DySky: Dynamic Skyline Queries on Uncertain Graphs

Suman Banerjee∗
Indian Institute of Technology
Gandhinagar-382355, India
suman.b@iitgn.ac.in

Bithika Pal
Indian Institute of Technology
Kharagpur-721302, India
bithikapal@iitkgp.ac.in

ABSTRACT
Given a graph, and a set of query vertices (subset of the vertices), the dynamic skyline query problem returns a subset of data vertices (other than query vertices) which are not dominated by other data vertices based on certain distance measure. In this paper, we study the dynamic skyline query problem on uncertain graphs (DySky). The input to this problem is an uncertain graph, a subset of its nodes as query vertices, and the goal here is to return all the data vertices which are not dominated by others. We employ two distance measures in uncertain graphs, namely, Majority Distance, and Expected Distance. Our approach is broadly divided into three steps: Pruning, Distance Computation, and Skyline Vertex Set Generation. We implement the proposed methodology with three publicly available datasets and observe that it can find out skyline vertex set without taking much time even for million sized graphs if expected distance is concerned. Particularly, the pruning strategy reduces the computational time significantly.

CSC CONCEPTS
• Information systems → Data management systems; • Mathematics of computing → Graph theory; • Theory of computation → Probabilistic computation.

KEYWORDS
Uncertain Graph, Reliability, Skyline Query, Edge Probability.

1 INTRODUCTION
‘Skyline’ has emerged as an effective multi-criteria decision making operator and hence an extensively researched topic in data management community for almost two decades [6]. Borzsony et al. [3], fist introduced this operator. Given a set of data points \( D \), the skyline operator in it returns the subset of them that are not dominated by other data points present in the dataset. For any two points \( d_1 \) and \( d_2 \), we say that \( d_1 \) dominates \( d_2 \), if with respect to each dimension \( d_1 \) is not worse than \( d_2 \), however, strictly better in at least one dimension. Without loss of generality, in this study, we assume that lower value means better in all dimensions. This problem has been studied in the context of graph data as well [46, 50].

In real-world scenarios, the relationship among agents are uncertain in nature and this uncertainty is caused due to several reasons noisy measurements, unknown values, explicit manipulations, etc. Hence, this kind of situations are modeled as an uncertain graph, where edges are marked with existence probabilities. In case of social networks, these probabilities signify the influence probability between two users, in case of computer networks these signify the successful packet transfer probability between two systems etc. Now, we report some recent literature on skyline query processing and analysis of uncertain graphs.

1.1 Related Work
After introduced by Borzsony et al. [3], skyline queries have been studied on different kinds of data, for different purposes, with different system architectures, such as road networks [9, 32, 49], bicriteria networks [16, 34], uncertain data [33, 48], incomplete data [26, 31], streaming data [7, 28], spatial data [40], encrypted data [30], knowledge graphs [21], wireless sensors networks [41]; for route recommendations [15, 42], finding perspective customers [43]; resisting outliers [14], favorite product queries [47]; with map reduce architecture [35, 36], multi-core architectures [8], cloud computing framework [13] and so on. Keeping the topic of this paper in our mind, here we elaborate the skyline query processing on probabilistic and uncertain data. He et al. [11] studied the skyline query on uncertain time series data and developed a two step methodology for to answer this probabilistically. Park et al. [35] studied the skyline query processing on uncertain data and proposed parallel algorithms for computing the same using map reduce framework. Zhou et al. [48] studied the skyline query processing over uncertain data in distributed environments. Le et al. [25] studied the skyline queries on uncertain data to return the user specific relevant results without enumerating all possible worlds. Recently, there are several studies in this directions [28, 29, 44]. However, to the best of our knowledge skyline query has not been studied yet in the context of uncertain graphs.

Due to different practical applications, in recent times analysis of uncertain graphs have emerged as an important research topic [22, 24]. Several problems have been studied such as clustering [4, 10], embedding [12], subgraph search [4, 17], structural pattern findings [2] and so an. Ke et al. [19] studied the ‘budgeted reliability maximization problem’, where the goal is to add small number of edges to increase the reliability between a given pair of nodes. ke et al. [20] recently studied the \( s \rightarrow t \) reliability problem which asks
with how much probability a target node $t$ is reachable from a source node $s$ in a given uncertain graph. Chen et al. [5] studied the frequent pattern finding in uncertain graphs and for this problem enumeration-evaluation algorithm for this problem. Look into [18] for survey.

1.2 Our Contribution
In this paper, we propose the noble problem "dynamic skyline query on uncertain graphs". Given an uncertain graph with a subset of vertices as query vertices, the goal of this problem is to obtain the subset of the data vertices that are not dominated by the other data vertices with respect to some distance measure from the query vertices. Particularly, we make the following contributions in this paper:

- We introduce the noble problem "Dynamic Skyline Queries on Uncertain Graph Problem" (DySky).
- We propose a solution methodology for this problem, which broadly divided into three steps, namely, pruning, distance computation and skyline vertex set generation.
- Proposed methodology has been implemented with three benchmark datasets and results show that the pruning strategy leads to less number of candidate nodes.

1.3 Organization
Rest of the paper is organized as follows: Section 2 describes required preliminary definitions and then define the problem formally. The proposed methodology has been described in Section 3. In Section 4 the experimental evaluations of the proposed methodology has been described. Section 5 draws conclusions and gives future directions.

2 PRELIMINARIES AND PROBLEM DEFINITION
In this section, we present required preliminary concepts and then define the dynamic skyline queries on uncertain graph problem formally. Initially, we start with a few basic definitions.

Definition 1 (Uncertain Graph). We denote an uncertain graph by $G(V, E, W, P)$, where $V(G) = \{v_1, v_2, \ldots, v_n\}$ is the set of $n$ vertices, $E(G) \subseteq V(G) \times V(G)$ is the set of $m$ edges, $W$ is the distance function that assigns each edge to a positive real number, i.e., $W : E(G) \rightarrow \mathbb{R}^+$, and $P$ is the existence function that assigns each edge to a probability value, i.e., $P : E(G) \rightarrow (0, 1]$.

In our study, we consider only simple, finite, undirected, and weighted graphs. The number of nodes and edges of the graph $G$ is denoted by $n$ and $m$, respectively. For an edge $e \in E(G)$ its weight and existence probability is denoted by $W(e)$ and $P(e)$, respectively. In the literature, an uncertain graph is conceptualized and analyzed by the possible world model, which we define next.

Definition 2 (Possible World Semantics). An uncertain graph $G(V, E, W, P)$ can be conceptualized as the probability distribution over a set of deterministic graphs, which is called as the possible world of the uncertain graph, and denoted as $L(G)$. Each $G(V, E, W) \in L(G)$ is obtained from $G$ by keeping all its vertices, keeping its edges with existing probability, and if an edge of $G$ is also there in $G$, then $W(e) = W(e)$. Now, the probability that the deterministic graph $G$ will be generated can be computed by the Equation 1.

$$\mathcal{P}_{G \subseteq G} = \prod_{e \in E(G)} P(e) \prod_{e \in E(G), e \notin E(G)} (1 - P(e)) \quad (1)$$

In any deterministic graph $G$, its two vertices $v_i$ and $v_j$ are said to be reachable if there exist a path from between $v_i$ and $v_j$. However, in case of uncertain graphs, the reachability between any two given vertices can be defined in probabilistic way, which we call reliability.

Definition 3 (Reliability). Given an undirected, uncertain graph $G$, the reliability between its any two vertices $v_i$ and $v_j$ is defined as the probability that the vertices $v_i$ and $v_j$ can be reachable from each other. We denote the reliability between the vertices $v_i$ and $v_j$ by $R_{(v_i, v_j)}^G$ and defined by the following equation:

$$R_{(v_i, v_j)}^G = \sum_{G \in L(G)} I_G^{(v_i, v_j)} \mathcal{P}_{G \subseteq G} \quad (2)$$

Here, $I_G^{(v_i, v_j)}$ is the boolean variable whose value is $1$ if $v_i$ and $v_j$ are connected in $G$ and $0$ otherwise.

In case of a deterministic weighted graph, distance between any two vertices is defined as the sum of individual edge weights constituting shortest path. However, in case of uncertain graphs distance between any two vertices can be defined in many ways. Here, we quote two of them that we use in our study.

Definition 4 (Majority Distance). [37] Given an uncertain graph $G$ and its two vertices $v_i, v_j \in V(G)$, its majority distance is denoted by $\text{dist}_{\text{md}}(v_i, v_j)$ and defined as the most probable shortest path distance. Mathematically, it can be given by the following equation.

$$\text{dist}_{\text{md}}(v_i, v_j) = \arg\max_d p_{v_i, v_j}(d) \quad (3)$$

where $p_{v_i, v_j}$ is the shortest path distribution between the vertices $v_i$ and $v_j$ that gives probability value for every distance $d$.

$$p_{v_i, v_j}(d) = \sum_{G \in d_{\mathcal{L}(G) \rightarrow d}} \mathcal{P}_{G \subseteq G} \quad (4)$$

Definition 5 (Expected Distance). Given an uncertain graph $G$ and its two vertices $v_i, v_j \in V(G)$, let $l_{(v_i, v_j)}$ denotes the set of paths upto length $l$. For each path $p_k \in l_{(v_i, v_j)}$, the path probability is defined as

$$p_{(v_i, v_j)} = \frac{\prod_{e \in p_k} P(e)}{\sum_{p_j \in l_{(v_i, v_j)}} \prod_{e \in p_j} P(e)} \quad (5)$$

Expected distance between $v_i$ and $v_j$ is defined as the

$$\text{dist}_E(v_i, v_j) = \sum_{p_k \in l_{(v_i, v_j)}} \text{dist}(p_k) \mathcal{P}(p_k) \quad (6)$$

For any $p \in \mathbb{Z}^+$, $|p|$ denotes the set $\{1, 2, \ldots, p\}$. Given a set of $2$ or more dimensional data points $D$, the problem of skyline query computation asks to find out the data points that are not dominated by any other data points in $D$, which is formally defined in Definition 6.
The vertices 2 and 4 are the Query Vertices (denoted as Q2 and Q4) and remaining are data vertices (i.e., D1, D3, D5, and D6). (b) Different distance tables between the query and data vertices. Skyline vertices in each cases are marked in Orange.

Figure 1: (a) An uncertain graph with 6 vertices and 8 edges. The vertices 2 and 4 are the Query Vertices (denoted as Q2 and Q4) and remaining are data vertices (i.e., D1, D3, D5, and D6). (b) Majority Distance (MD), Expected Distance, and Shortest Path Distance (in deterministic version).

3 PROPOSED METHODOLOGY

Now, we describe the proposed methodology for solving the DySky Problem. Initially, we start by describing an overview of it.

3.1 Overview

The proposed methodology is broadly divided into three steps:

- **Step 1 (Pruning):** In this step, a subset of the data vertices are returned as the candidate skyline vertices. This step comprises of two subsets. First, pruning is done by performing **Breadth First Search** (B.F.S., henceforth) from the query vertices and subsequently, pruning is done based on distance computation.

- **Step 2 (Distance Computation):** In this step, distance computation is done between the candidate skyline vertices and query vertices. As mentioned previously, in our study we have used majority distance and expected distance.

- **Step 3 (Skyline Vertex Set Generation):** Based on the previously computed distance, any existing skyline finding algorithm can be used to find out the actual skyline vertices. In our study, we have used the **Block Nested Loop (BNL)** Algorithm proposed by Borzsonyi et al. [3].

Next, we proceed towards representing the proposed methodology in an algorithmic form and its detailed analysis.

3.2 The Algorithm

Algorithm 1, 2, and 3 together constitute the proposed methodology for the DySky Problem. We describe the entire procedure in two subsections. First we start with describing the pruning step.

3.2.1 The Pruning Step

Algorithm 1 describes the B.F.S. and distance-based pruning strategies, which takes the uncertain graph, the set of query vertices, and distance threshold as inputs and outputs the candidate skyline vertices. In B.F.S. pruning, from each of the query vertices, B.F.S. trees are constructed to check the connectivity. First, we create the dictionary $D$. If a query vertex and data vertex is connected and the data vertex has the entry in the dictionary $D$, the query vertex is included as a value to this ‘key’. Otherwise, a key corresponding to the data vertex is created and the query vertex is included as a value corresponding to this key.

Next, we proceed towards representing the proposed methodology in an algorithmic form and its detailed analysis.

3.2.1 The Pruning Step

Algorithm 1 describes the B.F.S. and distance-based pruning strategies, which takes the uncertain graph, the set of query vertices, and distance threshold as inputs and outputs the candidate skyline vertices. In B.F.S. pruning, from each of the query vertices, B.F.S. trees are constructed to check the connectivity. First, we create the dictionary $D$. If a query vertex and data vertex is connected and the data vertex has the entry in the dictionary $D$, the query vertex is included as a value to this ‘key’. Otherwise, a key corresponding to the data vertex is created and the query vertex is added as a value to this ‘key’. Now, the data vertices that are reachable from all the query vertices are kept as the candidate skyline vertices. In B.F.S. pruning, from each of the query vertices, B.F.S. trees are constructed to check the connectivity. First, we create the dictionary $D$. If a query vertex and data vertex is connected and the data vertex has the entry in the dictionary $D$, the query vertex is included as a value to this ‘key’. Otherwise, a key corresponding to the data vertex is created and the query vertex is added as a value to this ‘key’. Now, the data vertices that are reachable from all the query vertices are kept as the candidate skyline vertices. Here, the B.F.S. pruning ends.

In reality, even if two vertices are connected by a path of large distance (i.e., more than certain threshold), reachability becomes costlier. Hence, to eliminate such vertices, we perform the distance-based pruning. For this purpose, distance between candidate skyline vertex and query vertex is computed. For a candidate skyline vertex, if there exist at least one query vertex for which the computed distance value is more than the user defined threshold, the candidate skyline vertex set is updated by removing the candidate skyline vertex.

Any pruning strategy to work correctly should guarantee that it does not remove any skyline vertex. Hence, we show that the Algorithm 1 is a correct pruning strategy in Lemma 1.

**Lemma 1.** The proposed pruning strategy (Algorithm 1) is correct.

**Proof.** Follows from the description. □
Now, we do an analysis for time and space requirement of Algorithm 1. Let $q$ be the number of query vertices, i.e., $|Q| = q$. For creating the B.F.S. trees rooted at the query vertices requires $O(q(m + n))$ time. The maximum number of values associated with a ‘key’ in the dictionary $D$ is of $O(q)$. Execution time from Line No. 3 to 14 and 16 to 20 requires $O(q(n - q)^2)$ and $O(n^2 d)$ time. Hence, total running time is $O(q(n - q))$. Computing shortest path between two vertices in a weighted graph with positive edge weights requires $O(m + n \log n)$ time. Hence, time requirement for distance-based pruning requires $O(q(n - q)(m + n \log n))$ time. Total time requirement for Algorithm 1 is of $O(q(m + n) + nq(n - q) + q(n - q)(m + n \log n)) = O(q(n - q)(m + n \log n))$. Extra space requirement of Algorithm 1 is to store the dictionary $D$, which is of $O(q(n - q))$, to store the candidate skyline vertices, which is of $O(n - q)$, and to perform the B.F.S., which is of $O(n)$. Hence, total space requirement of Algorithm 1 is of $O(q(n - q))$. Lemma 2 describes the formal statement.

**Lemma 2.** Time and space requirement of Algorithm 1 is of $O(q(n - q)(m + n \log n))$ and $O(q(n - q))$, respectively.

3.2.2 Distance Computation and Skyline Vertex Set Generation.

Now, we describe Step 2 and 3 of our proposed methodology. It is important to observe that depending upon which distance measure is used (i.e., majority distance or expected distance) Step 2 will be different. Algorithm 2 and 3 describes the last two steps for the majority distance and expected distance, respectively.

**Algorithm 2:** Step 2 and 3 (Distance Computation and Skyline Vertex Set Generation) for Majority Distance

**Data:** Candidate Skyline Vertices $CS$

1. **Result:** The Skyline Vertex Set $S$.

2. Generate $|R|$ number of samples graphs ;
3. Store the graph probabilities in $P_G[1...|R|]$;
4. Create_Matrix $M \in \mathbb{R}^{CS \times |Q|}$;
5. for All $u \in CS$ do
6.  for All $v \in Q$ do
7.  if $d = \text{shortest distance between } u \text{ and } v \text{ in } r$;
8.  $\text{Temp}[d] = \text{Temp}[d] + P_G[r]$;
9.  end
10. end
11. end
12. end
13. $S = \text{Apply BNL on } M$;
14. return $S$;

For the majority distance case, first we generate $|R|$ number of subgraphs as mentioned in Definition 2, and the corresponding generation probabilities are stored in the array $P_G$. Next, the majority distance is computed between a candidate skyline vertex and a query vertex. Finally, the BNL Algorithm is applied on the distance matrix $M$ to obtain the skyline vertex set.

Now, we analyze Algorithm 2 for time and space requirement. As mentioned in Definition 2, generation of $|R|$ number of subgraphs require $O(m| R |)$. Using Dijkstra’s algorithm computing the shortest path between a pair of vertices requires $O(m + n \log n)$ time. Hence, execution time from Line 5 to 14 requires $O(q(n - q)|R|(m + n \log n))$. Now, BNL algorithm requires $O((n - q)^2)$ time. Extra space consumed by Algorithm 2 is to store the array $P_G$, Temp, and the matrix $M$ which requires $O(|R|)$, $O(|R|)$, and $O(q(n - q))$ space, respectively. The formal statement is mentioned in Lemma 3.

**Lemma 3.** Time and space requirement of Algorithm 2 is of $O(q(n - q)|R|(m + n \log n) + (n - q)^2)$ and $O(q(n - q) + |R|)$, respectively.

Lemma 2 and 3 together imply the statement mentioned in Theorem 1.

**Theorem 1.** If majority distance is concerned, the proposed methodology returns the skyline vertex set in $O(q(n - q)|R|(m + n \log n) + (n - q)^2)$ time and $O(q(n - q) + |R|)$ space.

It is trivial to observe that Algorithm 3 just implements the expected distance, and hence, without explanation we move to analyze the algorithm. Assume that maximum degree of the input uncertain graph is $d_{max}$. Hence, the maximum number paths upto length $l$ between any pair of vertices is of $O(d_{max}^l)$. Hence, total running
time of Algorithm 3 is of $O(q(n - q)d_{\max}^l + (n - q)^2)$. Extra space consumed by the Algorithm 3 is to store the matrix $M$, array $Path$, $Prob$ and $dist$ which requires $O(q(n - q) + ld_{\max}^l)$. Hence, Lemma 4 holds.

**Lemma 4.** The running time and space requirement of Algorithm 3 is of $O(q(n - q)ld_{\max}^l + (n - q)^2)$ and $O(q(n - q) + ld_{\max}^l)$, respectively.

Lemma 2 and 4 together imply the statement mentioned in Theorem 2.

**Theorem 2.** If expected distance is concerned, the proposed methodology returns the skyline vertex set in $O(q(n - q)(m + n \log n + ld_{\max}^l) + (n - q)^2)$ time and $O(q(n - q) + |R|)$ space.

---

**Algorithm 3:** Step 2 and 3 (Distance Computation and Skyline Vertex Set Generation) for Expected Distance

**Data:** Candidate Skyline Vertices $CS$

1. **Result:** The Skyline Vertex Set $S$.
2. Create Matrix $M \in \mathbb{R}^{|CS| \times |Q|}$;
3. for All $u \in CS$ do
   4. for All $v \in Q$ do
      5. $path = $ Compute all paths from $u$ to $v$ up to length $l$;
      6. prob[1...|path|] = 0; $dist[1...|path|] = 0$;
      7. for All $t \in path$ do
         8. for All $e \in E(t)$ do
            9. $prob[t] = prob[t] + P(e)$;
            10. $dist[t] = dist[t] + P(e) \times W(e)$;
         end
      end
   end
3. $M[u][v] = \sum dist / \sum prob$;
4. end
5. $S = $ Apply BNL on $M$;
6. return $S$;

---

### 4 Experimental Evaluations

In this section we describe the experimental validations of our proposed approach. Initially, we start by describing the datasets.

#### 4.1 Datasets

In our study, we have used three different datasets appeared in three different contexts described below.

- **Minnesota Road Network (MRN)** [39]: This is a road network dataset of the Minneapolis city. Here, the junctions are represented by the nodes, and if two junctions are connected by a road then the corresponding two vertices are connected by an edge.
- **P2P Network** [27, 38]: This dataset contains a sequence of snapshots of the Gnutella peer-to-peer file sharing network from August 2002. There are total of 9 snapshots of Gnutella network collected in August 2002. Nodes represent hosts in the Gnutella network topology and edges represent connections between the Gnutella hosts.

#### 4.2 Experimental Setup

In our study the following three different query vertex selection strategies have been adopted:

- **RAND**: By this method, to select $k$ query vertices first one is chosen randomly and remaining $(k - 1)$ query vertices are chosen from the two hop neighbors of the initially selected vertex uniformly at random.
- **HDEG**: By this method, to select $k$ query vertices first the subset of the nodes whose degree is more than a threshold value are marked and a node is chosen uniformly at random as a query vertex. Remaining $(k - 1)$ are chosen from the two hop neighbors of the initially selected vertices uniformly at random.
- **HCLUS**: This method is exactly the same as HDEG, except the case that, for choosing the first query vertex the subset of vertices are chosen based on the clustering coefficient of nodes.

Based on the selection strategy, we choose the query size from the set $\{2, 3, 5, 8, 10, 15, 20\}$. The experiments are repeated for 10 times. All the algorithms have been implemented with Python 3.5 + NetworkX 2.1 environment on a HPC Cluster with 5 nodes each of them having 64 cores and 160 GB of memory and the implementations are available at https://github.com/BITHIKA1992/Skyline_Uncertain_Graph/

#### 4.3 Goals of the Experiments

The experiments that have been conducted here aim to address the following questions:

- **Efficiency of the Pruning Strategies**: As the number of query vertices increases, what is the fraction of data vertices removed before distance computation?
- **Query Size Vs. Skyline Vertices**: Under different query vertex selection strategies how the cardinality of the skyline vertex set changes with respect to the query size?
- **Distance Metric Vs. Skyline Vertices**: For a fixed query selection strategy and query size, how the cardinality of the skyline vertices changes with respect to the distance metric?
Query Selection Strategy Vs. Skyline Vertices: For a fixed query size and distance metric, how the cardinality of the skyline vertices changes with respect to query selection strategy?

Query Size Vs. Computational Time: For a fixed query size and distance metric, how computational time grows with respect to query size?

4.4 Results and Discussion
Here, we address the research questions that we have raised.

4.4.1 Efficiency of the Pruning Strategies. As we apply BFS pruning in each dataset, it returns the vertices from the largest component. This reduces 3000, and 114 number of vertices for URN and P2P network dataset. For distance based pruning, we have taken the threshold value as 400, considering 4-hop path with the maximum edge weight 100. In Figure 2, we show the box plot for the candidate size with respect to each query size and the query selection strategy. It can be observed that the candidate size for RAND selection strategy is less than other two, in all the datasets, which is trivial to convince. For P2P network, the inter quartile range is very high compared to other datasets. This is due to the reason of high average degree in the network. Also, for RAND selection strategy, this range is the highest for small query size. This is due to the existence of various small size component in the network. Both the road networks are very sparse and for the large query sizes like 15, 20, the candidate size becomes very small and the variance also reduces. With this sparsity for small road network MRN, it is impossible to find the connected vertices from all the query vertices within the distance of 400. So, we remove the results for query size of 15 and 20 in MRN dataset.

4.4.2 Query Size Vs. Skyline Vertices. In Figure 3, we show the plot for query size Vs. skyline size, with two distance metrics and three query selection strategies. In this part, we describe the comparison of sizes. From all the 10 executions, here we report the mean values for the skyline size. With the increase in query size, the skyline size increases. However, for URN dataset in Figure 3(c), the skyline size decreases for large value of query size. The reason is due to small size of candidate skyline, which can be verified from the Figure 2(c). Also, for both the road network datasets the maximum skyline size reaches to approximately 15, whereas for the P2P network it reaches to around 1500. This due to its candidate size. For both the cases, at large value of query size the ratio of candidate to skyline size is very small. As the number of query vertices increase, the chance of domination decreases.

4.4.3 Distance Metric Vs. Skyline Vertices. In this part, referring to Figure 3, we describe the behavior of skyline size with respect to different distance metrics. For the road networks in Figure 3(a) and (c), the skyline size is similar in both the datasets. However, for the P2P network in Figure 3(b), the skyline size in the expected distance (≈ max 1500) is much more than the majority distance (≈ max 300). The reason lies on the networks high average degree value and the density. As the number of paths increases between a query vertex to a data vertex, the expected distance value is unable to dominate.
other data vertices. This results into large size of skyline vertex set. This can be verified from Figure 9(b), by looking into HDEG and and HCLUS selection strategies, where it differs from the expected distance results. However, for RAND, the size is similar in both the distances. From the experiments, we also observe that for a particular query vertex set the skyline vertices may not be the same from both the distance metrics.

4.4.4 Query Selection Strategy Vs. Skyline Vertices. In this part, referring to Figure 3, we describe the behavior of skyline size with respect to different query selection strategies. First, we describe the threshold value selected for HDEG and HCLUS for different datasets. As the P2P network dataset consists of high degree nodes, we select the high degree threshold value as 15, and it returns 440 nodes. In case of both the road networks, the maximum degree is around 5. Hence, for MRN and URN datasets, this threshold value is considered as 2 and 3, respectively. The clustering coefficient threshold is taken as 0 as the clustering coefficient for all the networks are very less. From Figure 3, the main observation is that for all the selection strategies the skyline size does not vary much for smaller query size. Whereas, for the large value of query size, HCLUS gives maximum skyline vertices.

4.4.5 Computational Time. Table 2 contains the stepwise computational time requirement to find skyline vertices for different datasets. From the table, it can be observed that for all the datasets as the query size increases, time requirement for finding out the skyline vertex set also increases. Due to the change in the query size, required time for distance-based pruning, distance and skyline computation (using BNL) increases. Also, for all the datasets, the main time requirement is due to the sample graph generation. As in case of expected distance sample generation is not required, hence, in this distance setting time requirement is much less compared to the majority distance. In particular, for query size 2, the ratio between the computational time requirement for majority distance to expected distance for MRN, P2P, and URN are 47, 28, and 2556, respectively.

Now, we proceed for the dataset specific observations. For the P2P Network dataset, when the query size increases beyond 10, there is a sharp increase in the skyline computation time. This is due to the following two reasons. From the Figure 2(b) and 3(b), it can be observed that candidate and skyline size are more compared to the previous query sizes.

5 CONCLUSION AND FUTURE DIRECTIONS

In this paper, we introduce the problem of dynamic skyline queries on uncertain graphs for two different distance measures, namely, majority distance and expected distance. For this problem, we have proposed a methodology having three main steps: pruning, distance computation, and skyline vertex set generation. The proposed methodology has been analyzed to understand its time and space requirement. The experimental results demonstrate that it can find out the skyline vertex set with reasonable computation time, particularly for the expected distance.

Now, this study can be extended in several directions. It will be an interesting future study to come up with efficient methodology, which can reduce the computational time even for majority distance. One possible way could be parallelizing the sample graph generation. It will be an important future work to provide a sample bound for the majority distance case. The minimum number of samples from the possible world, one should choose to answer the skyline with more than certain threshold probability.

ACKNOWLEDGMENTS

Authors want to thank Ministry of Human Resource and Development (MHRD), Government of India, for sponsoring the project, E-business Center of Excellence under the scheme of Center for Training and Research in Frontier Areas of Science and Technology (FAST), Grant No. F.No.5-5/2014-TS.VII.

REFERENCES

[1] Mikhail J Atallah and Yanian Qi. 2009. Computing all skyline probabilities for uncertain data. In Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART
