región. Para cada arco a, establecemos la entrada

fa (i) a true, si a es el comienzo de cualquier ruta más corta a al menos un nodo en la región i ∈ {1,. . . , pag}. Además, para cada arco (v, w) con v, w ∈ V configuramos la entrada de bandera f (v, w) (rw) a true.

Para una consulta de ruta más corta específica de sa t, el algoritmo de Dijkstra puede restringirse al subgrafo Gt inducido por aquellos arcos donde la entrada de la bandera corresponde a la región objetivo (la región donde t pertenece a) es true.

**Bibliografía**

https://drops.dagstuhl.de/opus/volltexte/2016/6398/pdf/LIPIcs-ESA-2016-56.pdf

https://publikationen.bibliothek.kit.edu/1000104171/51132444