DySky: Dynamic Skyline Queries on Uncertain Graphs

Suman Banerjee[1](✉), Bithika Pal[2], and Mamata Jenamani[2]

[1] Indian Institute of Technology, Gandhinagar, India
suman.b@iitgn.ac.in
[2] Indian Institute of Technology, Kharagpur, India
bithikapal@iitkgp.ac.in, mj@iem.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 the context of 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.

Keywords: Uncertain graph · Reliability · Skyline query

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 [4]. Borzsony et al. [2] 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 [20]. In real-world scenarios, the relationship among agents are uncertain in nature and this uncertainty is caused due to several reasons, like noisy

The work of the first author is supported by the institute post-doctoral fellowship grant by IIT-Gandhinagar. (Project Number: MIS/IITGN/PD-SCH/201415/006).

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.

After introduced by Borzsony et al. [2], skyline queries have been studied on different kinds of data, for different purposes, with different system architectures, such as on road networks [14], on uncertain data [21], on spatial data, finding perspective customers [18], resisting outliers, in distributed environment [21], map-reduce framework [15], and so on. Keeping the topic of this paper in our mind, here we briefly elaborate the skyline query processing on probabilistic and uncertain data. He et al. [5] studied the skyline query on uncertain time series data and developed a two step methodology to answer this probabilistically. Park et al. [15] studied the skyline query processing on uncertain data and proposed parallel algorithms for computing the same using map reduce framework. Zhou et al. [21] studied the skyline query processing over uncertain data in distributed environments. Le et al. [12] studied the skyline queries on uncertain data to return the user specific relevant results without enumerating all possible worlds. Though there are several studies in this direction, skyline query has not been studied yet in the context of uncertain graphs, to the best of our knowledge.

Due to different practical applications, in recent times analysis of uncertain graphs have emerged as an important research topic [11]. Several problems have been studied such as embedding [6], subgraph search [7], structural pattern findings [1] and so an. ke et al. [9] recently studied the $s-t$ 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. [3] studied the frequent pattern finding in uncertain graphs and enumeration-evaluation algorithm for this problem. Look into [8] for survey. Motivated by the scenario that most of the real world networks are uncertain in nature, in this paper, we introduce the problem of skyline queries on uncertain graphs. Particularly, it has the following contributions:

- We propose the noble problem "***Dy****namic* ***Sky****line Queries on Uncertain Graph Problem*" (**DySky**). 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.
- We propose a solution approach for this problem, which is divided into three steps: pruning, distance computation, and skyline vertex set generation.
- The proposed methodology has been implemented with three datasets, for different query size, selection strategies, and distance measures.

Rest of the paper is organized as follows: Sect. 2 describes required preliminaries and defines the problem formally. Section 3, contains the proposed methodology, followed by the experimental evaluations in Sect. 4. Finally, Sect. 5 draws the conclusions.

2 Preliminaries and Problem Definition

In this section, we present required preliminary concepts and then define the problem formally. We denote an *uncertain graph* by $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{P})$, where $\mathcal{V}(\mathcal{G}) = \{v_1, v_2, \ldots, v_n\}$ is the set of n vertices, $\mathcal{E}(\mathcal{G}) \subseteq \mathcal{V}(\mathcal{G}) \times \mathcal{V}(\mathcal{G})$ is the set of m edges, $\mathcal{W}$ is the distance function that assigns each edge to a positive real number, i.e., $\mathcal{W} : \mathcal{E}(\mathcal{G}) \longrightarrow \mathbb{R}^+$, and $\mathcal{P}$ is the existence function that assigns each edge to a probability value, i.e., $\mathcal{P} : \mathcal{E}(\mathcal{G}) \longrightarrow (0, 1]$. In our study, we consider only simple, finite, undirected, and weighted graphs. The number of nodes and edges of the graph $\mathcal{G}$ is denoted by n and m, respectively. For an edge $e \in \mathcal{E}(\mathcal{G})$ its weight and existence probability is denoted by $\mathcal{W}(e)$ and $\mathcal{P}(e)$, respectively. In the literature, an uncertain graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{P})$ 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 $\mathcal{L}(\mathcal{G})$. Each $G(V, E, W) \in \mathcal{L}(\mathcal{G})$ is obtained from $\mathcal{G}$ by keeping all its vertices, keeping its edges with existing probability, and if an edge of $\mathcal{G}$ is also there in G, then $\mathcal{W}(e) = W(e)$. Now, the probability that the deterministic garph G will be generated can be computed by the Eq. 1.

$$\mathcal{P}_{G \sqsubseteq \mathcal{G}} = \prod_{e \in E(G)} \mathcal{P}(e) \prod_{e \in E(\mathcal{G}) \setminus E(G)} (1 - \mathcal{P}(e)) \tag{1}$$

In any deterministic graph G, two vertices v_i and v_j are said to be reachable if there exist a path 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*. The term reliability between any two vertices v_i and v_j in the uncertain graph $\mathcal{G}$ is defined as the probability that the vertices v_i and v_j are reachable from each other. We define the reliability ($\mathcal{R}^{\mathcal{G}}_{(v_i v_j)}$) between two vertices v_i and v_j as,

$$\mathcal{R}^{\mathcal{G}}_{(v_i v_j)} = \sum_{G \in \mathcal{L}(\mathcal{G})} I^{G}_{(v_i v_j)} \mathcal{P}_{G \sqsubseteq \mathcal{G}}. \tag{2}$$

where, $I^{G}_{(v_i v_j)}$ 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 1 (Majority Distance). *[16] Given an uncertain graph $\mathcal{G}$ and its two vertices $v_i, v_j \in V(\mathcal{G})$, its majority distance is denoted by $dist_{md}(v_i, v_j)$ and defined as the most probable shortest path distance. Mathematically, it can be represented by Eq. 3, where $p_{v_i v_j}$ (Eq. 4) is the shortest path distribution between the vertices v_i and v_j that gives probability value for every distance d.*

$$dist_{md}(v_i, v_j) = \underset{d}{argmax}\ p_{v_i v_j}(d) \tag{3}$$

$$p_{v_iv_j}(d) = \sum_{G|d_G(v_i,v_j)=d} \mathcal{P}_{G \sqsubseteq \mathcal{G}} \tag{4}$$

Definition 2 (Expected Distance). *Given an uncertain graph $\mathcal{G}$ and its two vertices $v_i, v_j \in V(\mathcal{G})$, let $P^l_{(v_iv_j)}$ denotes the set of paths upto length l. For each path $p_k \in P^l_{(v_iv_j)}$, the path probability is defined by $\mathbb{P}(p_k)$ in Eq. 5, and Eq. 6 defines the expected distance between v_i and v_j.*

$$\mathbb{P}(p_k) = \frac{\prod_{e \in p_k} \mathcal{P}(e)}{\sum_{p_j \in P^l_{(v_iv_j)}} \prod_{e \in p_j} \mathcal{P}(e)} \tag{5}$$

$$dist_E(v_i, v_j) = \sum_{p_k \in P^l_{(v_iv_j)}} dist(p_k).\mathbb{P}(p_k) \tag{6}$$

For any $p \in \mathbb{Z}^+$, $[p]$ denotes the set $\{1, 2, \ldots, p\}$. Given a set of 2 or more dimensional data points $\mathcal{D}$, the problem of *skyline query computation* asks to find out the data points that are not dominated by any other data points in $\mathcal{D}$, which is formally defined in Definition 3.

Definition 3 (Skyline Query). *Given a set of p dimensional data points $\mathcal{D} = \{d_1, d_2, \ldots, d_{|\mathcal{D}|}\}$, we say that d_i dominates d_j, if for all $k \in [p]$, $d_i(k) \leq d_j(k)$ and there exist atleast one $k \in [p]$ such that $d_i(k) < d_j(k)$. Skyline of the dataset $\mathcal{D}$ is the subset of the data points that are not dominated by any of the data points in $\mathcal{D}$.*

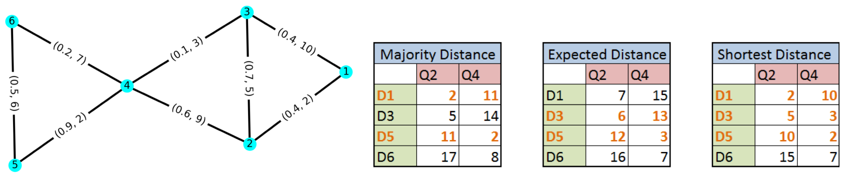
Since past one decade or so, skyline queries have been studied extensively [10] in graphs as well, which we define next.

Definition 4 (Skyline Query in Graphs). *Given a graph $G(V, E)$, and a subset of vertices $\mathcal{Q}$ (called query vertices), for any two data vertices (vertices that are not query vertices, i.e., $V(G) \setminus \mathcal{Q}$) v_i and v_j, we say v_i dominates v_j, if $\forall w \in \mathcal{Q}$, $dist(w, v_i) \leq dist(w, v_j)$ and $\exists x \in \mathcal{S}$, such that $dist(x, v_i) < dist(x, v_j)$. The skyline query asks to return data vertices that are not dominated by other data vertices.*

Though, the skyline query problem has been studied in the context of probabilistic data [19], to the best of our knowledge this problem has not been studied in the context of uncertain graphs. In this paper, we introduce the problem of finding the dynamic skyline queries on uncertain graphs (DySky) as follows.

Definition 5 (Dynamic Skyline Queries on Uncertain Graphs). *Given an uncertain graph $\mathcal{G}$, a subset of vertices $\mathcal{Q}$ (called query vertices), and a distance measure (i.e., expected distance, majority distance etc.) the problem of dynamic skyline queries on uncertain graphs asks to find out the subset of the data vertices such that none of them are dominated by the other data vertices.*

Just to highlight, here the term ‘Dynamic’ does not mean the graph is time varying; rather it means that the query vertices are not fixed and can be varied. Figure 1 shows a toy example of an uncertain graph with its majority distance, expected distance, and shortest path distance (for deterministic version) tables, where the skyline vertices are marked in orange color. It is important to observe as the distance measure changes, the skyline vertex set is also getting changed. This motivates us to study the DySky Problem under two different distance measures.



Majority Distance

	Q2	Q4
D1	2	11
D3	5	14
D5	11	2
D6	17	8

Expected Distance

	Q2	Q4
D1	7	15
D3	6	13
D5	12	3
D6	16	7

Shortest Distance

	Q2	Q4
D1	2	10
D3	5	3
D5	10	2
D6	15	7

(a) (b)

Fig. 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) Different distance tables (Majority Distance, Expected Distance, Shortest Path Distance in deterministic version) between the query and data vertices. Skyline vertices in each cases are marked in Orange. (Color figure online)

3 Proposed Methodology

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* (henceforth mentioned as B.F.S.) from the query vertices and subsequently, pruning is done based on path length computation.
- **Step 2 (Distance Computation):** In this step, distance computation is done between the candidate skyline vertices and the 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. [2].

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.

The Pruning Step. Algorithm 1 describes the B.F.S. and distance based pruning strategies. It 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 $\mathcal{D}$. If a query vertex and data vertex is connected and the data vertex has the entry in the dictionary $\mathcal{D}$, the query vertex is included as a value corresponding 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 path length-based pruning. For this purpose, shortest path length between candidate skyline vertex and query vertex is computed. For a candidate skyline vertex, if there exist atleast one query vertex for which the computed length 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 vertices. In Lemma 1, we state the proposed pruning strategy is correct. Due to space limitation, we are not able to give the proof, though it is easy to follow.

Algorithm 1: Step 1 (B.F.S and Distance based pruning)

```
Data: Uncertain Graph $\mathcal{G}(\mathcal{V},\mathcal{E},\mathcal{W},\mathcal{P})$, the
      Set of Query Vertices $\mathcal{Q} \subseteq \mathcal{V}(\mathcal{G})$,
      Distance Threshold $T$
Result: Candidate Skyline Vertices
        $\mathcal{CS} \subseteq \mathcal{V}(\mathcal{G}) \setminus \mathcal{Q}$
1  . Create Dictionary $\mathcal{D}$
2  for All $u \in \mathcal{Q}$ do
3      for All $v \in V(\mathcal{G}) \setminus \mathcal{Q}$ do
4          if Isconnected(uv) then
5              if $v \in \mathcal{D}.Keys()$ then
6                  $\mathcal{D}[v].values() = \mathcal{D}[v].values() \cup \{u\}$
7              else
8                  $\mathcal{D}.Create\_Key(v)$
9                  $\mathcal{D}[v].Add\_Value(u)$
10 $\mathcal{CS} = \emptyset$
11 for All $u \in \mathcal{D}.Keys()$ do
12     if $\mathcal{D}[u].Values() == \mathcal{Q}$ then
13         $\mathcal{CS} = \mathcal{CS} \cup \{u\}$
14 for All $v \in \mathcal{CS}$ do
15     for All $u \in \mathcal{Q}$ do
16         if $distance(uv) > T$ then
17             $\mathcal{CS} = \mathcal{CS} \setminus \{v\}$
```

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

Now, we do an analysis for time and space requirement of Algorithm 1. Let q be the number of query vertices, i.e., $|\mathcal{Q}| = q$. For creating the B.F.S. trees rooted at the query vertices requires $\mathcal{O}(q(m+n))$ time. The maximum number of values associated with a 'key' in the dictionary $\mathcal{D}$ is of $\mathcal{O}(q)$. Execution time from Line No. 2 to 9 and 11 to 17 requires $\mathcal{O}(q(n-q)^2)$ and $\mathcal{O}((n-q)q^2)$. Now, in distance-based pruning, the number of distance computations is $\mathcal{O}(q(n-q))$. Computing shortest path between two vertices in a weighted graph with positive edge weights requires $\mathcal{O}(m + n\log n)$ time. Hence, time requirement for distance-based pruning requires $\mathcal{O}(q(n-q)(m + n\log n))$ time. Total time requirement for Algorithm 1 is of $\mathcal{O}(q(m+n)+nq(n-q)+q(n-q)(m+n\log n)) = \mathcal{O}(q(n-q)(m+n\log n))$. Extra space requirement of Algorithm 1 is to store the dictionary $\mathcal{D}$, which is of $\mathcal{O}(q(n-q))$, to store the candidate skyline vertices, which is of $\mathcal{O}(n-q)$, and to perform the B.F.S., which is of $\mathcal{O}(n)$. Hence, total space requirement of Algorithm 1 is of $\mathcal{O}(q(n-q))$. Lemma 2 describes the formal statement.

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

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 $\mathcal{CS}$
Result: The Skyline Vertex Set $\mathcal{S}$
1  Generate $|\mathcal{R}|$ number of sample graphs
2  Store the graph probabilities in $\mathcal{P}_G[1 \dots |\mathcal{R}|]$
3  Create_Matrix $\mathcal{M} \in \mathbb{R}^{|\mathcal{CS}| \times |\mathcal{Q}|}$
4  for All $u \in \mathcal{CS}$ do
5      for All $v \in \mathcal{Q}$ do
6          Create dictionary $Temp$
7          for All $r \in \mathcal{R}$ do
8              $d$ = shortest distance between $u$ and $v$ in $r$
9              $Temp[d] = Temp[d] + \mathcal{P}_G[r]$
10         $\mathcal{M}[u][v] = \underset{d}{argmax}\ Temp[d]$
11 $\mathcal{S}$ = Apply BNL on $\mathcal{M}$
12 return $\mathcal{S}$
```

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

```
Data: Candidate Skyline Vertices $\mathcal{CS}$
Result: The Skyline Vertex Set $\mathcal{S}$
1  . Create_Matrix $\mathcal{M} \in \mathbb{R}^{|\mathcal{CS}| \times |\mathcal{Q}|}$
2  for All $u \in \mathcal{CS}$ do
3      for All $v \in \mathcal{Q}$ do
4          $path$ = Compute all paths from $u$ to $q$ upto length $l$
5          $prob[1 \dots |path|] = 0$ ; $dist[1 \dots |path|] = 0$
6          for All $t \in path$ do
7              for All $e \in \mathcal{E}(t)$ do
8                  $prob[t] = prob[t] + \mathcal{P}(e)$
9                  $dist[t] = dist[t] + \mathcal{P}(e) * \mathcal{W}(e)$
10         $\mathcal{M}[u][v] = \sum dist / \sum prob$
11 $\mathcal{S}$ = Apply BNL on $\mathcal{M}$
12 return $\mathcal{S}$
```

For the majority distance case, first we generate $|\mathcal{R}|$ number of subgraphs, as mentioned, while defining possible world semantics, and the corresponding generation probabilities are stored in the array $\mathcal{P}_G$. 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 $\mathcal{M}$ to obtain the skyline vertex set.

Now, we analyze Algorithm 2 for time and space requirement. As mentioned in the definition of possible world semantics, generation of $|\mathcal{R}|$ number of subgraphs require $\mathcal{O}(m|\mathcal{R}|)$ time. Using Dijkstra's algorithm computing the shortest path between a pair of vertices requires $\mathcal{O}(m + n\log n)$ time. Hence, execution time from Line 4 to 10 requires $\mathcal{O}(q(n-q)|\mathcal{R}|(m+n\log n))$. Now, BNL algorithm requires $\mathcal{O}((n-q)^2)$ time. Extra space consumed by Algorithm 2 is to store the array $\mathcal{P}_G$, $Temp$, and the matrix $\mathcal{M}$ which requires $\mathcal{O}(|\mathcal{R}|)$, $\mathcal{O}(|\mathcal{R}|)$, and $\mathcal{O}(q(n-q))$ space, respectively. The formal statement is given in Lemma 3.

Lemma 3 *Time and space requirement of Algorithm 2 is of* $\mathcal{O}(q(n-q)|\mathcal{R}|(m+n\log n)+(n-q)^2)$ *and* $\mathcal{O}(q(n-q)+|\mathcal{R}|)$*, 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* $\mathcal{O}(q(n-q)|\mathcal{R}|(m+n\log n)+(n-q)^2)$ *time and* $\mathcal{O}(q(n-q)+|\mathcal{R}|)$ *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 $\mathcal{O}(d_{max}^l)$. Hence, running time from Line 2 to 10 is of $\mathcal{O}(q(n-q)ld_{max}^l)$. Hence, total running time of Algorithm 3 is of $\mathcal{O}(q(n-q)ld_{max}^l+(n-q)^2)$. Extra space consumed by the Algorithm 3 is to store the matrix $\mathcal{M}$, array $Path$, $Prob$ and $dist$ which requires $\mathcal{O}(q(n-q)+ld_{max}^l)$. Hence, Lemma 4 holds.

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

4 Experimental Evaluations

In this section, we describe the experimental validations of our proposed approach. We have used three different **datasets**, appeared in two different contexts, which are described below.

- **Minnesota Road Network (MRN)** [17]: This is a road network dataset of the Minnasota city, with $n = 2642$, $m = 3300$, and avg. degree $= 2$. Here, the junctions are represented by the nodes. If two junctions are connected by a road, then the corresponding two vertices are connected by an edge.
- **P2P Network** [13]: This dataset contains a sequence of snapshots of the Gnutella peer-to-peer file sharing network from August 2002, with $n = 8114$, $m = 26013$, and avg. degree $= 6.41$. 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.

– **USA Road Network (URN)** [17]: This dataset describes a road network from the United States, with $n = 129164$, $m = 165435$, and avg. degree $= 2.56$. Here, vertices represent junctions, and an edge between signifies that the corresponding junctions are are connected by road.

All the datasets are undirected and unweighted. Probability of existence and weight of each edge is chosen from the intervals $(0, 1]$ and $[10, 100]$ uniformly at random. In our experiments, we consider l as 4. In our study the following query vertex selection strategies have been adopted, for the **experimental setup**.

– **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/. Now, the **goals of the experiments** are defined below.

– *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 the query size?

Now, we discuss the **experimental results** of the proposed approach.

Efficiency of the Pruning Strategies. It is easy to observe that the B.F.S pruning will return the vertices from the component in which the query vertices belong. As we have selected the query vertices from the largest component, the BFS pruning returns the vertices from the largest component only. For distance-based pruning, we have taken the threshold value as 400, considering 4-hop path with the maximum edge weight 100. In Fig. 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 the other two, in all the datasets, which is easy to convince. For P2P network, the interquartile range is very high compared to other datasets. This is due to the fact that the network is of high average degree. Also, for RAND selection strategy, this range is the highest for small query size. This is due to the existence of various small size components 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 for the MRN dataset.

Query Size Vs. Skyline Vertices. In Fig. 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

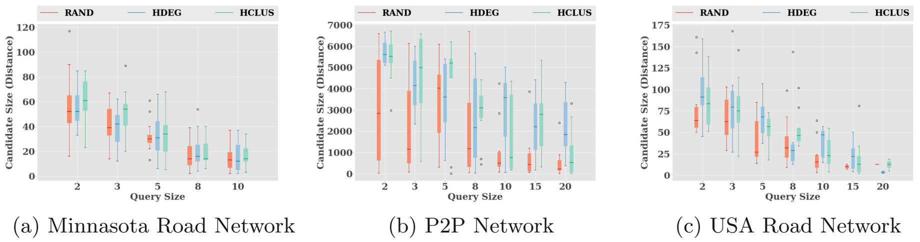


Fig. 2. Box plot for the candidate skyline size with respect to the query size for the Minnasota Road Network, P2P Network, and USA Road network datasets.

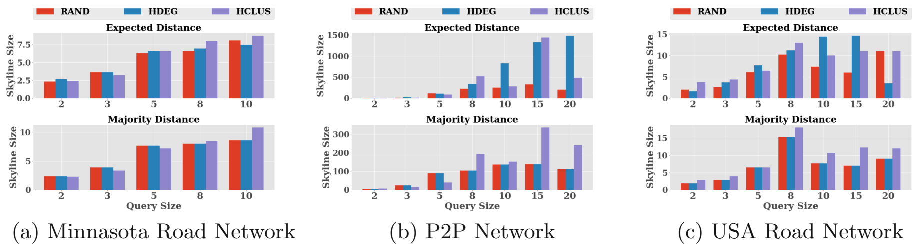


Fig. 3. Query size Vs. Skyline size plot for the Minnasota Road Network, P2P Network, and USA Road network datasets.

size, the skyline size increases. However, for the URN dataset in Fig. 3(c), the skyline size decreases for large value of query size. The reason is due to the small size of the candidate skyline, which can be verified from Fig. 2(c). Also, for both the road network datasets the maximum skyline size reaches approximately 15, whereas for the P2P network it reaches around 1500. This is due to its candidate size. For, both the cases, at a large value of query size, the ratio of candidate to skyline size is very small. As the number of query vertices increases, the possibility of domination decreases.

Distance Metric Vs. Skyline Vertices. In this part, referring to Fig. 3, we describe the behavior of skyline size with respect to different distance metrics. For the road networks in Fig. 3(a) and (c), the skyline size is similar in both the datasets. However, for the P2P network in Fig. 3(b), the skyline size in the expected distance (≈max 1500) is much more than the majority distance (≈max 300). The reason lies in the networks' high average degree value and 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 in the large size of the skyline vertex set. This can be verified from Fig. 3(b), by looking into HDEG 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 distances.

Query Selection Strategy Vs. Skyline Vertices. In this part, referring to Fig. 3, we describe the behavior of skyline size with respect to two 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 Fig. 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 the maximum skyline vertices.

Computational Time. Due to space limitation, we are unable to report the plots for the computational time. However, we briefly describe our key observations regarding this. 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, the required time for path length-based pruning, distance, and skyline computation (using BNL) increase. 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, for the P2P Network dataset, when the query size increases beyond 10, there is a sharp increase in the skyline computation time. This is because for higher query sizes the candidate and skyline size are more compared to the previous query sizes and observed in 2(b) and 3(b).

5 Conclusion

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 requirements. The experimental results demonstrate that it can find out the skyline vertex set with reasonable computation time.

References

1. Bonchi, F., Gullo, F., Kaltenbrunner, A., Volkovich, Y.: Core decomposition of uncertain graphs. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1316–1325. ACM (2014)
2. Borzsony, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings 17th International Conference on Data Engineering, pp. 421–430. IEEE (2001)
3. Chen, Y., Zhao, X., Lin, X., Wang, Y., Guo, D.: Efficient mining of frequent patterns on uncertain graphs. IEEE Trans. Knowl. Data Eng. **31**(2), 287–300 (2018)
4. Chomicki, J., Ciaccia, P., Meneghetti, N.: Skyline queries, front and back. ACM SIGMOD Rec. **42**(3), 6–18 (2013)
5. He, G., Chen, L., Zeng, C., Zheng, Q., Zhou, G.: Probabilistic skyline queries on uncertain time series. Neurocomputing **191**, 224–237 (2016)
6. Hu, J., Cheng, R., Huang, Z., Fang, Y., Luo, S.: On embedding uncertain graphs. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 157–166. ACM (2017)
7. Jin, R., Liu, L., Aggarwal, C.C.: Discovering highly reliable subgraphs in uncertain graphs. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 992–1000. ACM (2011)
8. Kassiano, V., Gounaris, A., Papadopoulos, A.N., Tsichlas, K.: Mining uncertain graphs: an overview. In: Sellis, T., Oikonomou, K. (eds.) ALGOCLOUD 2016. LNCS, vol. 10230, pp. 87–116. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57045-7_6
9. Ke, X., Khan, A., Quan, L.L.H.: An in-depth comparison of st reliability algorithms over uncertain graphs. Proc. VLDB Endowment **12**(8), 864–876 (2019)
10. Khan, A., Singh, V., Wu, J.: Finding skyline nodes in large networks. In: 28th International Conference on Data Engineering Workshops, pp. 198–204. IEEE (2012)
11. Khan, A., Ye, Y., Chen, L.: On uncertain graphs. Synth. Lect. Data Manage. **10**(1), 1–94 (2018)
12. Le, T.M.N., Cao, J., He, Z.: Answering skyline queries on probabilistic data using the dominance of probabilistic skyline tuples. Inf. Sci. **340**, 58–85 (2016)
13. Leskovec, J.: Gnutella peer-to-peer network, august 4 2002 (2002). https://snap.stanford.edu/data/p2p-Gnutella04.html

14. Miao, X., Gao, Y., Guo, S., Chen, G.: On efficiently answering why-not range-based skyline queries in road networks. IEEE Trans. Knowl. Data Eng. **30**(9), 1697–1711 (2018)
15. Park, Y., Min, J.K., Shim, K.: Processing of probabilistic skyline queries using mapreduce. Proc. VLDB Endowment **8**(12), 1406–1417 (2015)
16. Potamias, M., Bonchi, F., Gionis, A., Kollios, G.: Nearest-neighbor queries in probabilistic graphs. Technical report, Boston University Comp. Science Department (2009)
17. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI (2015). http://networkrepository.com
18. Yin, B., Gu, K., Wei, X., Zhou, S., Liu, Y.: A cost-efficient framework for finding prospective customers based on reverse skyline queries. Knowl. Based Syst. **152**, 117–135 (2018)
19. Zhang, K., Gao, H., Han, X., Cai, Z., Li, J.: Modeling and computing probabilistic skyline on incomplete data. IEEE Trans. Knowl. Data Eng. **32**, 1405–1418 (2019)
20. Zheng, W., Zou, L., Lian, X., Hong, L., Zhao, D.: Efficient subgraph skyline search over large graphs. In: Proceedings of the 23rd ACM International Conference on Information and Knowledge Management, pp. 1529–1538. ACM (2014)
21. Zhou, X., Li, K., Zhou, Y., Li, K.: Adaptive processing for distributed skyline queries over uncertain data. IEEE Trans. Knowl. Data Eng. **28**(2), 371–384 (2015)