Approximate Shortest Distance Computing A Query-Dependent Local Landmark Scheme

ABSTRACT:

Shortest distance query is a fundamental operation in large-scale networks. Many existing methods in the literature take a landmark embedding approach, which selects a set of graph nodes as landmarks and computes the shortest distances from each landmark to all nodes as an embedding. To answer a shortest distance query, the precomputed distances from the landmarks to the two query nodes are used to compute an approximate shortest distance based on the triangle inequality. In this project, this project analyze the factors that affect the accuracy of distance estimation in landmark embedding. In particular, this project find that a globally selected, query independent landmark set may introduce a large relative error, especially for nearby query nodes. To address this issue, this project propose a query-dependent local landmark scheme, which identifies a local landmark close to both query nodes and provides more accurate distance estimation than the traditional global landmark approach. This project propose efficient local landmark indexing and retrieval techniques, which achieve low offline indexing complexity and online query complexity.
EXISTING SYSTEM:

Querying shortest paths or shortest distances between nodes in a large graph has important applications in many domains including road networks, social networks, communication networks, the Internet, and so on. For example, in road networks, the goal is to find shortest routes between locations; in social networks, the goal is to find the closest social relationships such as friendship or collaboration between users; while in the Internet, the goal is to find the nearest server to reduce access latency for clients. Although classical algorithms like breadth-first search (BFS), Dijkstra’s algorithm, can compute the exact shortest paths in a network, the massive size of modern information networks and the online nature of such queries make it infeasible to apply the classical algorithms online.

On the other hand, it is space inefficient to precompute and store the shortest paths between all pairs of nodes. Recently, there have been many different methods for estimating the shortest distance between nodes based on graph embeddings. A commonly used embedding technique is landmark embedding, where a set of graph nodes is selected as landmarks and the shortest distances from a landmark to all the other nodes in a graph are precomputed. Such precomputed distances can be used online to provide an approximate distance between two graph nodes based on the triangle inequality.
PROPOSED SYSTEM:

In this project, this project revisit the landmark embedding approach. According to the findings in the literature, the problem of selecting the optimal landmark set is NP-hard, by a reduction from the classical NP-hard problems such as vertex cover or minimum K-center. As a result, the existing studies use random selection or graph measure-based heuristics such as degree, betweenness centrality, closeness centrality, coverage, and so on. Despite various heuristics that try to optimize landmark selection, all the existing methods follow the triangulation based distance estimation, which estimates the shortest distance between a pair of query nodes as the sum of their distances to a landmark.

As the landmark selection step is query independent, the landmark set provides a single global view for all possible queries that could be diameter apart or close by. Thus, it is hard to achieve uniformly good performance on all queries. As a consequence, the landmark embedding approach may introduce a large relative error, especially when the landmark set is distant from both nodes in a query but the two nodes themselves are nearby.
SOFTWARE REQUIREMENT:

<table>
<thead>
<tr>
<th></th>
<th>Operating System</th>
<th>Windows 2000 / XP, Linux based systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Languages/Software</td>
<td>Java Runtime Environment, Java Software Development Kit 1.6 Java NetBeans IDE</td>
</tr>
</tbody>
</table>

HARDWARE REQUIREMENT:

1. Pentium-4 PC with 20 GB hard-disk
2. 256 MB RAM
3. Keyboard
4. Mouse