Shortest and longest path algorithms

David Gabriel Corzo Mcmath

2020 April 03, 02:22PM

1. Dial's algorithm

Dial's algorithm is an optimized version of Dijkstra's shortest path algorithm, as we already know, Dijkstra's Algorithm sorts shrotests paths according to the weight that exists in each edge, according to these prexisting weight distribution along the graph the algorithm starts to determine the shortest paths using two lists, a visited list and an unvisited list, the visited list holds all the vertices that have been visited and calculated shortest paths with their neighbors. The Dial's Algorithm takes the concept of the Dijkstra's Algorithm solution and ultimately makes it better for graphs that hold edges which weights are defined in a close range, let's say that the standard deviation of all edges is quite small and therefore there is more complexity at the time in which one has to calculate the shortest paths from one vertex to another in the graph. This algorithm makes use of tha bucket data structure, and the time complexity of this algorithm thus becomes $O(E+WV)$ where W is the weight, E is the number of edges and V is the number of vertices.

1.1. Pseudocode

1. Maintains some buckets, numbered 0, 1, 2,. . . ,wV.
2. Bucket k contains all temporarily labeled nodes with distance equal to k.
3. Nodes in each bucket are represented by list of vertices.
4. Buckets 0, 1, 2,..wV are checked sequentially until the first non-empty bucket is found. Each node contained in the first non-empty bucket has the minimum distance label by definition.
5. One by one, these nodes with minimum distance label are permanently labeled and deleted from the bucket during the scanning process.
6. Thus operations involving vertex include:
7. Checking if a bucket is empty
8. Adding a vertex to a bucket
9. Deleting a vertex from a bucket.
10. The position of a temporarily labeled vertex in the buckets is updated accordingly when the distance label of a vertex changes.
11. Process repeated until all vertices are permanently labeled (or distances of all vertices are finalized).

From: `https://www.geeksforgeeks.org/dials-algorithm-optimized-dijkstra-for-small-range-weights/`

1.2. Applications

This algorithm is sort of a subset of Dijkstra's shortest path algorithm, thus it can have practically the same real world applications, below are some examples.

- Telephone network
- Flight agenda
- Designate file server

1.3. Advantages

Provides an optimized way of finding the shortest path in a graph, more optimized than the Dijkstra's shortest path algorithm.

2. Large Label Last Algorithm / LLL Algorithm

The Large Label Last algorithm is an optimization over the Bellman Ford Algorithm, this algorithm has a time complexity of $O(|V| \cdot |E|)$ with an average running time of $O(|E|)$. It is also based on the idea of the priority queue, there are two ways of optimizing the Bellman Ford Algorithm, either by Small Label First (SFL) or with Large Label First, which is the algorithm this section is focusing on. Using iterations and vectices the algorithm determines which path is optimal during which a queue is saved with the paths examined thus far.

2.1. Pseudocode

```
procedure Shortest-Path-Faster-Algorithm(G, s)
 1    for each vertex v ≠ s in V(G)
 2        d(v) := ∞

 3    d(s) := 0
 4    offer s into Q
 5    while Q is not empty do
 6        u := poll Q
 7        for each edge (u, v) in E(G) do
 8            if d(u) + w(u, v) < d(v) then
 9                d(v) := d(u) + w(u, v)
10                if v is not in Q then
11                    offer v into Q
procedure Large-Label-Last(G, Q)
 x := average of d(v) for all v in Q
 while d(front(Q)) > x
     u := pop front of Q
     push u to back of Q
```

from: `https://en.wikipedia.org/wiki/Shortest_Path_Faster_Algorithm`

2.2. Applications

1. Distance-vector routing protocol
2. Internet protocol applications

2.3. Advantages

It provides a better and more optimized way of checking which path provides the shortest path.

3. Doubling algorithm

Again the Doubling algorithm is based on iterations whose primary consern is to calculate the shortest paths for all vertices composed of an increasing number of edges. Let's say, the graph starts out with one edge, this algorithm would calculate if one edge connecting the initial vertices would be better than two edges of paths connecting the target vertices, thus it finds the shortest paths doubling the edges and checking if such operation increased the eficiency of the path and if it shortened it. All of this happens in computational complexity of $\Theta(n^3)\log_2(N)$.

4. SLF/LLL Algorithm

This algorithm is a "best of both worlds" type of algorithm that combines the SFL and the LLL algorithms. Thus we get efficiency is increased and time complexity decreased. First it combines the SLF algorithm for the addition of vertices to the candidate list and the LL algorithm method for theit retrieval from the list. This algorithm requires thus less iterations, but it does have a down side, it needs to perform more calculations.

5. Generic Algorithm

It sets a First In First Out queue for the candidate list V to store all the vertices that are to be checked. Next in the queue at which operations of additions and retrieval of a vertex tothe end of it or from its head, respectively, are performed. [researchgate.net]

6. Sources

1. https://www.youtube.com/watch?v=pVfj6mxhdMw
2. http://www.csl.mtu.edu/cs2321/www/newLectures/30_More_Dijkstra.htm
3. https://www.youtube.com/watch?v=PwJwc5oj8cw
4. https://brilliant.org/wiki/bellman-ford-algorithm/#applications
5. https://en.wikipedia.org/wiki/Shortest_Path_Faster_Algorithm
6. https://www.researchgate.net/publication/242013156_Efficiency_Evaluation_of_Shortest_Path_Algorithms