Probabilistic Graphs and Influence Algorithms

Pablo Mollá Chárlez

Contents

1 Introduction

Probabilistic graphs are an extension of traditional graph models where edges, and sometimes nodes, have associated probabilities representing uncertainty or likelihood of connections. These graphs are particularly useful in scenarios where relationships are not deterministic, such as social networks, biological interactions, or communication networks.

Influence algorithms, on the other hand, focus on modeling and maximizing the spread of influence or information across such graphs. They play a critical role in areas like viral marketing, epidemic control, and network resilience. The interplay between probabilistic graphs and influence algorithms allows us to predict, optimize, and control how information or behaviors propagate through uncertain and complex systems.

2 Deterministic and Uncertain Graphs

2.1 Deterministic Graphs

A (deterministic) graph $G = (V, E)$ is defined as:

- A set V of vertices (nodes)
- A set $E \subseteq V \times V$, of edges

2.2 Uncertain Graphs

An uncertain graph $G = (V, E, p)$ is defined as:

- A set V of vertices (nodes)
- A set $E \subseteq V \times V$, of edges
- A function $p : E \to [0, 1]$, representing the probability $p(e)$ that the edge $e \in E$ exists or not

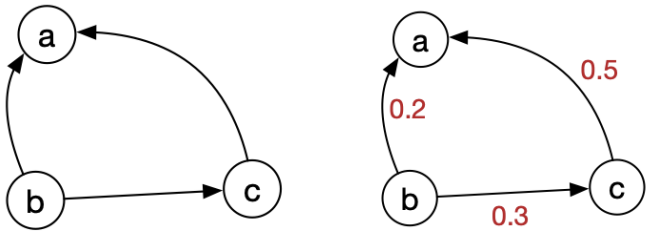

Figure 1: Deterministic and Uncertain Graph

2.3 Uncertain Graphs: Possible Worlds

A possible world of G, denoted $G' \sqsubseteq G$, is a deterministic graph $G' = (V, E_{G'})$, where each $e \in E_{G'}$ is chosen from E. The probability of G' is:

$$\Pr[G'] = \prod_{e \in E_{G'}} p(e) \cdot \prod_{e \in E \setminus E_{G'}} (1 - p(e))$$

In simple terms, a possible world is one specific version of an uncertain graph where we decide, for every edge, whether it exists or not based on its probability. It's like taking the "uncertainty" out of the graph and creating a fully connected or disconnected graph where each edge is either included or excluded. Each possible world represents one of the many potential configurations the uncertain graph could take.

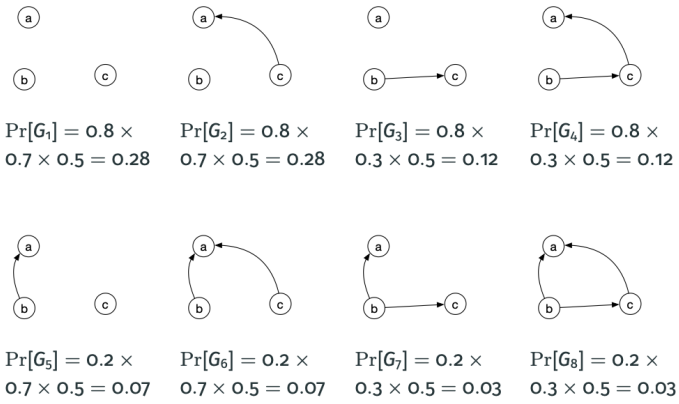

Figure 2: Possible World of a Directed Uncertain Graph

For instance, let's explain $Pr[G_2]$ for the possible world G_2, where only the edge $a \to c$ exists. using as the original uncertain graph 5. We define G_2 as the graph which has only $a \to c$. The probability of G_2 is calculated as:

$$Pr[G_2] = \prod_{e \in E_{G_2}} p(e) \cdot \prod_{e \notin E_{G_2}} (1 - p(e))$$

Where:

- $E_{G_2} = \{a \to c\}$: the edge that exists and its probability is $p(a \to c) = 0.5$.
- $E \setminus E_{G_2} = \{a \to b, b \to c\}$: edges that do not exist and their probabilities are $1 - 0.2 = 0.8$ and $1 - 0.3 = 0.7$ respectively.

Therefore, applying the possible world formula:

$$Pr[G_2] = p(a \to c) \cdot (1 - p(a \to b)) \cdot (1 - p(b \to c))$$

$$Pr[G_2] = 0.5 \cdot 0.8 \cdot 0.7 = 0.28$$

The total probability reflects the combined likelihood of these edges.

3 Queries on Graphs: Deterministic and Uncertain Graphs

3.1 Distance Queries: Reachability and Distance Distribution

In many real-world scenarios, we are interested in determining the distance between two nodes, whether working with deterministic or uncertain graphs:

- **Reachability (or reliability) query:** Find the probability that two nodes, s and t, are connected in the graph.
- **Distance distribution query:** Determine the probability distribution of distances between two nodes s and t, such as:

$$\Pr[d(s,t) = x] = \sum_{G \mid d_G(s,t) = x} \Pr[G]$$

where the summation is over all possible worlds G such that the distance $d_G(s,t)$ equals x.

3.1.1 Queries on Deterministic Graphs

For instance, considering the deterministic graph from 5, what is the distance (in hops) between b and a? The Breadth-First Search (BFS) algorithm or Dijkstra's algorithm can be used to find the shortest path or distance in terms of the number of hops. Since $b \to a$ exists in this example, the distance is exactly 1 hop and the cost of finding the distance is $\mathcal{O}(|E|)$, where $|E|$ is the number of edges in the graph, i.e., $\mathcal{O}(3)$.

3.1.2 Queries on Uncertain Graphs

In this case, we use the uncertain graph from 5 and we ask ourselves, what is the probability that we can reach a from b? We can answer the question from different approaches.

- **Logical Approach: Direct and Indirect Path**
  - **Direct Edge Contribution**: The direct edge $b \to a$ has a probability $p(b \to a) = 0.2$.
  - **Indirect Path Contribution**: If $b \to c$ (with $p = 0.3$) and $c \to a$ (with $p = 0.5$) both exist, then there is an alternative path $b \to c \to a$. The probability of this path is calculated as:

$$(1 - p(b \to a)) \cdot p(b \to c) \cdot p(c \to a) = 0.8 \cdot 0.3 \cdot 0.5 = 0.12$$

  - **Combination of Probabilities**:

$$\Pr[b \to a] = p(b \to a) + (1 - p(b \to a)) \cdot p(b \to c) \cdot p(c \to a) = 0.2 + 0.12 = 0.32$$

- **Counting Number of Possible Worlds**

  By counting the number of possible worlds (deterministic graphs) in which b is connected to a, either directly or indirectly, we can add up the probabilities of those worlds to arrive at:

$$\Pr[b \to a] = Pr[G4] + Pr[G5] + Pr[G6] + Pr[G7] + Pr[G8] = 0.32$$

  Both paths are valid when it comes to finding the probability of two nodes being reached.

Although, what about the distance distribution between b and a in the uncertain graph? The distance distribution will summarize all possible cases of those two nodes being connected. We calculate the probabilities for $d(b, a) = 1$ (b reaches a directly), $d(b, a) = 2$ (b reaches a in 2 steps or hops), and $d(b, a) = \infty$ (b can't reach a, neither directly nor indirectly). Previously, we computed the probability for the direct and indirect case:

- **Direct Path**: $\Pr[d(b, a) = 1] = p(b \to a) = 0.2$.
- **Indirect Path**: $\Pr[d(b, a) = 2] = (1 - \Pr[d(b, a) = 1]) \cdot p(b \to c) \cdot p(c \to a) = 0.12$.
- **No path**: $\Pr[d(b, a) = \infty] = 1 - \Pr[d(b, a) = 1] - \Pr[d(b, a) = 2] = 1 - 0.2 - 0.12 = 0.68$.

The distance distribution between b and a tells us:

- There's a 20% chance that b and a are directly connected ($d(b, a) = 1$).
- There's a 12% chance that b can reach a in 2 hops via the path $b \to c \to a$.
- There's a 68% chance that b and a are not connected in any possible world ($d(b, a) = \infty$).

The reality is that in real-world scenarios it becomes truly complicated:

- Distance query answering in uncertain graphs is at least as hard as in relational databases, where logical formulas of paths can lead to an exponential number of possible paths.
- Computing the reachability probability (i.e., the probability of a path existing between a source and a target) is known to be #P-hard, which is as hard as model counting.

3.2 Monte Carlo Sampling

Monte Carlo sampling is a statistical method used to estimate the probability of an event or the expected value of a quantity by generating random samples. In the context of uncertain graphs, Monte Carlo sampling can be used to estimate the reachability probability or the distance between nodes in a graph where edges have uncertain probabilities.

1. Generating sampled graphs for r rounds.
2. Computing the desired measure (reachability probability or distance distributions) by averaging the results from the sampled graphs.

The main issue in Monte Carlo sampling is determining the required number of rounds for accurate estimation.

3.2.1 Example of Monte Carlo Sampling

Consider a graph with nodes s, t, and some intermediate nodes, where **edges have associated probabilities of existing**. Let's assume that we want to estimate the reachability probability between s and t using Monte Carlo sampling.

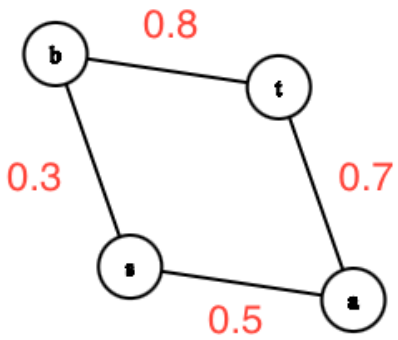

Figure 3: Monte Carlo Graph Example

To apply the sampling according to the Monte Carlo Sampling:

- In each round, we sample the graph by randomly choosing the edges that exist according to their probabilities. For example:
  1. **Round 1**: From s, the edge $s \to a$ exists (probability 0.5), and we move from $a \to t$ (probability 0.7), so we find a path.
  2. **Round 2**: From s, the edge $s \to b$ exists (probability 0.3), and we move from $b \to t$ (probability 0.8), so we find a path.
  3. **Round 3**: From s, neither $s \to a$ nor $s \to b$ exists, so no path is found.
- **Estimate**: After repeating the sampling for many rounds, we calculate the fraction of rounds where a path from s to t exists. This gives an estimate of the reachability probability.

3.2.2 Optimizing Sampling Graphs

Instead of generating the entire graph G_i for each round, which can be inefficient, we can optimize the process by focusing on the relevant edges. Here's how:

1. **Start from** s: We initiate a search from the source node s, and only sample the outgoing edges of s. This way, we avoid unnecessary computations for edges that are not directly relevant to the reachability between s and t.

2. **BFS/Dijkstra**: We perform a breadth-first search (BFS) or Dijkstra's algorithm on the sampled graph, but instead of exploring all edges, we only consider the outgoing edges of the current node, based on the sampled probabilities.

3. **Continue Search**: For each newly reached node, we repeat the process by sampling its outgoing edges, continuing the search until we either find t or exhaust all reachable nodes.

This procedure is more efficient and reduces the computational complexity due to multiple reasons:

- **Avoid Full Graph Generation**: By only sampling outgoing edges at each step, we avoid the cost of generating and storing the entire graph for each round.
- **Focus on Relevant Paths**: Since we're interested in paths from s to t, we focus only on the nodes and edges involved in those paths, making the search more efficient.
- **Re-use Computation**: Each sampled path is extended using previously computed results, avoiding redundant calculations.

4 Influence Maximization

4.1 Social Influence

Social Influence is an important problem in social networks, with applications in marketing, computational advertising, and many other areas. The objective is to maximize the expected influence spread in a social network, given a promotion budget of k social network users. This can be modeled by selecting an optimal seed set of users to maximize influence propagation.

A promotion budget k refers to the number of users that can be selected (or "seeded") to start a promotion or marketing campaign. The goal is to choose the k users in such a way that maximizes the spread of influence within the network. This budget represents the limited resources (e.g., **time**, **money**, or **effort**) available to activate these users and trigger a viral effect.

4.1.1 Data Model

The data model for social influence is represented by an uncertain graph $G(V, E, p)$, where:

- V and E represent the nodes and edges of the social network, respectively.
- p represents the influence probability for each edge. That is, for each edge $(u, v) \in E$, the influence probability p_{uv} quantifies the likelihood that u influences v.

4.2 Influence Spread via Cascades

In the following graph, we aim to understand how influence spreads in the network using a propagation model.

The **Independent Cascade Model** describes a discrete-time process of influence propagation:

1. At time 0, a seed node u is activated.
2. For a node i activated at time t, it attempts to activate each of its neighbors v at time $t+1$ with probability p_{iv}.
3. Once a node is activated, it cannot be reactivated or deactivated.

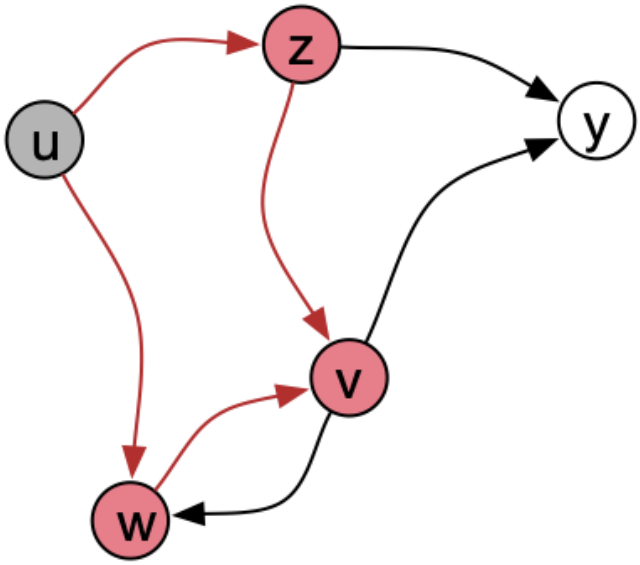

Figure 4: Influence Spread via Cascade Model

We aim to compute the expected influence spread from a seed set S, denoted as $\sigma(S)$. By the linearity of expectation:

$$\sigma(u) = \sum_{v \in V} \Pr[u \to v]$$

For a seed set S, the computation of influence spread is more complex and has the same computational hardness as reachability.

4.2.1 Example of Influence Spread

Consider the original directed probabilistic graph with edges:

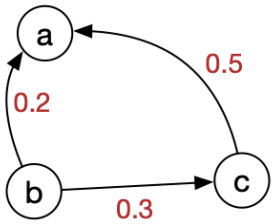

Figure 5: Original Uncertain Graph

The influence spread of each node in V is computed as:

$$\sigma(a) = \Pr[a \to a] + \Pr[a \to b] + \Pr[a \to c] = 1 + 0 + 0 = 1$$
$$\sigma(b) = \Pr[b \to a] + \Pr[b \to b] + \Pr[b \to c] = 0.32 + 1 + 0.3 = 1.62$$
$$\sigma(c) = \Pr[c \to a] + \Pr[c \to b] + \Pr[c \to c] = 0.5 + 0 + 1 = 1.5$$

4.3 Maximizing the Influence

Influence maximization is computationally hard due to the following two reasons:

1. Computing $\sigma(S)$ is #P-hard, as it is equivalent to reachability, which we previously saw is hard to compute. Monte Carlo methods with additive approximations are often used.
2. Computing the selection of k seeds in S is NP-hard, as it involves the maximization of a submodular function.

A function is submodular if for any two sets $S \subseteq T$ and an element $u \notin T$, the following holds:

$$\sigma(S \cup \{u\}) - \sigma(S) \geq \sigma(T \cup \{u\}) - \sigma(T)$$

In other words, the marginal gain of adding an element u to a smaller set S is at least as large as the marginal gain from adding u to a larger set T.

4.3.1 Influence Maximization: Greedy Algorithm

To approximate the influence maximization problem, we can use a greedy algorithm that provides a $(1 - \frac{1}{\epsilon})$-approximation factor:

1. Initialize $S = \varnothing$.
2. Choose the user u that maximizes $\sigma(S \cup \{u\}) - \sigma(S)$.
3. Update $S = S \cup \{u\}$.
4. Repeat steps 2 and 3 for k iterations.
5. Return the set S of selected users.

4.3.1.1 Example of Greedy Algorithm for Influence Maximization

We are given a small social network with 4 nodes $V = \{A, B, C, D\}$ and the following directed edges with influence probabilities p:

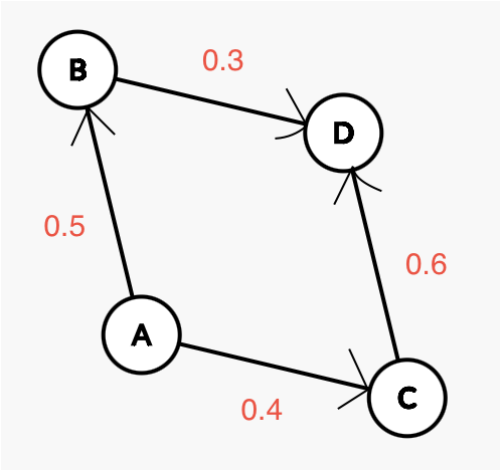

Figure 6: Greedy Algorithm: Maximizing Influence

We aim to maximize the expected influence spread by selecting $k = 2$ seed nodes.

Steps of the Greedy Algorithm

1. **Initialize:** Set $S = \varnothing$ (no seed nodes selected yet).
2. **Compute the marginal gain for each node:** For each node $u \notin S$, compute the influence spread $\sigma(S \cup \{u\}) - \sigma(S)$:

- Assume $S = \varnothing$:

$$\begin{aligned}
&\sigma(\{A\}) = 1 + 0.5 + 0.4 = 1.9,\\
&\sigma(\{B\}) = 1 + 0.3 = 1.3,\\
&\sigma(\{C\}) = 1 + 0.6 = 1.6,\\
&\sigma(\{D\}) = 1 \quad \text{(no outgoing influence from } D\text{)}.
\end{aligned}$$

  The node with the highest marginal gain is A ($\sigma = 1.9$).

3. **Add A to S:** Now $S = \{A\}$.

4. **Recompute the marginal gain for remaining nodes $\{B, C, D\}$:**

   - For $S = \{A\}$:

$$\begin{aligned}
&\sigma(\{A, B\}) = 1.9 + 0.3 = 2.2 \implies \sigma(\{A, B\}) - \sigma(\{A\}) = 0.3\\
&\sigma(\{A, C\}) = 1.9 + 0.6 = 2.5 \implies \sigma(\{A, C\}) - \sigma(\{A\}) = 0.6\\
&\sigma(\{A, D\}) = 1.9 \implies \sigma(\{A, D\}) - \sigma(\{A\}) = 0 \text{ (no additional gain)}.
\end{aligned}$$

   The node with the highest marginal gain is C.

5. **Add C to S:** Now $S = \{A, C\}$.

6. **Stop:** We have selected $k = 2$ seeds.

Using the greedy algorithm, the selected seed nodes are: $S = \{A, C\}$. These nodes maximize the expected influence spread in the network.

4.4 Learning Propagation Probabilities

The probability that node v is influenced by its neighbors is given by:

$$\Pr(v) = 1 - \prod_u (1 - p_{uv})$$

where p_{uv} is the probability that node u influences node v.

- Maximum Likelihood Estimation: Given a log of actions $A = \{(act, u, v), \dots\}$, the maximum likelihood estimate for p_{uv} is:

$$p_{uv} = \frac{A_{uv}}{A_v}$$

  where A_{uv} is the number of times node u influences node v, and A_v is the total number of actions involving node v.

- Jaccard Similarity: Another way to estimate the probability p_{uv} is using the Jaccard similarity:

$$p_{uv} = \frac{A_{uv}}{A_u \cup A_v}$$

  where A_u is the set of neighbors of u, and A_v is the set of neighbors of v.