|  |  |
| --- | --- |
| **GRAPH SPANNER AS A DATA STRUCTURE TO BETTER PERFORMANCE**  **Sharvari Karkare (18MCA1025) | Dr. Jayaram B | VIT University,Chennai**  **Introduction** | |
| A connected weighted graph G=(V,E) and its subgraph G’=(V,E’) is said to be t-spanner of G if the distance of subgraph G’ is at most t times greater than the distance of original graph G i.e.,  G’(V,E’) ≤ t \* G(V,E)  This t is known as the stretch factor. | *JOINING OF VERTICES FROM THEIR NEIGHBOURING CLUSTERS:-*  Repeat all above steps for each vertices of graph G(V’,E’). For each cluster incident on v then move that edge who has least weight from E’ to Es and discard all. |
| **Methodology** |
| **Algorithm: based on greedy strategy**  This algorithm is based on greedy strategy so edges are selected in greedy fashion. This algorithm is similar to Kruskal’s algorithm for computing minimum spanning tree (MST). The edges of graphs are selected in increasing order of their weight. Initially, spanner is null and gradually we adds edges to them. Whether the edge adds or not is depends on the condition that if the separation among u and v (u and v are nodes of graph) in the subgraph initiated by the present spanner edges ES is more than t·weight (u,v), at that point include the edge (u,v) to Es, otherwise reject that edge. As a result, Pt(x,y) would hold for each edge of E that are not present in Es and at the end, the subgraph (V,Es) will be a t-spanner. A result shows that a graph with more than n1+1/k edges must have a cycle of length at most 2k. From above calculation, we say that the length of any cycle in the subgraph (V,ES) must be at least t + 1. Consequently for t = 2k −1, the quantity of edges in the subgraph (V,ES) will be under n1+1/k. Accordingly algorithm I depicted above registers a (2k−1)- spanner of size O(n1+1/k).  **Algorithm: based on very local pproach**  This algorithm is based on very local approach. Suppose weighted graph G(E,V) having k>1 , a (2k-1) spanner takes O(n) time to run. The calculation executes in O(k) rounds, and in each round it basically investigates nearness rundown of every vertex to prune nonessential edges. As a declaration of its straightforwardness, we will show the whole calculation for 3-spanner.  *Computation of 3-spanner:-*  *Formation of clusters:-*  We choose X⸦V sample by selecting each vertex with probability 1/√n. The cluster will form about this sample vertices. At first, Es is null. Each vertex which is not included in cluster is checked by following method:-   1. If the selecting vertex v is not adjacent of sampled vertex, then move each edge that is incident on that selecting edge to Es. 2. If the selecting edge v is adjacent to one or more sample vertices, then select the edge that has minimum distance from v and select all the edge that is incident on v with weight less than the selecting edge and move them to Es.   Then remove those edge (u,v) from E’ where u and v are not from X and are from same cluster. After completing first phase of algorithm, we assume that V’ be the set of vertices that are joining with endpoints of E’ that are left after 1 phase of algorithm. There are two conditions either they are from X having sample vertices or from different clusters. As the result of last step of 1 phase of algorithm, each set of edges E’ are called inter cluster. Then the remaining edges like graph G(V’,E’) and clusters correspond to V’ are moved for phase 2. |
| **Conclusion** |
| In our survey, we focus on two algorithms i.e., greedy strategy algorithm and clustering algorithm which is based on local approach. In first algorithm, time complexity is O(n1+1/k) and in second algorithm, time complexity is O(n). We use these algorithms in our future work. |
| **Result and Future Work** |
| This proposed methodology is to implement the above mentioned algorithms to construct graph spanners and make a comparative study using simulation tools to observe the practical behaviour of the role of graph spanners in congestion avoidance in a given network. We propose to evaluate the performance of a given network upon link failures and also seek to make a comparative analysis on the above algorithms that which one of them performs better to avoid link failures due to congestion. |
| **Contact Details** |
| Sharvari Karkare  Email: [sharvari.karkare@gmail.com](mailto:sharvari.karkare@gmail.com)  8225802514 |
| **References**  [1]Surender Baswana, Sandeep Sen, “Simple Algorithms for Spanners in Weighted Graphs (2003; Baswana, Sen)”  [2] D. Peleg and A. A. Schaffer. Graph spanners. Journal of Graph Theory, 13:99–116, 1989. |



d

w

p

N

M

m

A