International Journal of Computer Applications (0975 – 8887)
Volume 47– No.25, June 2012

# An Improved Algorithm for Finding All Pair Shortest Path

Himanshu Garg
Dept. of Computer Science,
RGGI, UP Technical University,

Paramjeet Rawat
Dept. of Computer Science,
IIMT Engg. College, UP Technical University

## ABSTRACT

Floyd Warshall's Algorithm is a simple and widely used algorithm to compute shortest path between all pairs of vertices in an edge weighted directed graph. It can also be used to detect the presence of negative cycles. Many researchers have given many other approaches for finding all pair shortest path but they reduced the complexity by using complex data structures. In this paper, we suggests a technique for finding shortest path based on Floyd Warshall's algorithm with reduced time complexity and also by not using complex data structures. We present an $O(n^{(3-\in)})$ time algorithm for finding all pair shortest paths. Our proposed algorithm is an improvement on the previous algorithm whose best result was $O(n^3)$

## Keywords

Shortest paths, Floyd-Warshall algorithm, complexity.

## 1. INTRODUCTION

The shortest path problem is the problem of finding path between two vertices (or nodes) in a graph, such that the sum of the weights of its constituent edges is minimized. An example of it can be, finding the quickest way to get from one location to another on a road map. In this case, the vertices represent locations and the edges represent segments of road and are weighted by the time needed to travel to that location.
The single source shortest path is one of the oldest classical problems in algorithm theory. Given a positively weighted directed graph 'G', with a source vertex s, this problem asks for finding the shortest path from 'S' to all other vertices. So it can be considered the mother of all routing problems.
Given a weighted directed graph G = (V, E, w) with two special vertices, a source 's' and a target 't', we want to find the shortest directed path from 's' to 't'. In other words, we want to find the path 'P' starting at 's' and ending at 't' minimizing the function:

$$w(p) = \sum_{e \in p} w(e)$$

Specifically, for every pair of vertices 'u' and 'v', we need to compute the following information:

- dist(u, v) is the length of the shortest path (if any) from u to v;
- pred(u, v) is the second-to-last vertex (if any) on the shortest path (if any) from u to v.

Given a weighted digraph with a weight function w : E→R, R is the set of real numbers that determines the length of the shortest path between all pairs of vertices in G. Given an input, n*n matrix, 'W' represents the edge weights of n vertices; i.e., W= ( $w_{ij}$ ), where

$$w_{ij} = \begin{cases} 0 & if\,(i = j) \\ w(i,j)\,if\,(i \neq j)\,and\,(i,j) \in E \\ \infty \quad if\,(i \neq j)\,and\,(i,j) \notin E \end{cases}$$

## 2. BACKGROUND AND RELATED CONTEXT

Almost all developments concerning this problem have evolved round the famous Dijkstra's algorithm, presented first in 1959. The original version of this algorithm ran in $O(n^2 + m)$ time. The complexity has since then, been reduced to $O(m + n\log^2 n)$ using Fibonacci heaps (Fredman and Tarjan, 1987).

P K Singh, Rajendra Kumar Member, IACSIT and Vijay Shankar Pandey [1] suggest The run time of the algorithm is of order □ □$n^{(5/2)}$), where n is the number of vertices present in the graph.but this algorithm is only based on unweighted and undirected garaph.my approach is based on weighted and directed graph .as well as in the presence of negative weight.
Timothy M. Chan[10] suggest fast matrix multiplication to obtain truly subcubic APSP algorithms for a large class of "geometrically weighted" graphs, where the weight of an edge is a function of the coordinates of its vertices. For example, for graphs embedded in Euclidean space of a constant dimension d, we obtain a time bound near O(n^{3-(3-w)/(2d+4)}), where w < 2.376; in two dimensions, this is O(n^{2.922}). Timothy M. Chan[7] suggest an $O(n^3/\log n)$-time algorithm for a real-weighted directed graph with $n$ vertices for the all-pairs-shortest-paths problem. Further improvement by Tadao Takaoka [12] suggests a in complexity to $O(n^3 \log\log n/\log n)$. Seth Pettie [13] suggests a fundamental comparison-addition model that runs in $O(mn + n^2 \log\log n)$ time, where m and n are the number of edges & vertices, respectively. Yijie Han[9] suggests another approach for all pairs shortest path algorithm with time complexity $O(n^3/\log n)$. Uri Zwick [11] suggest a $O(n^3\sqrt{\log\log n/\log n})$ time algorithm for the All Pairs Shortest Paths (APSP) problem for directed graphs with real edge lengths.

There exist several other algorithms with a better worst case runtime, but these algorithms are much more complicated than the Floyd-Warshall algorithm and involve complicated data structures. Therefore, in many cases the Floyd-Warshall algorithm is still the best choice which has a worst-case runtime of $O(n^3)$ for graphs with n vertices.

## 2.1 Floyd Warshall Algorithm

Let $m_{ij}^{(p)}$ be the weight of a shortest path from vertex i to vertex j with all intermediate vertices in the set {1,2,3,…………,p }[6].

$$m_{ij}^{(p)} = \begin{cases} w_{ij} & if\,(p=0) \\ \min\left(m_{ij}^{(p-1)}, m_{ip}^{(p-1)} + m_{pj}^{(p-1)} \quad if\,(p \geq 1)\right) \end{cases}$$

## Algorithm

```
1. n ⟵ row[W ]
2. $M^{(0)}$ ⟵ W
3. For p ⟵ 1 to n
4.     do for i ⟵ 1 to n
5.         do for j ⟵ 1 to n
6.             do $m_{ij}^{(p)}$ ⟵ min $\left(m_{ij}^{(p-1)}\right), m_{ip}^{(p-1)} + m_{pj}^{(p-1)}$
7. return $M^{(n)}$
```

## 3. PROPOSED ALGORITHM

Our approach suggests that the complexity of the proposed algorithm can be reduced to O($n^{(3-\epsilon)}$). Although, our approach is based upon Floyd Warshall's Algorithm, but with slight improvement.

### Algorithm

```
1.  n← W (rows)
2.  $M^{(0)}$ ← W
3.  initialize counter to 0
4.  counter=counter+1
5.    do $M^{(p)}$ ← $m\,ij^{(p)}$
6.  for i ← 1 to n
7.    do j← 1 to n
8.      do if (i== counter || j== counter)
9.            $M^{(p)}$ [i][j]← $M^{(p-1)}$ [i][j]
10.  else if ($M^{(p-1)}$ [counter][j]+$M^{(p-1)}$
     [i][counter]< $M^{(p-1)}$ [i][j])
11.  $M^{(p)}$ [i][j]←
     $M^{(p-1)}$ [counter][j]+$M^{(p-1)}$ [i][counter]
12. else  $M^{(p)}$ [i][j]← $M^{(p-1)}$ [i][j]
13. return M
```

## 4. RESULT ANALYSIS

For proving that our algorithm is better than the original Floyd's Warshall Algorithm, we have considered three different matrices i.e. 5x5, 10x 10 and 15x15. Then, we have executed both the algorithms on these matrices and found that our algorithm performs better.

To prove the above said statement, suppose we have a n*n size of matrix, where n=5, we will have 5*5 matrix. So there are 6 different matrix that we have to calculate. Suppose the name of matrix is 'M', so we will have to find the values of M0, M1, M2, M3, M4 and M5 matrix. All these matrix will be of size 5*5. According to Floyd Warshall algorithm, the algorithm statement through which we calculate the matrix positions, should run approx 125 times. But with our approach, it will run approx 81 times and the result of matrices obtained after running both the algorithm are same. Likewise, we considered another matrix 10x10 in which the algorithm run's approx 1000 times but with our algorithm it runs for approx 810 times. And for 15x15, Floyd Warshall runs 3375 times and our algorithm runs 2940 times.
The above information is represented with the help of a graph given below:

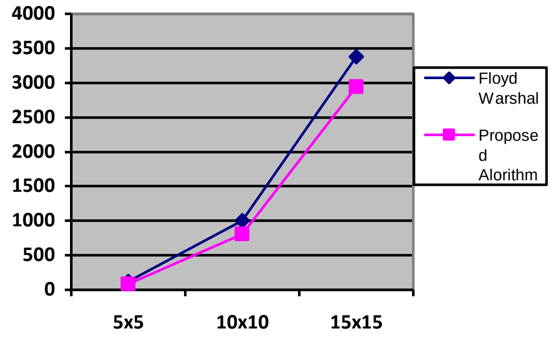

**Fig. 1 Matrices Number Of Execution**

According to the above figure, we can see that Floyd Warshall runs longer then our proposed algorithm. Hence the complexity is decreased using our approach.

## 5. CONCLUSION

In this paper, we described an O($n^{(3-\epsilon)}$) running time algorithm to compute all pairs shortest paths in an weighted and directed graph. At present we have considered only weighted and directed graph but in future we will consider unweighted and undirected graph for reducing the all-pair shortest path complexity.

## 6. REFERENCES

[1] P K Singh, Rajendra Kumar *Member, IACSIT* and Vijay Shankar Pandey,"An Efficient Algorithm for All Pair Shortest Paths", International Journal of Computer and Electrical Engineering, Vol.2, No.6, December, 2010

[2] Stefan Hougardy , "The Floyd-Warshall Algorithm on Graphs with Negative Cycles ",Information Processing Letters 110 (2010), 279-281

[3] Udaya Kumar Reddy K. R, and K. Viswanathan Iyer, "All-pairs shortest-paths problem for unweighted graphs in O($n^2$ log n) time", International Journal of Computational and Mathematical Sciences 3:5 2009

[4] Yijie Han, "An O($n^3$ log log n/ $\log^2$ n) time algorithm for all pairs shortest paths", Manuscript, 2009.

[5] Wikipedia. Floyd-Warshall algorithm — Wikipedia, The Free Encyclope-dia, 2009. [Online; accessed 20-November-2009].

[6] Gary J. Katz1,2 and Joseph T. Kider Jr1 , "All-Pairs Shortest-Paths for Large Graphs on the GPU",Graphics

Hardware (2008) David Luebke and John D. Owens (Editors)

[7] Timothy M. Chan, "All-pairs shortest paths with real weights in O($n^3$ / log n) Time", Algorithmica, 50:236–243, 2008.

[8] Yijie Han," An O($n^3$ (log log n/ log n)$^{5/4}$ ) time algorithm for all pairs shortest paths", Algorithmica, 51:428–434, 2008.

[9] Yijie Han, "A note of an O($n^3$ / log n) time algorithm for all pairs shortest paths", Information Processing Letters, 105:114–116, 2008.

[10] Timothy M. Chan, "More algorithms for all-pairs shortest paths in weighted Graphs", In STOC07, pages 590–598, 2007.

[11] Uri Zwick, "A slightly improved sub-cubic algorithm for the all pairs shortest paths problem with real edge lengths", Algorithmica, 46:181–192, 2006.

[12] Tadao Takaoka, "An O($n^3$ log log n/ log n) time algorithm for the all-pairs shortest path problem", Information Processing Letters, 96:155–161, 2005.

[13] Seth Pettie, "A new approach to all-pairs shortest paths on real-weighted Graphs", Theoretical Computer Science, 312:47–74, 2004.

[14]Tadao Takaoka, "A new upper bound on the complexity of the all pairs shortest path problem", Information Processing Letters, 43:195–199, 1992.

[15] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein, "Introduction To Algorithms" MIT. Press, Mcgraw-Hill Book Company, ISBN 0-262-03141-8, 1990.

[16] Michael L. Fredman, "New bounds on the complexity of the shortest path Problem" ,SIAM Journal on Computing, 5(1):83–89, 1976.

[17] Tadao Takaoka, "A faster algorithm for the all-pairs shortest path problem and its application" In K.-Y. Chwa and J.I. Munro, Springer-Verlag LNCS Vol 3106, pp 278–289.