IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

REVOLUTIONARY METHOD FOR SHORTEST PATH IDENTIFICATION THROUGH OPTIMIZATION STRATEGY

Mrs.C.Ajitha[1], Mrs. Akila M [2].,
Assistant Professor , Assistant Professor
Computer Science and Engineering
Unnamalai Institute of Technology
Tamilnadu, India.

ABSTRACT:Our paper presents a new method for finding the top-k shortest paths between nodes in a graph. We combine graph preprocessing with the Bellman-Ford algorithm and introduce a top-k path join technique, enhanced with optimization strategies like early join constraint and lazy pruning. Our approach outperforms existing methods in efficiency and time complexity. Future work will explore different shortest path algorithms for further improvement.

IndexTerms: Path join method , Early join constraint, Lazy pruning

1. INTRODUCTION

Data mining, an interdisciplinary subfield of computer science, is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. Aside from the raw analysis step, it involves database and data management aspects, data preprocessing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating.

Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.Data mining, also known as Knowledge-Discovery in Databases (KDD) is the process of automatically searching large volumes of data for patterns.Most of the successful business decisions are made from reliable data source and their validation through the application of tools and techniques. Most of the literature on data mining focuses on its benefits and burdens in making business decisions.

1.1 DATA MINING ELEMENTS

- Extract, transform, and load transaction data onto the data warehouse system.
- Store and manage the data in a multidimensional database system.
- Provide data access to business analysts and information technology professionals.
- Analyze the data by application software.
- Present the data in a useful format, such as a graph or table.

1.2 TOOLS USED FOR DATA MINING

1. Artificial neural networks
i.Non-linear predictive models that learn through training and resemble biological Neuralnetworks in structure.
2. Decision trees
i.Tree-shaped structures that represent sets of decisions. These decisions generate rules for the classification of a dataset.
3. Rule induction

i. The extraction of useful if-then rules from data based on statistical significance.

4. Genetic algorithms

i. Optimization techniques based on the concepts of genetic combination, mutation, and natural selection.

5. Nearest neighbor

i. A classification technique that classifies each record based on the records most similar to it in an historical database.

1.3 OBJECTIVE

To propose holistic methods of top-k path join strategy for group relationship analysis This is used to discover k shortest paths between a pair of node sets.Also to propose efficient method to find the constrained top- k shortest paths between the two virtual nodes .This is used to reduce the cost of candidate path searching.Two optimization techniques used in top-k for further improvements which is used to pruning the search space with determine threshold in each candidate path generation.

1.4 OVERVIEW

Graphical data representation and analysis is used in a number of viable applications such as ontology graphs, biological and chemical pathways and social networks, etc. On the other hand, nodes in the graphs also become more complex that tends to numerous graph analysis problems such as centrality computation, node separation, and community detection all rely on the simple nodedistance (length of shortest path) primitive, which scales badly with graph (or network) size.. Moreover, applications on graphs require two things:

- Granularity of nodes
- Operations on node sets

Graph search is a fundamental aspect of graph management, crucial for identifying sub-graphs that serve specific purposes, such as finding the shortest path between two nodes, determining the minimal spanning tree, or solving the traveling salesman problem. This study places particular emphasis on the shortest path identification problem due to its widespread applicability. Shortest path discovery is integral to various applications, like analyzing social networks to understand relationship dynamics between individuals. The primary objective of this research is to address the top-k shortest paths problem, which entails discovering the k shortest paths between a pair of nodes in a graph in non-descending order of their costs. This problem holds significant relevance, especially when multiple paths between a node pair are desired. For instance, in a large social network, users might be interested in exploring all accounts within the top-k shortest paths between two sensitive accounts. While numerous approaches exist for tackling this problem, continuous improvements are necessary to efficiently handle large graphs. For example, consider Figure 1,where the shortest path distance between nodes A and B is 3 in the graph, while the Euclidean distance between their coordinates is 3.1.

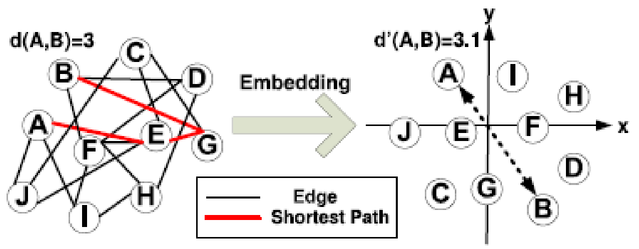


Figure 1 SAMPLE GRAPH-SHORTEST PATH DETERMINATION

While discussing about group analysis, a motivation case comes into effect. Let S and T be the two node groups that represent two criminal gangs in a social network. While there is intent to find the top-k simple shortest paths between the two groups that may show the hubs by which these groups are connected. The path which contains more than one node in S or T brings no new information to the results. In this work, a novel approach has been introduced called top-k path join addressing the above problem. Typically, given two node sets S and T, the top-k path join discover a list of paths P in an ascending order of their lengths.

The major contributions of the process are to find an optimized join strategy and to improve the performance in the top-k shortest path discovery for each one node pair. An efficient method to find the constrained top-k shortest paths between the two virtual nodes has been designed. The result of the proposed work is compared with the existing classic Yen's algorithm. Further, the performance has been improved by two optimization strategies namely; push join constraint and pruning search space. Experimentation is conducted on synthetic dataset and compared with other query evaluation strategies.

1.5 GRAPH NOTATION

Let$G = (V, E)$ be a weighted, directed graph, where V is a set of nodes and E is a set of edges. Each node comprises a unique node id and other attributes and each edge is represented by $e = (u, v), u, v \epsilon V$that has a non-negative weight. By general description, an edge from node u to v is called u's outgoing edge and v's incoming edge. Typically, a path is defined as a sequence of edges (u_1,u_2), (u_2,u_3),…, (u_{z-1},u_z) where $ui(1 \leq i \leq x)\epsilon V$. The number of nodes in the path P is denoted by nodes (p) and the ith $(1 \leq i \leq nodes\ (p))$ node is the path referred as p[i]. The length of a path P is defined as the sum of the weight of its constituent edges. Let P(s,t) represent all simple paths starting from node s and ending at node t. the shortest distance between s and t in a graph G, δ_G (s,t) is defined as the minimal length of all paths in P(s,t).

1.6 GRAPH PREPROCESSING

The proposed work is started with node creation according to the aforementioned graph notations in hierarchical order. Following that, edges are drawn between those nodes. A complete weighed DAG is formed by allotting weights for each edge. The generic flow of the proposed work is given below in the Figure 2.

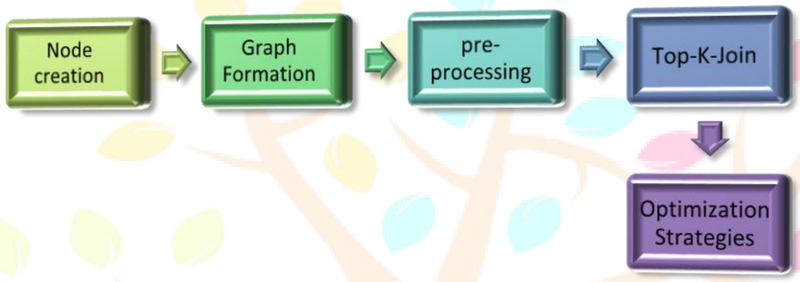


Figure 2 Generic Flow

Specifically, in this process, a virtual source node v_s and a virtual target node are initialized. Then, the shortest path tree spt(v_t,G) has been built using Bellman-ford algorithm and also the shortest distance from node u to v_t with u.cost is recorded, which is followed by the assessment of edge cost on each edge for each node in the graph based on the steps presented below. The output obtained is the transformed graph G_{tran}.

1. Form Graph G with source nodes S and target nodes T
2. Initialize a virtual node v_s and zero-weighted edges from $v_s$to each node in S
3. Initialize a virtual node v_t and zero-weighted edges from each node in T to vt
4. Build shortest path tree spt(v_t, G) with edges on each node
5. Assign interval encoding (pre,post,parent) on each node in spt(v_t,G)

Following that, the edge cost calculation is determined by,

$$e.edgecost = e.weight - (u.cost - v.cost) \quad (1)$$

Here, the shortest path tree is effectively determined by the Bellman-ford algorithm. It is an efficient algorithm that calculates the shortest path to all nodes in the graph from a single source even when the edges between nodes having negative costs.

The major difference between Dijkstra's algorithm and Bellman is that Bellman-Ford's algorithm relaxes all edges, v-1 times (where v is the number of vertices in the graph) while Dijkstra's algorithm greedily selects the not-yet-visited minimum-weight node. Hence, the algorithm affords more efficiency in finding the shortest path between the node pair than other algorithms.

1.7 TOP-K SHORTEST PATH

One of The important operations on the graph is top-k shortest path discovery. The k shortest path between two nodes can be discovered in a non-descending order of their length. Applications such as transportation, social networks use shortest paths. The top-k shortest paths can be used in the sensitive relationship analysis in social networks. The top-k shortest paths can be classified into two categories: the top-k general shortest path and the top-k simple shortest path.The top-k general shortest path allows loops whereas the top-k simple shortest paths are withoutloops. These two problems look similar but they face different complexities. The top-k simple shortest path requires more cost to detect loop and more search space than the former one.

- The single shortest path is insufficient to provide new information in different applications. So to address this problem, we propose a new graph operation, called as top-k path join. Consider two node sets S and T, the top-k path join discover all path in an ascending order of their lengths.
- The straightforward method to top-k path join first discovers top-k paths between each node pair (s, t) and then locates the top-k paths from the union of these results. This strategy requires |S|*|T| times top-k simple shortest path discovery. Improvements to the straightforward method can be made by concentrating on the performance improvement in the top-k shortest path discovery for one node pair and finding the optimized join strategy.
- We design a holistic top-k path join strategy and transform the top-k path join problem into the problem of finding constrained top-k shortest paths. An efficient method to reduce the cost of candidate path searching is also proposed. Two optimization techniques are proposed to avoid paths violating the join constraint. A threshold to prune the path search is introduced. Policies like Lazy and Eager policies are used to determine the threshold value.

1. LITERATURE SURVEY

In this section, some of the related works addressing the shortest path identification scenario is discussed. The problem of finding the k-shortest paths connecting a pair of vertices in a digraph was sensibly covered in [8]. The authors have claimed about the dynamic programming problems including sequence alignment, maximum inscribed polygons, knapsack problem and genealogical relationship discovery. The workalso comprised the description about maintaining path properties, faster path heap construction and improved space and time. Further, the shortest path problem was effectively studied in [5]. The authors have considered two problems in this work: unconstrained and constrained k shortest paths problem. It was stated that ranking loopless paths did not comprise the optimality principle. Moreover, the ranking of shortest paths was attained by determining the set of candidates to the following path. In a different way, optimal XML query processing was discussed in [1]. The pattern matching was performed by structural join and stitching algorithms using holistic twig joins. There, the authors used chain of linked stacks for compactly represent partial results to root-to-leaf query paths. Furthermore, the authors of [4] discussed about the classical Yen's algorithm for ranking the k-shortest loopless paths. They have also claimed that the computational complexity attained there was $O(K\,n(m+n\log n))$ while concerning a worst case analysis. Deviation algorithm, which is the base of Yen's algorithm, was effectively described there.

A new algorithm was proposed for finding k shortest simple paths between nodes in [10]. The algorithm was imposed based on the replacement paths. There was a loophole that the fast replacement paths subroutine is known to fast for some directed graphs. There stated that the processing time could be minimized in further enhancements. The problem of graph pattern matching was effectively studied in [3] using a large data graph. For that purpose, a new two-step R-join algorithm with filter step and fetch step based on a cluster-based join-index with graph code. Using interleaving R-joins and R-semi joins optimization was performed there. A methodology for path tree cover for reachability query was proposed in [6]. Path decomposition was performed in Directed Acyclic Graph (DAG) for effective path determination. Path sub-graph and minimal equivalent edge set was framed. A comparison chart for optimal tree approach and path tree approach was also given in that work to prove its efficiency.

Following that, the authors of [2] described the fast computing distance aware 2-hop covers which can encodethe all-pairs shortest paths of a graph in$O(|V|.|E|^{\frac{1}{2}})$. Graph partitioning was performed there to gain speed. The partitioning was made with two methods: top-down partitioning and bottom-up partitioning. Also, two strategies: fixed and flexible was used for dense graph and sparse graph respectively. Moreover, the authors of [11] described about the distance join method for pattern matching in large graph database. For reducing the search space significantly, transformation of vertices into points in a vector space via graph embedding techniques was performed. Several pruning strategies and a join order selection method was incorporated to process join processing efficiently. Cost optimization techniques could be effectively imposed in further studies.

In order to reduce the processing cost for identifying the top-k simple shortest paths in a graph, a novel description was given in [7]. The authors have used the pre-computation techniques for minimizing the discovery cost at running time. By this algorithm, the total iterations needed for shortest path discovery was effectively reduced and the search space pruning also performed with determined thresholds. A unique approach named, holistic top-k shortest path determination was proposed in [9]. Transformation of graph was processed there for encoding the pre-computed shortest path to the target node. That transformed graph was being used for the estimation of candidate path. The time complexity and cost of this process can further be reduced. Further, a relational approach based shortest path discovery mechanism was proposed in [12]. The bi-directional set Dijkstra's algorithm was used in the process

3. SYSTEM STUDY

3.1 EXISTING SYSTEM

There are two types of existing algorithms .Yen's algorithm for top- k simple shortest path problem and another one is Eppstein's algorithm for the top- k general shortest path problemThe top-k simple shortest path requires more cost for loop detection.And more search space than the former one.

Yen's algorithm locates the shortestpath p=s- b – e- g- t from s to t using Dijkstra's algorithm as the first shortest path, and then initializes apseudo result tree. The pseudo result tree not only is acompact structure to store discovered paths, but also playsan important role in generating other paths.

3.1.1 DISADVANTAGE

The shortest path between the given nodes pair is obviously loop less. The single shortest path is insufficient. There is a worst case time complexity for top- k shortest paths of Yen's algorithm.

3.2PROPOSED SYSTEM

- The holistic top-k path join used two virtual nodes such as source node and target node, which is pre computed first and transform the original graph (G) into another graph (G_{side}).
- And the top- k shortest paths are discovered on (G_{side})..
- The two optimization strategies, including considering the join constraint early and pruning search space with an adaptively determined threshold.

3.2.1Push Join Constraint

Since the basic holistic join method may lead to a large number of candidate paths, which violate the join constraint, push join constraint method is incorporated here. The earlier consideration of join constraints is more specific to address the problem stated above. The candidate path searching process is effectively accomplished with the join constraint, where the overall time complexity needed by this process is given as, $O(Kn(m + \text{n} \log n))$.

3.2.2 Prune Search Space

The candidate path search has been made with a valid deviation node. However, there are no more that k paths in the final results. Thus, it is viable to stop searching when the currently dealt path has no chance to be the final top-k.

Threshold value is considered here to prune the search space. The path having maximum length among the obtained shortest paths is identified and the length value is fixed here as threshold. Let $p_k^{'}$ be the current kth shortest path. The computation of threshold is given as,

$$\textbf{Threshold = Len} (P_{k'}) - \boldsymbol{\delta_G(v_s, v_t)} \qquad \textbf{(2)}$$

Where $\delta\ (v_s, v_t)$ is the shortest distance from v_s to v_t on the original graph G.

The path searching process has been terminated on G_{tran}, when the sum of edge cost of edges from virtual source node exceeds the threshold value. The pruning will also be done at the determination of the kth path $p_k^{'}$ above. It is obvious that the optimal threshold can be achieved when $p_k^{'} = p_k$, where p_k is the appropriate kth shortest path. Moreover, there are two cases discussed for getting an idea of p_k at the beginning of the path searching process. 1. Eager policy and 2. Lazy policy.

In Eager policy, generating as many as possible paths from the deviation node until getting k paths is performed, instead of generating a single path. Then, maximal length is considered for threshold estimation and works in the candidate path generation process. In lazy policy, only one candidate path is generated for each deviation node. The advantage here is that no extra searching is performed. But, the major disadvantage is paths are generated without any valid threshold.

3.3 ADVANTAGE

- Pruning optimization method can save a lot of time.
- The efficiency is improved.
- The holistic path joins works much better than existing.

3.4. SYSTEM DESIGN SPECIFICATION

3.4.1 System Architectural Design

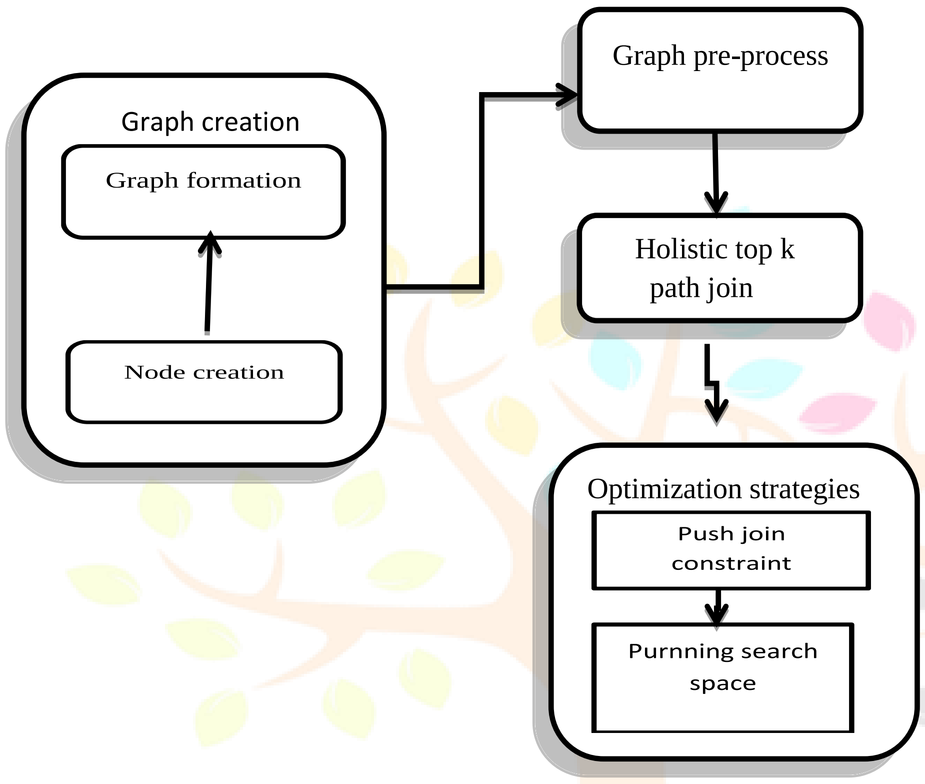


Figure 3.4.1 : System architecture

5. IMPLEMENTATION TECHNIQUES

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

5.2.1 Node Creation

The nodes are initialized first.Next the nodes are created according to hierarchical order.The names of the nodes are displayed inside the node. Each edgeis drawn between the nodes.These creations are done with the help of graphics method.

5.2.2 Graph formation

Next module is to construct the graph, with respect tothe weight of the edge is allocated here.So, each edge weight is displayed in corresponding edges. And each edge connected according to directions of nodes.The full graph formation is completed.

5.2.3Graph Preprocessing

There are two virtual nodes such as source node is added to the root node.Then virtual target node is added to the destination node. And assign these two node weights are zero.The Dijkstra algorithm is applied to graph which is used to find the shortest path in graph. Also the label of the graph is computed.

5.2.4 Holistic Top-K Join Path

The constrained top-k shortest paths on the transformed graph are computed.Holistic top-k join path is similar to yen's method.The deviation node to be selected according to the graph. The first sub path is created before deviation node.The second sub path is created from deviation node. The result of shortest path is merging of two sub path.

5.2.5Push Join Constraint

Push join constraint provides the single source property for the sub path from source to terminated node.This is the post processing of the holistic join method in graph.To avoid multiple source nodes in the candidate path searching to use push join constraintThe multiple source case is considered in push join.

5.2.6Prune Search Space

Pruning search space is one of the threshold methods.This is used to derive a threshold from the maximum length of the graph.It provides two policies such as eager it is used to search for extra paths to get the threshold.Another one is lazythere is no extra searching is provided.

5.2.7Bellman Ford Algorithm

Here, using bellman ford algorithm, the shortest path is calculated.The only difference is that Bellman-Ford's algorithm relaxes all edges, v-1 times (where v is the number of vertices in the graph) while Dijkstra's algorithm greedily selects the not-yet-visited minimum-weight node.This is a graph analysis algorithm for finding shortest paths in a weighted graph (with positive or negative edge weights.A single execution of the algorithm will find the lengths (summed weights) of the shortest paths between all pairs of vertices.

5.3 IMPLEMENTATION OF MODULES

Nodes are created and a weighted, directed graph has been formed with those nodes. Fig.5.1 shows the sample graph that we have generated for this process.

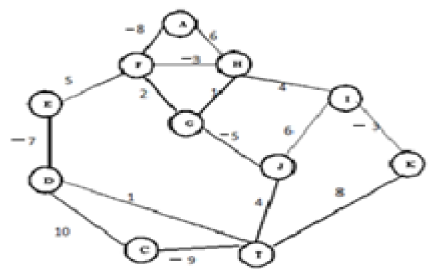
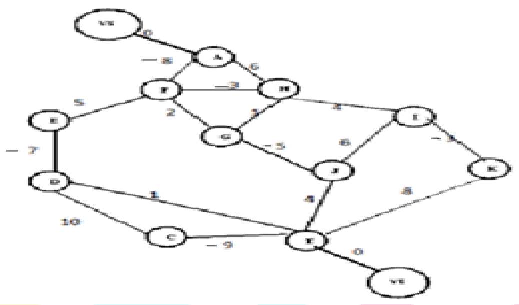


Figure 5.3.1 : Original Graph

In this process, a virtual source node v_s and a virtual target node $v_t$are initialized. Then, the shortest path tree spt(v_t,G) has been built using Bellman-ford algorithm and also the shortest distance from node u to v_t with u.cost is recorded, which is followed by the assessment of e.edgecost on each edge for each node in the graph based on the steps presented below. The output obtained is the transformed graph G_{tran}isgiven below in Figure 5.3.1



`

Figure 5.3.2 Transformed Graph

ALGORITHM 1: Graph initialization(G,S,T)

Input: Graph G=(V,E) ,source node S,target node T.

Output: transformed graph G_{side}

PROCEDURE

```
1.  Form Graph G with source nodes S and target nodes T
2.  Initialize a virtual node $v_s$ and zero-weighted edges from $v_s$to each node in S
3.  Initialize a virtual node $v_t$ and zero-weighted edges from each node in T to vt
4.  Build shortest path tree spt($v_t$, G) with edges on each node
5.  Assign interval encoding (pre,post,parent) on each node in spt($v_t$,G)
6.  for each edgee= (u,v) in G do
        $e.sidecost = e.weight - (u.cost - v.cost)$
7.  return G
```

Here, the shortest path tree is effectively determined by the Bellman-ford algorithm. It is an efficient algorithm that calculates the shortest path to all nodes in the graph from a single source even when the edges between nodes having negative costs.

The major difference between Dijkstra’s algorithm and Bellman is that Bellman-Ford's algorithm relaxes all edges, v-1 times (where v is the number of vertices in the graph) while Dijkstra’s algorithm greedily selects the not-yet-visited minimum-weight node. Hence, the algorithm affords more efficiency in finding the shortest path between the node pair than other algorithms. It aids in determining shortest path of node pairs in a graph efficiently, even there appears some negative weights on edges.

ALGORITHM 2: Bellman-Ford-BGL(G,w,s)
Input:Graph G,Source node S, target node T,source vertexsrc
Output:Shortest distance to all vertices from source node S
PROCEDURE:

```
1.    Initializes distance from source to all vertices as infinite
2.    Set a distance to source itself  as 0.
3.    Create an array d[ ],size[v],with all values infinite except dist[src]
4.     Calculate the distance for |V|-1 times where |V| is the number of vertices in given graph.
5.    Repeat the step 4 for each edge u-v
6.    Check the condition for if there is a negative weight cycle in graph
a.    if dist[v] >dist[u] + weight of edge uv, then update dist[v]
b.    dist[v] = dist[u] + weight of edge uv
   7. If we iterate through all edges one more time and get a shorter path for any vertex.
```

6. PERFORMANCE ANALYSIS

Performance Analysis is a Formal determination of an actions and their outcomes within a particular position or setting.It uncovers several perspectives on a problem. Project performance reporting provides a graphical and tabular overview of performance. It visualizes the improvement of project and is used to compare with existing reports to improve the performance. It is obvious from the Fig.7.1 graph that the travelling cost of the proposed work is much reduced than the existing work because of efficient shortest path determination.

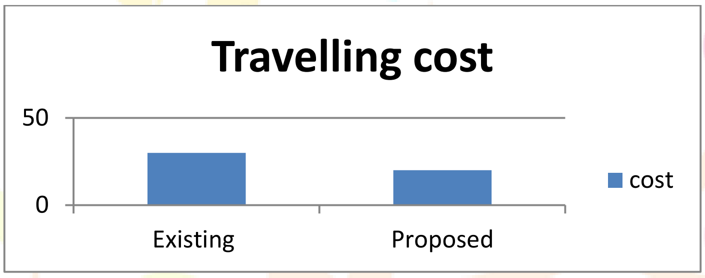


Figure Graph Depicts Reduction Of Travelling Cost

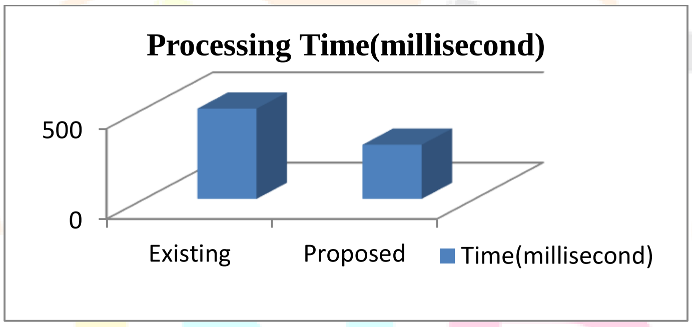


Figure Processing Time for Routing

Figure depicts the path length comparison between the existing and the proposed. It is clear from the graph that the path length between the considered virtual node is considerably reduced than the existing using efficient algorithms such as bellman-ford, top-k path join and path optimization strategies.

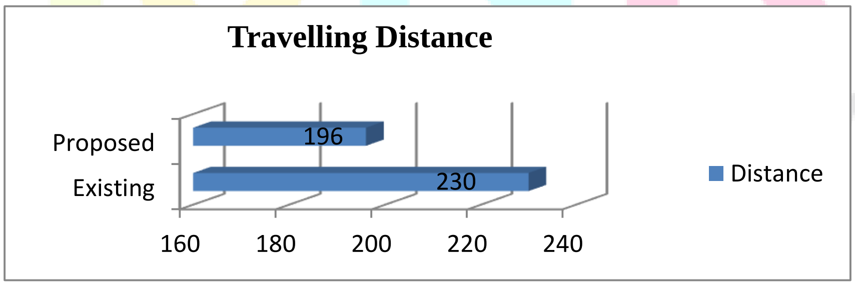


Figure Path Length Between Virtual Nodes

7. CONCLUSION

Our novel approach for identifying top-k shortest paths between node pairs in a graph has shown promising results. By combining efficient preprocessing techniques, the bellman-ford algorithm, and innovative optimization strategies such as the top-k path join method, early join constraint, and lazy pruning, we have achieved superior efficiency and reduced time complexity compared to existing methods. Moving forward, further exploration of different shortest path algorithms holds potential for enhancing efficiency even further.

8. REFERENCES

[1] Nicolas Bruno, Nick Koudas and DiveshSrivastava, “Holistic Twig Joins: Optimal XML Pattern Matching,” In Proceedings of the ACM SIGMOD international conference on Management of data, 2002, pp. 310-321.

[2] Jiefeng Cheng and Jeffrey Xu Yu, “On-line Exact Shortest Distance Query Processing,” In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, 2009, pp. 481-492.

[3] Jiefeng Cheng, Jeffrey Xu Yu, Bolin Ding, Philip S. Yu and Haixun Wang, “Fast Graph Pattern Matching,” IEEE 24th International Conference on Data Engineering (ICDE), 2008, pp. 913-922.

[4] Ernesto Q.V. Martins and Marta M.B. Pascoal, “A new implementation of Yen’s ranking loopless paths algorithm,” 4OR: A Quarterly Journal of Operations Research, 2003, pp. 121-133.

[5] Ernesto, De Queirós Vieira Martins, Marta Margarida BrazPascoal, and Jose Luis Esteves Dos Santos, “Deviation algorithms for ranking shortest paths," International Journal of Foundations of Computer Science 10.03, 1999, pp. 247-26.

[6] Ruoming Jin, Yang Xiang, NingRuan and Haixun Wang, “Efficiently Answering Reachability Queries on Very Large Directed Graphs,” In Proceedings of the ACM SIGMOD international conference on Management of data, 2008, pp. 595-608.

[7] Jun Gao, HuidaQiu, Xiao Jiang, Tengjiao Wang and Dongqing Yang, “Fast Top-k Simple Shortest Paths Discovery in Graphs,” In Proceedings of the 19th ACM international conference on Information and knowledge management, 2010, pp. 509-518.