Contents

Chapter 1

Introduction

1.1 Background and Motivation

The shortest path problem is a classical combinatorial optimization problem that aims to find a shortest path between two given vertices. There are many applications of this problem and many algorithms have been proposed and used in real situations. Nowadays, it has become necessary to find shortest path distance in various domains like social networks, road networks,communication networks etc. In road network, such as in web mapping, shortest distance between two locations are often desired,in social network, the social connection between two users is the goal and in communication networks, the goal is to find the closest server for connection.And for this purpose, many algorithms have been proposed and devised. For a network with non-negative edge length, the Dijkstra algorithm is the most well-known algorithm which can be implemented to have the running time of O(m+ nlog n), where m and n denotes the number of edges and the number of vertices respectively.For unweighted graphs,shortest paths can be computed using Breath First Search in time O(m+n).These methods require O(n^2) space for storing distance and O(n^3)space to store shortest paths. But with the increasing size of the graph networks, application of these classical methods has become difficult and infeasible due to the online nature of the queries.

Hence, to achieve the desired goal of finding shortest distance in very large graphs , *landmark embedding technique* has been devised. In this process, a set of graph nodes called landmarks are selected and distance of all other nodes are precomputed from each landmark and these distances are then used to calculate the approximate distance based on the triangle inequality. In this report,this method has also been introduced with two modifications to improve the approximation error, namely (i)*Least Common Ancestor* and (ii)*Shortcut Paths*

1.2 Basic Definitions

It is necessary to define and pre-determine the notations that are to be used in the forthcoming sections for implementation,application and usage purpose.

1.2.1 Notations

Let $G = (V, E)$ denote a graph having $|V| = n$ vertices and $|E| = m$ edges.For simplicity, the graphs that are considered here are undirected and unweighted graphs i.e weight of all the edges is unity. .The *distance* between two vertices s and t in a graph is the length of the shortest path between s and t and it is denoted by $d(s, t)$ where $s, t \in V$. The shortest path,denoted by $\pi_{s,t}$,is the minimum number of nodes one has to go through to reach the second node from the first one (or vice versa, undirected). So, $\pi_{s,t}$ contains the sequence of vertices that are traversed to move from s to t. So, if $\pi_{s,t}$ is a path of length d then it can be expressed as $\pi_{s,t} = \{s, v_1, v_2, .., v_{d-1}, t\}$,where $\{v_1, v_2, ..., v_{d-1}\} \subseteq V$ and$\{(s, v_1), (v_1, v_2), .., (v_{d-1}, t)\} \subseteq E$.The path length of $\pi_{s,t}$ is denoted by $|\pi_{s,t}|$. The concatenation of two paths is $\pi_{s,t}$ and $\pi_{t,u}$ is $\pi_{s,u} = \pi_{s,t} + \pi_{t,u}$.

Degree of a vertex is the number of immediate neighbors it has or the number of edges that pass through the vertex. Higher the degree of a vertex, more would be the probability of the edges to pass through that vertex.This means that most of the paths that are computed have the vertex with the maximum degree.

Landmark is a vertex which is selected from the graph $G(V, E)$ to act as an intermediate via which the distance between two vertices would be calculated.This can be explained as instead of computing a direct path between vertices s and t,where the path length is denoted as $d(s,t)$, a path via the landmark l is calculated between s and t whose distance is denoted as $d_l(s,t)$ where $s,t,l \in V$.

1.2.2 Distance Bounds

The *triangle inequality* holds true for shortest distance in a graph, since it is a metric.Given that $s,t,u \in V$,then the following inequality holds:

$$d(s,t) \leq d(s,u) + d(t,u)$$

The equality sign holds when the shortest path between s and t passes through u.

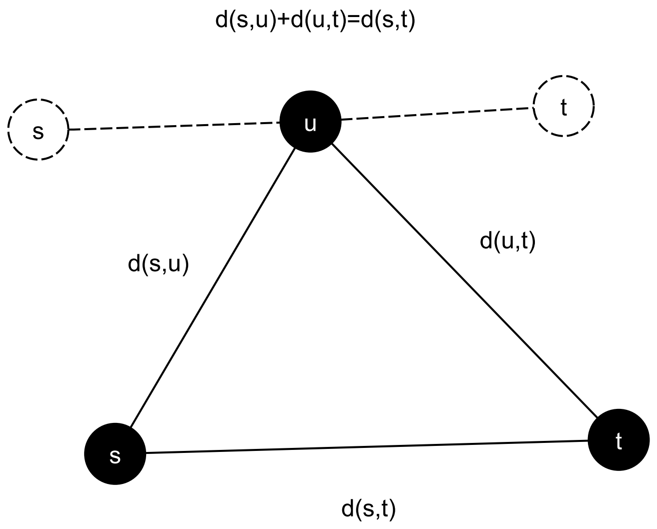


Figure 1.1: Illustrates the triangle inequality case for graph nodes

So this states that the exact distance calculated between the nodes using online methods such as Dijkstra or BFS is always less than or equals to the approximate distance calculated between them using the landmark-based methods.

Chapter 2

Algorithm Description

2.1 Introduction

In this section, we show the detail of the implementation and report the results of landmark-embedded distance estimation method. Landmarks are a few selected vertices from the graph in terms of which the shortest distance of all other vertices in the graph are measured. First, we describe the basis for selection of landmarks and its significance.The method of selection of the landmarks plays an important role in computing the approximate distance which is explained in following section. After the landmarks are selected on basis of the desired methods,it is noted that the complexity involved is much better than those in case of exact methods.Since the landmark-embedded distance estimation is an approximation method, it has a percentage of error related to it as compared to the distance that is calculated by exact methods namely Dijkstra's algorithm. Due to the less time complexity of the approximation methods these are preferred to exact methods neglecting the small relative errors[4]. The algorithm for Dijkstra's method has been explained in Algorithm 1.

Algorithm 1 :Dijkstra's Algorithm

```
Require: Graph $G = (V, E)$
 1: procedure DIJKSTRA(source)
 2:     for $v \in V$ do
 3:         distance[$v$] = infinity
 4:         predecessor[$v$] = undefined
 5:     end for
 6:     distance[$source$]= 0
 7:     $H$= the set of all nodes in Graph
 8:     while $H$ is not empty do
 9:         $u$ = vertex in $H$ with smallest distance in distance[]
10:         if distance[$u$] = infinity then
11:             break
12:         end if
13:         for each neighbor $v$ of $u$ do
14:             $alt = distance[u] + dist(u, v)$        ▷ for unweighted graph dist(u,v)=1
15:             if alt is less than distance[v]  then
16:                 distance[$v$] = $alt$
17:                 predecessor[$v$] = $u$
18:                 decrease-key $v$ in $H$
19:             end if
20:         end for
21:     end while
22:     OUTPUT
23: end procedure
```

Algorithm 1 :Dijkstra's Algorithm (continued)

```
24: function PrintPath($node$)
25:     if $node = source$ then
26:         Print ($node$..)
27:     else if No path from $node$ to $source$ then
28:         Print ($source$,$node$)
29:     else
30:         PrintPath(predecessor of $node$)
31:         Print($node$)
32:     end if
33: end function
34: function Output
35:     for $i = 1 : |V|$ do
36:         if $i = source$ then
37:             Print($source, source$)
38:         else
39:             PrintPath($i$)
40:         end if
41:         Print(distance($i$))
42:     end for
43: end function
```

2.2 Landmark selection

In this section, the basis of selecting landmarks is explained[2]. Let k denote the number of landmarks we would like to choose.The landmarks can be chosen on the following basis: (i) Random vertices are selected. (ii)Highest Degree nodes are selected.

Landmarks having higher degrees have more neighbors as a result of which ,their distance to the vertices of the queries have a greater possibility of having small values and can hence generate relatively shorter distance values as compared to the landmarks selected randomly.The distance calculated by the landmarks selected on the basis of decreasing degrees are closer to the exact distance calculated by Dijkstra Algorithm and hence have relatively less error and hence preferred to random selection methods.

Algorithm 2 :Landmark Selection Schemes

```
Require: Graph $G = (V, E)$, k number of landmarks
 1: function Random-Selection
 2:     D={$\phi$}
 3:     for $i = 1 : k$ do
 4:         Randomly select a vertex from $V - D$ and list it as $l_i$.
 5:         D $\leftarrow l_i$
 6:     end for
 7:     return $D = \{l_1, l_2, ...l_k\}$
 8: end function
 9: function Highest-Degree
10:     for $v \in V$ do
11:         $deg[v] \leftarrow$ Degree($v$)
12:     end for
13:     Sort $V$ by $deg[v]$ in decreasing order
14:     Let $v_i$ denote the vertex with $i^{th}$ largest degree
15:     return $\{v_1, v_2, ..., v_k\}$
16: end function
```

2.3 Implementation

From D = {set of k landmarks},a landmark $l \in D$ is fixed and we calculate the distance of all the vertices $v \in D$ in the graph from vertex l.This distance so calculated is denoted and stored as $d_l(v)$.

For each query (s, t) the approximate distance as measured from a landmark $l \in D$ is given as

$$d_l(s, t) = d_l(s) + d_l(t) = d(l, s) + d(l, t)$$

To obtain the approximate distance between the vertices in a query, distance is calculated from each of the k landmarks and the minimum is considered as the estimated distance.This is denoted as $\tilde{d}(s,t)$ as

$$\tilde{d}(s,t) = \min_{l \in D}(d_l(s,t))$$

Using Triangle Inequality, it can be shown that the approximated distance values are always greater than the exact distance.Because

$$d(s,t) \leq d(s,l) + d(l,t)$$

This method of estimation is known as *global landmark method* since a global set of landmarks are selected and computations are done.The method is applied for different number of landmarks i.e for k ranging from 20 to 100.

Algorithm 3 describes the steps involved to calculate the global landmark-based approximated distance.

2.4 Improved Landmark-Based Algorithms

2.4.1 Lowest Common Ancestor Method

In this section, the performance of landmark-based method is improved by introducing a small change. Instead of using a global set of landmarks, query-specific landmarks are selected and hence distance is estimated.These query-specific landmarks are closer to the $\pi_{s,t}$ than other global landmarks.

As we elaborate, it can be seen that to calculate the approximate distance, we compute $d_l(s) + d_l(t)$, where l∈D and (s,t)∈Q .In Figure 2.1, it is clear that the path from the landmark to the least-common-ancestor of s & t is added up twice and hence is subject to error.[1, 6]

Algorithm 3 :Global-Landmark Based Estimation Method

```
Require: Graph $G = (V, E)$, set of k landmarks,$Q = \{set\ of\ queries\}$
 1: procedure PRE-COMPUTE DISTANCE
 2:     D ← SELECT-LANDMARK($G$,$k$)                                    ▷ Algorithm 2
 3:     for $l \in D$ do
 4:         Do Dijkstra Algorithm for source =$l$.                       ▷ Algorithm 1
 5:         Store distance of $v \in V$ from l and store them as $d_l(v)$ in a matrix.
 6:     end for
 7: end procedure
 8: procedure BASIC-LANDMARKS-DISTANCE
 9:     for $(s, t) \in Q$ do
10:         Calculate $d_{approx}(s, t) = \min_{l \in D}(d_l(s) + d_l(t))$
11:     end for
12: end procedure
```

The actual estimated distance can be expressed as

$$d_{lca}(s, t) = d_l(s) + d_l(t) - 2d_l(r)$$

, where r= least-common-ancestor of s & t This problem arises because there is a common set of k landmarks for all the queries amongst which the minimum distance is selected.This global set of landmarks when used for computation of approximate distance may result is re-traversal of same common path twice. So in order to eliminate this error, local landmarks corresponding to each of the queries are identified using the global landmark as the origin.

For example, the shortest paths from landmark l to the query (s, t) are:

$$\pi_{l,s}(l, v_1, v_4, s) \quad and \quad \pi_{l,t}(l, v_1, v_3, v_2, t)$$

Here, it can be observed that the path from l to v_1 is common for both vertices and hence is added twice. Therefore ,it is necessary to eliminate common paths to decrease error.Hence, there is a need to locate the least-common ancestor of the vertices of the

query.
Of both, the minimum number of intermediate vertices is 3. Designating each of the intermediate vertex by a depth value, we get that : for $\pi_{l,s}$, depth 1 has v_1 and depth 2 has v_4.Likewise, for $\pi_{l,t}$, depth 1 has v_1 and so on. As we travel backwards from s and t towards l, we find that at depth 1 the first common vertex v_1 is found which is the least-common-ancestor.Pictorially,it can be represented as in Figure 2.1

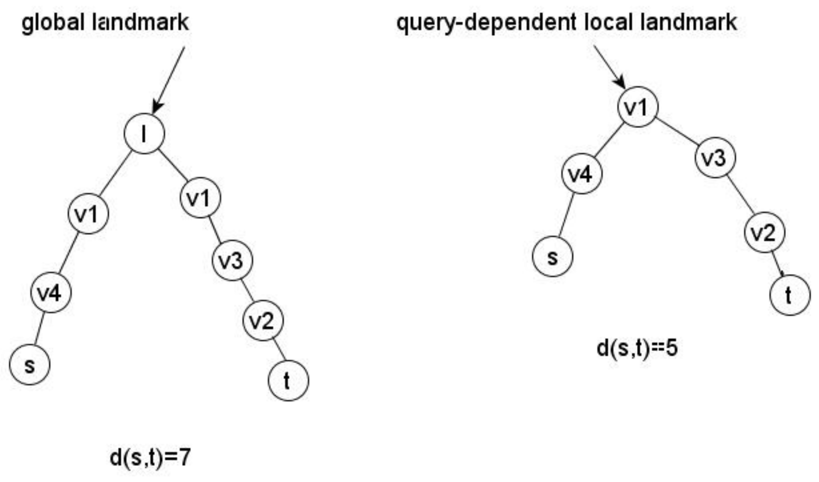


Figure 2.1: *Illustrates that the distance computed from the query dependent local landmark is shorter as computed from a global landmark.The LCA of(s,t) serves as the query dependent local landmark*

Here, we find that the if query-dependent local landmark is used for distance estimation then better results are obtained.Algorithm 4 describes the method to identify the LCA and calculate the improved approximate distance.

For each of the global-landmark, the shortest-path-tree to each of the vertices of a query is constructed and the least common ancestor of both the vertices is identified by traversing upwards from the vertices towards the global-landmark (origin) and checking the vertices at same depth. The first common vertex found is the least-common-ancestor (LCA) and this is considered as the local-landmark for that particular query. Similar pro-

Algorithm 4 Local-Landmark Approximation Method

```
Require: Graph $G = (V, E)$, k landmarks, $Q = \{set\ of\ queries\}$
 1: procedure PRE-COMPUTE-PATH
 2:     D ← SELECT-LANDMARK($G$,$k$)                                   ▷ Algorithm 2
 3:     for $l \in D$ do
 4:         Do Dijkstra Algorithm for source =$l$.                     ▷ Algorithm 1
 5:         for $(s, t) \in Q$ do
 6:             Store paths to $s$ and $t$ as $\pi_{l,s}$ and $\pi_{l,t}$.
 7:             Add these paths to set Π
 8:         end for
 9:     end for
10: end procedure
11: procedure DISTANCE-LCA
12:     for each path in Π do
13:         Find the LCA for each pair of queries and reset the paths $\pi_{l,s}$ and $\pi_{l,t}$
14:         Compute the distance $d_l(s, t) = d_{\tilde{l}}(s) + d_{\tilde{l}}(t)$, where $d_l(s)$ and $d_l(t)$ are the new
    distance of $s$ and $t$ to the newly found LCA,$\tilde{l}$
15:     end for
16:     for $(s, t) \in Q$ do
17:         $d_{approx}(s, t) = \min_{l \in D}(d_l(s) + d_l(t))$
18:     end for
19: end procedure
```

cedure is carried out for all the remaining landmarks and finally the minimum distance hence obtained is the approximate distance.These local landmarks and their paths to the queries are shorter as compared to the paths traversed by the global landmarks which provide a tighter upper bound in the triangle inequality and so, less error.

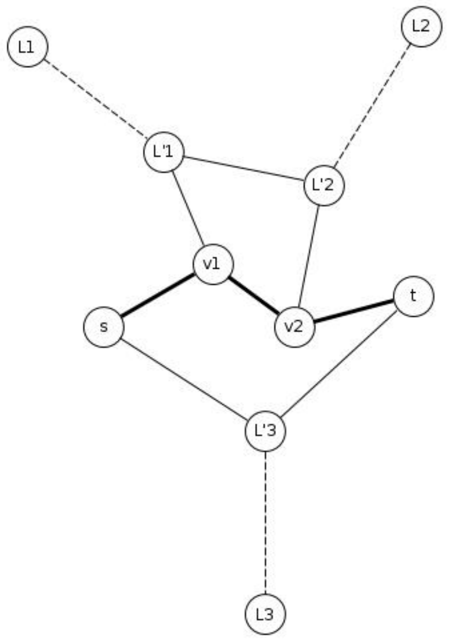


Figure 2.2: *Local landmarks have paths closer to the shortest paths as compared to global landmarks*

As in Figure 2.2, it can be clarified that $D = \{L1, L2, L3\}$ is a set of global landmarks and their paths to query (s, t) are as shown. It is also found that vertex $L'1$ is the local landmark specific to query (s, t) with respect to the global landmark $L1$.Similarly, $L'2$ and $L'3$ are local landmark to (s,t) as compared to the global landmark $L2$ and $L3$ respectively.

2.4.2 Shortcut Method

Apart from using landmark as an intermediate, there can exist a shortcut path between the two nodes of a query.This path corresponds to the vertices which are common neighbors of those nodes lying in the shortest path trees $\pi_{l,s}$ and $\pi_{l,t}$. The vertices lying on the SPT of the two nodes of the query have certain immediate neighbors. It is possible that there must be a few neighbors which are common to both the SPTs. And hence can generate a shortcut path between the two nodes and can give an alternate path between them which may have distance value less than the previous implementation methods.[1, 3, 7]

As we have seen in the example described in the previous section, the shorter paths that was found were

$$\pi_{l,s}(v_1, v_4, s) \quad \textit{and} \quad \pi_{l,t}(v_1, v_3, v_2, t)$$

Here the distance between s and t is 5.

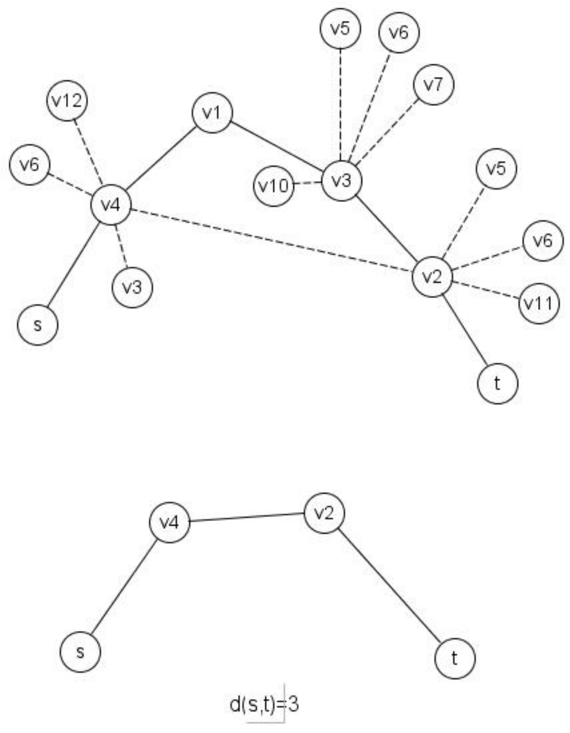


Figure 2.3: *Illustrates the neighbors of the intermediate vertices on the SPT (by dashed lines) and the shortcut path for (s,t)*

Now it is found that v_2 and v_4 share a common edge, then a shortcut path can be developed between s and t via these two nodes. Hence, the path would be shorter in length :

$$\pi_{s,t}(s, v_4, v_2, t)$$

Now the path length can be seen as 3.Since this value would be closer to the true distance this would hence generate less error and therefore, improves the LCA distance estimation method. (See Figure 2.3)

In order to locate the shortcuts, all pairs of vertices of $\pi_{l,s} \times \pi_{l,t}$ are needed to be analyzed to find the common neighbors,if any.If the pairs have any common edge, then the edge providing the best distance estimate is considered to be the shortcut-path length which is then compared with the true distance for error calculation.

Algorithm 5 : Shortcut Paths Method

```
Require: Graph $G = (V, E)$, k landmarks, $Q = \{set\ of\ queries\}$
 1: procedure PRE-COMPUTE-PATH
 2:     D ← SELECT-LANDMARK($G$,$k$)                        ▷ Algorithm 2
 3:     for $l \in D$ do
 4:         Do Dijkstra Algorithm for source =$l$.          ▷ Algorithm 1
 5:         for $(s, t) \in Q$ do
 6:             Store path to $s$ and $t$ as $\pi_{l,s}$ and $\pi_{l,t}$.
 7:             Add these paths to set $\Pi$
 8:         end for
 9:     end for
10: end procedure
```

Algorithm 5 : Shortcut Paths Method(continued)

```
11: procedure Shorcut-Distance
12:     for path in $\Pi$ do
13:         for each pair of vertices in $\pi_{l,s} \times \pi_{l,t}$ do
14:             if any common neighbor(common edge) is found then
15:                 calculate distance between the two nodes of the query and store it in $d_l(s,t)$
16:             end if
17:         end for
18:     end for
19:     for each query in $Q$ do
20:         Find the best distance(shortest distance) estimated amongst all the landmarks.
21:     end for
22: end procedure
```

Chapter 3

Experimental Evaluation

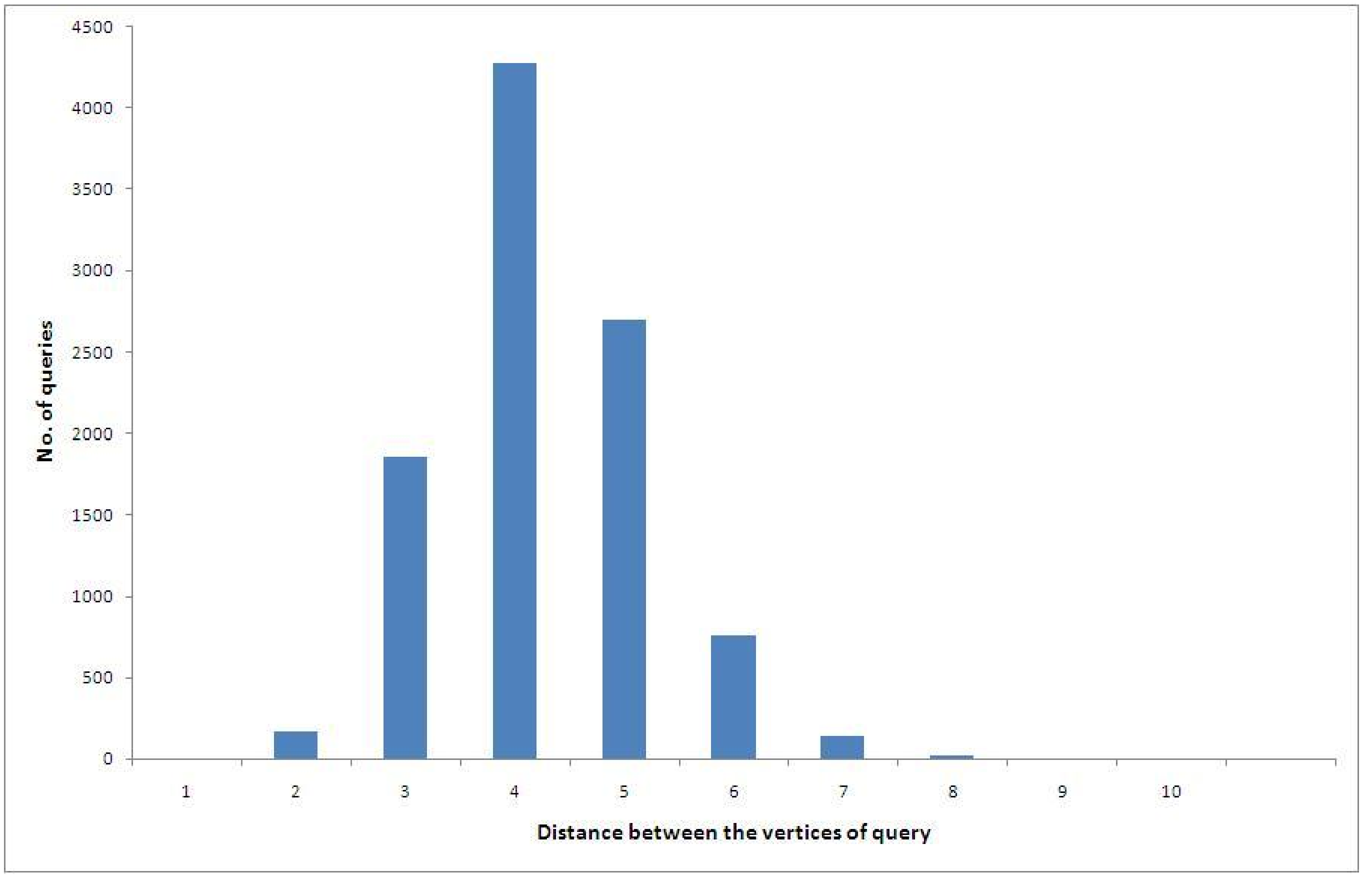
3.1 Datasets

The experiment is conducted to find the shortest distance between a given pair of nodes known as query. In order to get a very large graph to carry out this experiment, a sample dataset from Facebook is obtained. Facebook is a social networking site having over billions of users. Here, the users are connected to each other as 'friends' and many are yet to be connected amongst each other.
In this dataset, the users have been assigned numbers and are the vertices,V of this very large graph. Theses vertices are connected amongst each other on the basis of the fact that they are socially connected on the website.Their connections are the edges of the graph which are denoted by E.Here, only a small section of the huge database is used as a dataset for the experiment.
With the help of proper coding it was computed that the dataset had 63731 users(vertices).So, the experiment is carried out on a graph having 63731 vertices and over a million number of edges, which is a large graph.

3.2 Experimental Setup

In order to evaluate the performance of the algorithms, first a random sample of 10000 vertex pairs are selected.Each of these pairs, (s, t) is referred to as a *(*query) where

$u, v \in V$. This query set is denoted by Q. The same set of queries is used for all the algorithms for performance evaluation. For each of the pair in Q, the actual distance between the nodes in the query is calculated using the Dijkstra's Algorithm. One by one, each of the approximation methods are applied to evaluate the approximate distance for each query, which is then used to calculate the approximation error and time taken to calculate these values.
Approximation error is calculated as $(d' - d)/d$ where $d =$ exact distance calculated and $d' =$approximate distance calculated.
Query Time is the time taken to calculate the approximate distance between two nodes of a query. The true distance is calculated and has been represented in the following histogram to list the frequency in which distance values actually lie.



3.3 Results

3.3.1 Approximation Error

Implementation of the methods on the dataset reveal the difference between the true distance and the estimated distance values. Table 3.1 shows the relative error for random and degree-based selection of global landmarks and local landmarks and for the shortcut paths.

For randomly selected landmarks, it is found that relative error is more in global landmarks as compared to the query dependent local landmarks. This is because local landmarks have path shorter than the global ones.But the global landmarks perform much better when the landmarks are selected on the basis of degree.

But for the local landmarks, the relative error is less in case of randomly selected landmarks because the degree based landmarks are already in their best case and hence tend to provide less improvements in the results. However,the approximation error is minimum in case of shortcut method for distance estimation. Also it is observed that the relative error decreases as the number of landmarks selected is increased i.e for k = 20 ,40, 60, 80 ,100 it is found that the relative error was decreasing for all cases.The results have been summarized and shown in Figure 1.

Table 3.1: Relative Approximation Error

Landmark Selection	*Methods*	k= 20	40	60	80	100
Random Landmarks	GL	0.0805	0.0602	0.0445	0.0406	0.0368
	LL	0.0127	0.0061	0.0001	0.0001	0.0001
	Shortcuts	0.0061	0.0039	0.0001	0.0001	0.0001
Degree − Based Landmarks	GL	0.0097	0.0084	0.0076	0.0067	0.0059
	LL	0.0062	0.0043	0.0037	0.0031	0.0028
	Shortcuts	0.0032	0.002	0.0001	0.0001	0.0001

3.3.2 Query Time

It is observed that the query time increases as the approximation error decreases.More improvement in the method refers to more computations being carried out and hence the time taken to calculate the distance between the nodes in a query is higher.Query time for shortcut-path method is more as compared to other methods. For randomly selected landmarks the time taken by the local landmark method is about 3 times the query time in case of global landmarks for all values of k. For landmarks selected on the basis of degree, the time taken by the local landmarks is also around 3 time the global landmarks.But the shortcut method has maximum query time as compared to other methods. This has been summarised in Table 3.2.

Table 3.2: Query Time (in ms)

Landmark Selection	*Methods*	k= 20	40	60	80	100
Random Landmarks	GL	0.021	0.038	0.06	0.076	0.1006
	LL	0.07	0.149	0.219	0.292	0.368
	Shortcuts	12.039	21.648	31.562	41.368	51.086
Degree − Based Landmarks	GL	0.017	0.037	0.057	0.073	0.093
	LL	0.06	0.119	0.182	0.247	0.31
	Shortcuts	6.493	14.696	23.559	32.746	41.249

The following graph shows the growth of query time and approximation error with increasing number of landmarks chosen.It is observed that the query time increases with the number of landmark chosen. It is seen that approximation error decreases as the computations are done with more number of landmarks.

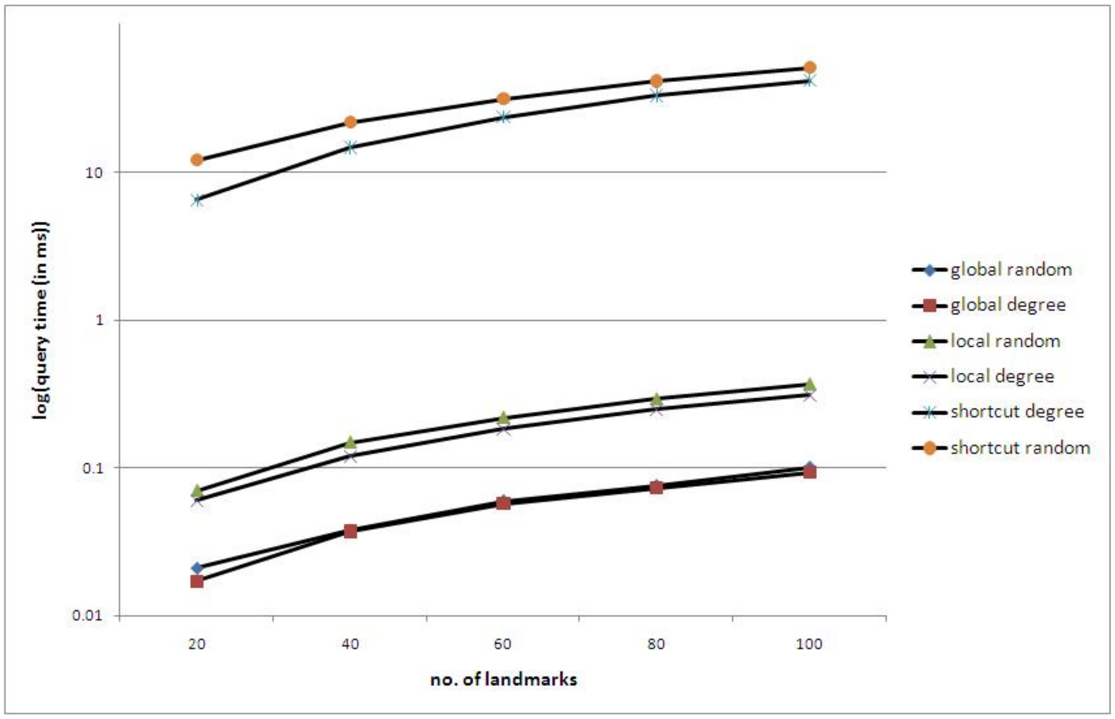


Figure 3.1: Query Time

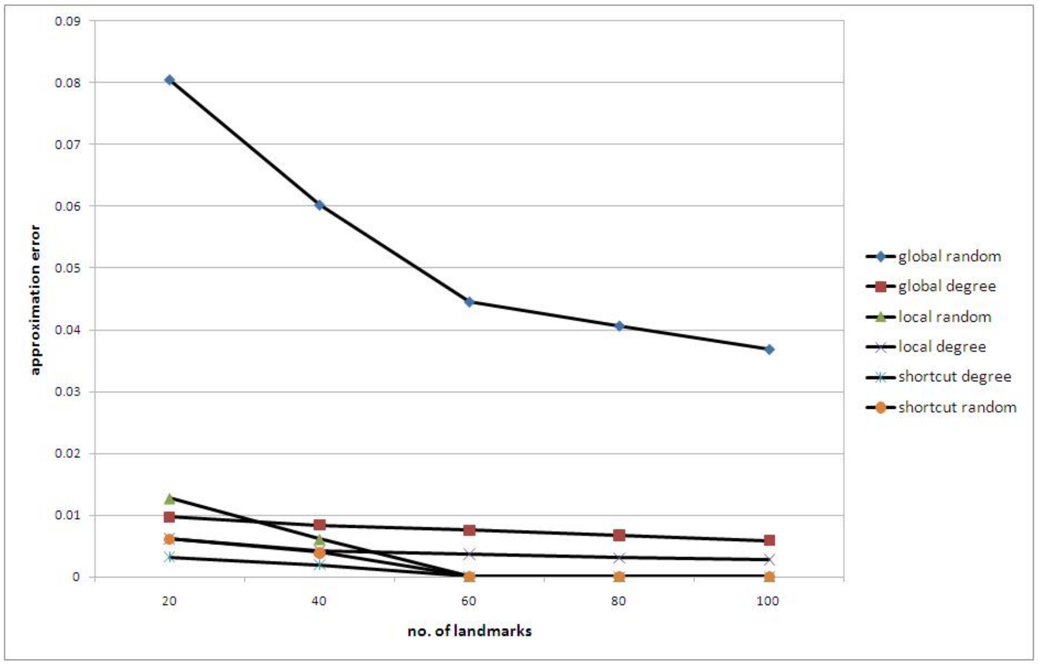


Figure 3.2: Approximation Error

Chapter 4

Conclusion

Taking into account all the observations hence obtained in the implementation of different methods on to the given dataset in order to calculate shortest path distance between the nodes, it is found that the *Shortcut Method for Distance Approximation* and *Local-Landmark Based Distance Approximation* provide the best results which are closest to the distance calculated by the Dijkstra's Algorithm.But of the two methods, the one with best approximation error and query time is the local landmark based distance approximation where the landmarks are sellected randomly.
It can hence be concluded that for large graphs where queries tend to be online, these classical methods do not serve the purpose because the time taken by them to do the computations and the disk space consumed to store the necessary values is very large as compared to those by the landmark-based approximation methods. As we can see, the landmarks that are selected are much fewer in number as compared to the number of nodes in the graph.These landmarks are then used to find their shortest paths and distances to all the other nodes present in the graph.Then depending upon the query requested, the desired distances and paths with respect to each landmark are considered for further analysis to generate the shortcut distances. So in short it can be described that in classical methods the Dijkstra algorithm has to be run as many times as there are number of queries while in the landmark based methods, the Dijkstra Algorithm has to be run k number of times where $k =$Number of landmarks selected which is however the precomputation part, and the real computation time only depends on searching,retrieval

and small adjustments in the paths to locate neighbors with common edges. It has been found that the query time per query is 34.02 ms for Dijkstra's Algorithm whereas for the local-landmark based method, the query time per query is about 0.22 ms with 60 landmarks selected and having a relative error of 0.0001. If the number of landmarks is increased further there is no significant improvement in the relative error but it marks as increase in query time.Similar is the case for shortcut-path method.Although there is less approximation error for lower number of landmarks, but the query time is very high as compared to the local-landmark based methods.

Hence, the final inference is that *Local-Landmark Based Distance Estimation* method with landmarks selected randomly provide results with less approximation error and query time and therefore can be used for distance computation between nodes in very large graphs.

Bibliography

[1] Miao Qiao, Hong Cheng, Lijun Chang, Jeffrey Xu Yu. Approximate Shortest Distance Computing: A Query Dependent Local Landmark Scheme. Department of Systems Engineering and Engineering Management The Chinese University of Hong Kong Hong Kong, China

[2] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis. Fast shortest path distance estimation in large networks. 2009. ACM.

[3] K.Tretyakov, A.A. Cervantes, L.G. Bañueios, J.Vilo, M.Dumas. Fast-Fully Dynamic Landmark-Based Estimation of Shortest Path Distances in Very Large Graphs.

[4] Mark Allen Weiss.Data Structures and Algorithm Analysis in C Efficient C Programming: A Practical Approach in C,1997,ISBN:0-201-49840-5

[5] Riivo Kikas.Shortest Path Distance Estimation in Large Graphs.Department of Computer Science,University of Tartu.

[6] Miao Qiao, Hong Cheng, and Jeffrey Xu Yu. Querying Shortest Path Distance with Bounded Errors in Large Graphs. Pages 256-258 Department of Systems Engineering and Engineering Management The Chinese University of Hong Kong Hong Kong, China

[7] Volodymyr Floreskul.Memory-Efficient Fast Shortest Path Distance Estimation in Large Graphs