All Pairs Shotest Path Algorithm for Planar Graphs

1. Introduction

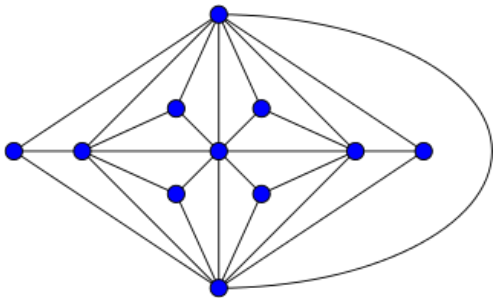
The all-pairs shortest path problem is the determination of the shortest graph distances between every pair of vertices in a given graph. The canonical way of solving this problem is by computing all pairs shortest paths (APSP) using the Floyd–Warshall algorithm, which runs in $O(n^3)$ where n is the number of vertices in the given graph. But in case of planar graph, we can exploit the fact that the graph is not dense and present ***Snowball*** algorithm which gives better running time complexity for the planar graphs.

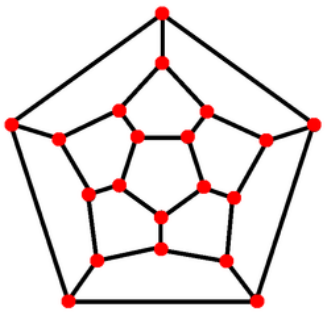
2. Formulation of the problem

Definition:

Planar Graph:

In graph theory, a planar graph is a graph that can be embedded in the plane, i.e., it can be drawn on the plane in such a way that its edges intersect only at their endpoints.Examples are shown below:





All Pair Shortest Path Problem

Let $G = (V, E)$ be a given graph with edge weights $f(e), e \in E$. The sequence $P(u, v)$ of edges $e_1 = (u, w_1), e_2 = (w_1, w_2), \ldots, e_k = (w_k - 1, v)$ is called a "path from the vertex u to v. We say that v is reachable from u if there exists at least one path $P(u, v)$. The total weight of this path is

$$f(P(u,v)) = \sum_{i=1}^{k} f(e_i)$$

For each vertex pair (u, v) such that v is reachable from u, it is required to indicate a path $P*(u, v)$ having the minimum possible total weight: $f*(u, v) = f(P*(u, v)) = minf(P(u, v))$.

2.1 Exploiting average degree Heuristic of vertex in planar graph

Connected planar graphs with more than one edge obey the inequality $2e \geq 3f$, because each face has at least three face-edge incidences and each edge contributes exactly two incidences. It follows via algebraic transformations of this inequality with Euler's formula $v - e + f = 2$ that for finite planar graphs the average degree is **strictly less than 6**. Graphs with higher average degree cannot be planar.

Thus we can make use of the average degree heuristic of a vertex to compute all pairs shortest path with better time complexity than standard Floyd-Warshall algorithm.

The Snowball algorithm computes all-pairs shortest paths in $O(n^2 w_d)$ time where w_d is the tree width of the input graph. In our case of planar graph we know from Planar Seperator Theorem that for a planar graph tree width is $O(\sqrt{n})$. Thus for planar graph Snowball algorithm takes effectively $O(n^2 * \sqrt{n})$ which is an improvement in comparision to Floyd Warshall which takes $O(n^3)$ for planar graph

3. Preliminaries

The Snowball algorithmm for all-pairs shortest paths rely on the fact that the graph has already been made directionally path-consistent (DPC).

3.1 Directed Path Consistency

A graph is said to be consistent when the graph contains no negative cycles. This algorithm takes as input a weighted directed graph $G = (V, E)$ and a vertex ordering d, which is a bijection between V and the natural numbers {1, . . . , n}.

3.1.1 Notations

$w_{i \to j}$: Weight on the arc from vertex i to j
$\{i, j\} \in E$: Existence of an arc between these vertices, in either direction
G_k : graph induced on vertices $\{1, \ldots, k\}$, thus $G_n = G$

3.1.2 DPC Working

In iteration k, the algorithm adds edges (in line 5) between all pairs of lower-numbered neighbours i, j of k, thus triangulating the graph. Meaning if there is a path exist in the graph from i→k and k→j then a direct edge added between i→j. A defining property of DPC is that it ensures that w i→j is no higher than the total weight of this path. This implies in particular that after running DPC, $w_{1 \to 2}$ and $w_{2 \to 1}$ are labelled by the shortest paths between vertices 1 and 2. The edges added by DPC are called fill edges and make the graph triangulated.

3.1.3 Algorithm : DPC (Dechter, Meiri, and Pearl 1991)

Input: A weighted directed graph $G = (V, E)$ and a vertex ordering $d : V \to 1, \ldots, n$
Output: ***CONSISTENT*** if DPC could be enforced on G;
INCONSISTENT if a negative cycle was found

for $k \leftarrow n$ to 1 **do**:
  forall $i < j < k$ **such that** $i, k, j, k \in E$ **do**:
  $w_{i \to j} \leftarrow min\{w_{i \to j}, w_{i \to k} + w_{k \to j}\}$
  $w_{j \to i} \leftarrow min\{w_{j \to i}, w_{j \to k} + w_{k \to i}\}$
  $E \leftarrow E \cup \{i, j\}$
  if $w_{i \to j} + w_{j \to i} < 0$ **then**
    return **INCONSISTENT**
return **CONSISTENT**

3.1.4 DPC Time Complexity

The run time of DPC $O(n * w_d^2)$ where w_d is the induced width which is exactly the highest number of neighbours $j < k$ encountered during the DPC algorithm.

4. Snowball Algorithm

The snowball algorithm computes APSP that has the same worst-case time bounds but is more efficient about it when the input graph is planar and directed path consistent.
The idea behind the algorithm is that we grow a clique of computed distances, one vertex at a time, starting with the trivial clique consisting of just vertex 1. When adding vertex k to the clique, we compute the distance to (from) each vertex $i < k$. We are then ensured by DPC that there exists a shortest path to (from) i that has an edge $\{k, j\}$ for some $j < k$ as its first (last) edge. This means that the algorithm only needs to look "*down*" at lower-numbered vertices.
The name of our algorithm derives from its "*snowball effect*": the clique of computed distances grows quadratically during the course of its operation.

Input : Weighted directed **DPC** graph $G = (V, E)$
Output : Distance matrix D

$\forall i, j \in V : D[i][j] \leftarrow \infty$
$\forall i \in V : D[i][i] \leftarrow 0$
for $k \leftarrow 1$ **to** n **do**
  forall $j < k$ **such that** $\{j, k\} \in E$ **do**
  forall $i \in \{1, \ldots, k-1\}$ **do**
    $D[i][k] \leftarrow min\{D[i][k], D[i][j] + w_{j \to k}\}$
    $D[k][i] \leftarrow min\{D[k][i], w_{k \to j} + D[j][i]\}$
return D

4.2 Theorem: The Snowball algorithm computes all-pairs shortest paths in $O(n^2 w_d)$ time.

Proof: The proof is by induction. After enforcing DPC, $w_{1\to2}$ and $w_{2\to1}$ are labelled by the shortest path between vertices 1 and 2. For $k = 2$ and $i = j = 1$, the algorithm then sets $D[1][2]$ and $D[2][1]$ to the correct values.
Now, assume that $D[i][j]$ is set correctly for all vertices $i, j < k$. Let
$\pi : i = v_0 \to v_1 \to \cdots \to v_{l-1} \to v_l = k$ be a shortest path from i to k, and let
$h_{max} = max\{h|v_h \in \pi\}$. By DPC, if $h_{max} > k$, there exists a path of the same weight where a shortcut $v_{h_{max}-1} \to v_{h_{max}+1}$ is taken.
This argument can be repeated to conclude that there must exist a shortest path π' from i to k that lies completely in G_k and, except for the last arc, in G_{k-1}. Thus, by the induction hypothesis and the observation that the algorithm considers all arcs from the subgraph G_{k-1} to k, $D[i][k]$ is set to the correct value. An analogous argument holds for $D[k][i]$. With regard to the algorithm's time complexity, note that the two outermost loops together result in each of the m_c edges in the chordal graph being visited exactly once. The inner loop always has fewer than n iterations, yielding a run time of $O(nm_c)$ time. From the observation above that $m_c \leq nw_d$, we can also state a looser time bound of $O(n^2w_d)$.

In our case of planar graph we know from Planar Seperator Theorem that for a planar graph tree width is $O(\sqrt{n})$. Thus for planar graph Snowball algorithm takes effectively $O(n^2\sqrt{n})$.

5. Floyd-Warshall algorithm for All Pair Shortest Path

Pseudocode for Floyd Warshall Algorithm

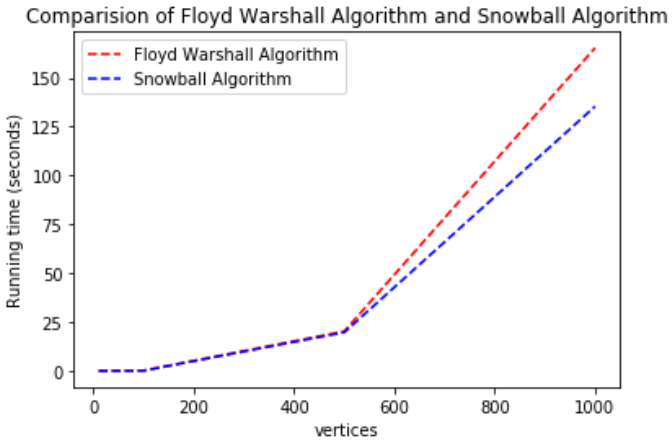
Input : Weighted graph $G = (V, E)$ **Output** : Distance matrix D

```
1  n = G.rows
2  D^(0) = G
3      for k in 1 to n do
4          let D^(k) = (d_{i,j}^(k))
5          for i in 1 to n do
6              for j in 1 to n do
7                  d_{i,j}^(k) = min(d_{i,j}^(k-1), d_{i,k}^(k-1) + d_{k,j}^(k-1))
8  return D^(n)
```

Time and Space Complexity : Floyd Warshall takes $O(n^3)$ time and $O(n^2)$ space to solve APSP for any graph

6. Comparison of Snowball algorithm with Floyd-Warshall algorithm

APSP for planar graph of different vertex sizes (10, 50, 100, 500, 1000) have been used for testing during the course of the this project to compare the performance of snowball algorithm against Floyd-Warshall algorithm and results generated as below.



7. References

- Leon Planken and Mathijs de Weerdt, Roman van der Krogt : Computing All-Pairs Shortest Paths by Leveraging Low Treewidth, https://aaai.org/ocs/index.php/ICAPS/ICAPS11/paper/view/2700/3150 (https://aaai.org/ocs/index.php/ICAPS/ICAPS11/paper/view/2700/3150)
- Planar graph, https://en.wikipedia.org/wiki/Planar_graph (https://en.wikipedia.org/wiki/Planar_graph)
- Planar separator theorem, https://en.wikipedia.org/wiki/Planar_separator_theorem (https://en.wikipedia.org/wiki/Planar_separator_theorem)
- Shortest Paths III: All-pairs Shortest Paths, Matrix Multiplication, Floyd-Warshall, Johnson, https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-introduction-to-algorithms-sma-5503-fall-2005/video-lectures/lecture-19-shortest-paths-iii-all-pairs-shortest-paths-matrix-multiplication-floyd-warshall-johnson/ (https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-introduction-to-algorithms-sma-5503-fall-2005/video-lectures/lecture-19-shortest-paths-iii-all-pairs-shortest-paths-matrix-multiplication-floyd-warshall-johnson/)