INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY, HYDERABAD

APS Project Report

Planar Graph All Pair Shortest Path and Its Variations

Ankit Mishra (2018201079)

Pranjal Patidar (2018201094)

Introduction :-

This is an important problem in graph theory and has applications for real life problems such as road networks (navigation) etc.

In this project, we have to optimize the all pair shortest path (APSP) algorithm for planar graphs. As, the standard algorithms find APSP for generic graphs, without categorizing in planar and non -planar graphs, but planar graphs are sparse or in other words they have less number of edges, which leads to unnecessary computation by considering every possible edge (as in Floyd Warshall algorithm) in standard APSP algorithms.

The Average Degree of a planar graph is strictly less than 6 (http://www-math.mit.edu/~djk/18.310/18.310F04/planarity_coloring.html). So using this heuristics we implemented an algorithm to optimize all Pair Shortest Path Algorithm for planar graph. We used Directional Path Consistency for detection of negative cycle in graph and triangulation.

Brief Background :-

The standard algorithms considers all possible pairs of edges, either they are present or not, which is seen as drawback for planar graphs, because it has very less edges as compared to the graph which has all possible edges(for whom the standard algorithm runs), hence most of the time on standard algorithm the planar graphs, execute loops without relaxing any edges, as edges are relaxed only by the edges connecting them to their immediate neighbours, which will be less than the 'n'(all possible) edges.

Let us consider vertex 'A' has 'X' immediate neighbours, graph has 'N' vertices and N > X, then on standard algorithms 'N – X' iterations will be wasted, as they are not connected with the A vertex, but the loop will be executed for them, without relaxing any edge or decreasing any pair's shortest path. Hence, our prime attention will be toward reducing those waste iterations, for planar graphs.

The property of graphs, which tell us about the immediate neighbours or the number of edges has it as end-point, is degree of vertex, and we have to explore this property to get the optimization, as we have to consider only edges which are present and connected to that vertex. Hence we will use degree as heuristic to improve the APSP algorithms, for planar graphs.

Approach :-

We are taking input in adjacency matrix.

This algorithm is divided into four phases,

- Finding vertex ordering
- Finding directed path consistency
- Find left and right neighbours
- Finding all pair shortest paths

◆ **Finding vertex ordering:-**

In this phase we order the vertices according to their degree, in increasing order, which will be used further in our implementation. We maintain the degree of vertices while taking input as, we will have vector of size equal to number of vertices, and increment the value of index equal to vertex number.

◆ **Finding directed path consistency(DPC) :-**

In this phase we go through each node, according to the order (in decreasing order), which has been generated in previous phase. And for every vertex we will check for every pair of vertices, which are ahead of it in order(in decreasing order), if there is any path through the selected vertex between the two vertices of pair, with less weight than they currently have. If there is any such path, then add that edge upon the previous edge, with lowering the path weight. And if any negative cycle is present or formed after adding the edges, in the graph then, terminate program by showing the message that graph is inconsistent.

◆ **Finding left and right neighbours:-**

In this phase, we check and store immediate neighbours after the DPC phase, in left and right side of that vertex according to the order (made in first phase), it will be used in reducing the loop iterations during calculating the APSP. For this we are using to vector-of-vectors one for left neighbours and one for right neighbours.

◆ **Finding all pairs shortest paths :-**

To find APSP, we implement Chleq's point-to-point algorithm on every vertex according to the order (created in first phase). The first loop of Chleq's point-to-point algorithm runs for left neighbours of all vertices which are left of the source (for which

call is done) including source, and then second one runs for right neighbours of all the vertices according to the order.
After implementing all the phases, finally we will have all pair shortest paths.

How it is optimized :-

In standard Floyd Warshall's algorithm, there are three nested loops which leads to $O(n^3)$ Time complexity, but in our implementation, while computing shortest path, there will be three nested loops, outer-most will run for 'n'(number of vertices) time, middle one will run at max 'n' times, but the last loop will run only maximum number of neighbours any vertex has(either from left or right side of the order), let the maximum number of neighbours of any vertices be w_d so, the asymptotic time complexity of our implementation will be $O(n^2 w_d)$.

Algorithms Used :-

- ***DPC Algorithm***

```
for k ← n to 1 do
    forall i < j < k such that {i, k}, {j, k} ∈ E do
        w i→j ← min{w i→j , w i→k + w k→j }
        w j→i ← min{w j→i , w j→k + w k→i }
        E ← E ∪ {{i, j}}
        if w i→j + w j→i < 0 then
            return INCONSISTENT
return CONSISTENT
```

- ***Min-Path Algo***

```
∀i ∈ V : D[i] ← ∞
D[s] ← 0
for k ← s to 1 do
    forall j < k such that {j, k} ∈ E do
        D[j] ← min{D[j], D[k] + w k→j }
for k ← 1 to t do
    forall j > k such that {j, k} ∈ E do
        D[j] ← min{D[j], D[k] + w k→j }
return D[t]
```

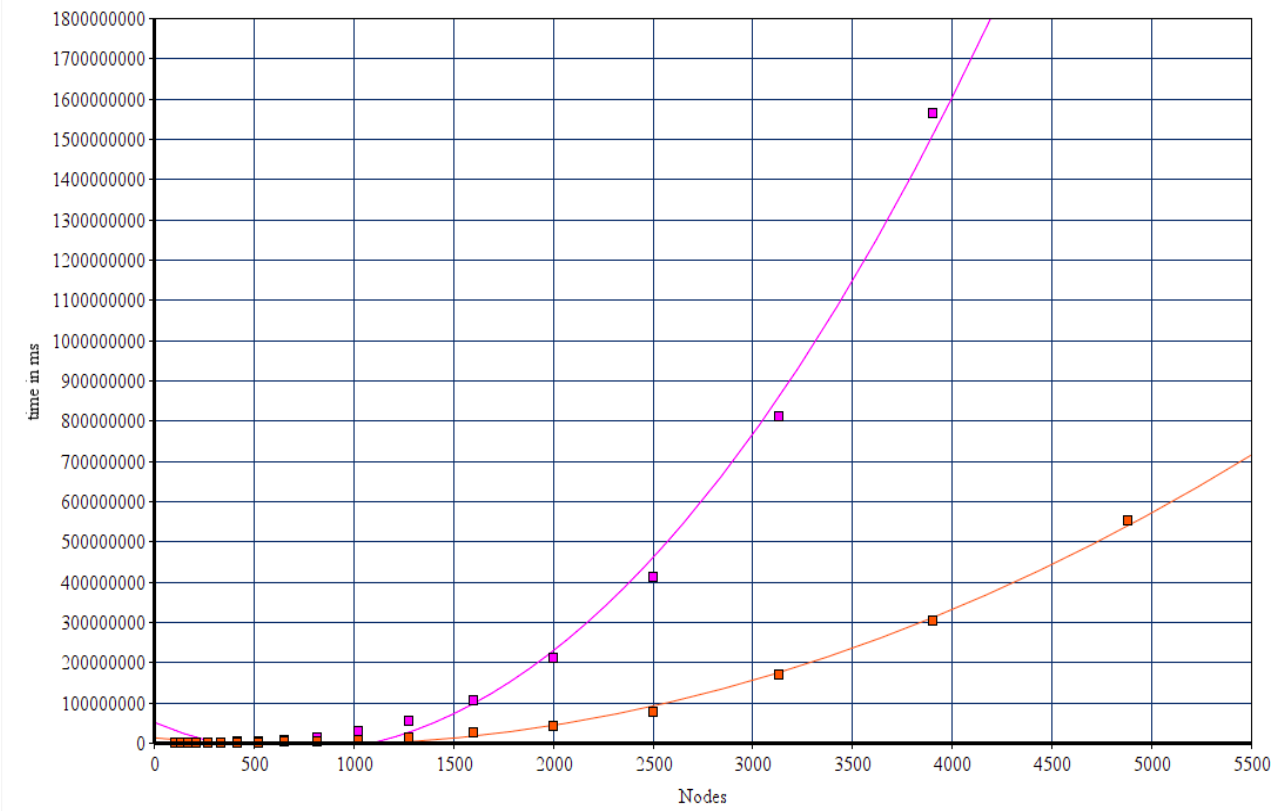
- ***Chleq-APSP***

```
for i ← 1 to n do
    D[i][∗] ← Min-paths (G, i)
return D
```

Results and Analysis :-

We tested both the algorithms Floyd-Warshall and algorithm by Chleq's algorithm on New York City road networks test cases and observed that for larger test cases Chleq's algorithm with min-degree heuristics works much efficiently for Planar Graphs. The test cases results are as follows –

nodes	Heuristic Planar	Floyd-Warshall's
108	27630	42411
136	40481	78840
171	71632	136713
214	119888	257293
268	257496	509384
335	487147	964233
419	849080	1896055
524	1452588	3680303
655	2553968	7209174
819	4208183	14139377
1024	7029084	27611309
1280	12708105	53885077
1600	25796686	105001426
2000	42051825	211996786
2500	78103020	412596394
3125	170179267	811822862
3906	303649131	1562460087
4882	550856994	--



Comparison graph

Git Repository :-

https://github.com/pranjal2018201094/APS_PROJECT

References :-

https://arxiv.org/abs/1401.4609

https://nptel.ac.in/courses/106106158/18

https://data.4tu.nl/repository/uuid:698db457-499f-48a1-bb26-5a54070b4dbe

http://www-math.mit.edu/~djk/18.310/18.310F04/planarity_coloring.html

https://en.wikipedia.org/wiki/Planar_graph