USIWO: A Local Community Search Algorithm for Uncertain Graphs

Yashar Talebirad*, Mohammadmahdi Zafarmand*, Osmar R. Zaïane*, Christine Largeron†
*Alberta Machine Intelligence Institute, University of Alberta, Edmonton, Canada
†Hubert Curien Laboratory, Université Jean Monnet, Saint-Etienne, France
Email: talebira@ualberta.ca, m.zafarmand@ualberta.ca, zaiane@ualberta.ca, largeron@univ-st-etienne.fr

***Abstract*—Community detection and community search are both critical tasks in graph mining, each serving unique purposes and presenting distinct challenges. The former aims to partition the graph vertices into densely connected subsets, while the latter adopts a more ego-centric approach, focusing on a specific node or group of nodes to identify a densely-connected sub-graph that contains these query nodes. However, many real-world networks are characterized by uncertainty, leading to the notion of uncertain or probabilistic graphs. The transition from deterministic graphs to uncertain graphs introduces new challenges. We present *USIWO*, an efficient and practical solution for community search in unweighted uncertain graphs with edge uncertainty. In addition to being accurate, the approach utilizes an efficient data structure for storing only the relevant parts of the network in main memory, eliminating the need to store the entire graph, making it a valuable tool in finding the core of a community on very large uncertain graphs, when there is limited time and memory available. The algorithm operates through a one-node-expansion approach, based on the concepts of strong and weak links within a graph. Experimental results on several datasets demonstrate the algorithm's efficiency and performance.**

I. Introduction

In the era of big data, complex networks have emerged as a powerful tool to model relationships between entities within data. These networks, often represented as graphs, consist of a finite number of nodes and edges that connect them. However, traditional graphs, where relationships between nodes are deterministic, often fall short when modelling real-world phenomena where uncertainty is inherent. In fact, numerous real-world networks are characterized by uncertainty, which can arise from various sources such as the data collection process or pre-processing methods using machine learning [14]. This has led to the development of "Uncertain Graphs" also known as "Probabilistic Graphs", which incorporate uncertainty into the graph structure. This uncertainty can be associated with one or several components of an uncertain graph: edges, nodes, and attributes. Understanding these different types of uncertainty is crucial as it directly impacts the potential applications of these graphs. For instance, uncertain graphs with edge uncertainty are often used to represent noisy connections between data, such as in Protein-Protein Interaction (PPI) networks, social media, and brain activity representation [20].

ASONAM '23, November 6-9, 2023, Kusadasi, Turkey
© 2023 Association for Computing Machinery.
ACM ISBN 979-8-4007-0409-3/23/11…$15.00
https://doi.org/10.1145/3625007.3627337

While studying graphs that stem from real world networks, two tasks have gained significant attention due to their potential to reveal hidden structures and patterns in the network: community detection and community search. These tasks, while sharing similarities, serve unique purposes and present distinct challenges. Community detection aims to partition the graph vertices into subsets, taking the edge structure of the graph into account. The goal is to create groups where there are many edges within each community and relatively few between them. However, this approach can face limitations when applied to large, dense networks due to their focus on the overall structure of the graph that cannot fit in main memory.

In contrast, community search adopts a more ego-centric approach, focusing on a specific node or a group of nodes. The aim here is to find a densely-connected sub-graph, or "community", that contains all query nodes. This task is less concerned with the overall structure of the graph and more with the immediate surroundings of the query nodes. Thus, while community detection provides a broad view of the graph's structure, community search offers a more localized perspective, making it a valuable tool for exploring specific areas of interest within a (especially large) graph. It can be noted however that community search can be called iteratively on unprocessed nodes until all communities are discovered.

Although these issues have been widely researched on deterministic graphs, transitioning to uncertain graphs introduces new complexities, as it necessitates the redefinition of existing problems and the development of novel algorithms. A significant challenge persists due to the absence of definitive algorithms for correctly partitioning uncertain graphs in polynomial time without certain shortcomings [20]. Moreover, naive methods for community search or clustering in uncertain graphs, such as using edge probabilities as weights or setting a threshold value to the probabilities of the edges, have been shown to generate low-accuracy results in practice [12].

Given these challenges, local community search, which is executable on large graphs, presents a promising avenue for community detection in uncertain graphs, as it allows for analysis on large-scale networks without the need to partition the entire graph. This approach can offer significant benefits across a broad spectrum of application domains. For instance, in biology, it can aid in the detection of functional modules, thereby facilitating critical clinical diagnoses of diseases like cancer. As interest grows in dense sub-graphs such as k-core and k-truss in uncertain graphs, the challenge of extending community models and search techniques to uncertain graphs becomes increasingly more important [14]. In response to these needs, we aim to contribute to the field by proposing a new algorithm for community search in unweighted uncertain graphs that have independent probabilities on edges. This algorithm, called *USIWO*, builds upon the notion of "Edge Strengths" that we introduced in [6], [26]. We adapt this notion to accommodate uncertain graphs, thereby offering a practical solution for community search in uncertain graphs.

The remainder of this paper is organized as follows: Section II reviews the work that is related to performing community mining or community search on uncertain graphs. Necessary preliminary concepts are presented in Section III. Section IV introduces and discusses our proposed algorithm, *USIWO*, for community search in uncertain graphs. In Section V we evaluate the performance of *USIWO* in generating communities and compare it with several other algorithms. Finally, Section VI lists possible directions for further research in this field and concludes with a summary of the findings.

II. Related Work

In this section, we take a close look at the relevant literature and existing methodologies focused on uncertain graphs with an emphasis on two key areas: Community Mining, and Community Search. Although these tasks are different in nature, there is a strong parallel between them since one could perform partitioning by iteratively applying a community search algorithm on a query node, removing the found community from the network, and repeating this process until the entire network is partitioned into communities. Each community detected by the algorithm corresponds to a cluster in the network. Therefore, while our literature review discusses community mining algorithms (often referred to as clustering by misuse of language), the techniques used in these algorithms are highly relevant to our primary focus on community search.

It is important to note that in addition to uncertain graphs, other models like Fuzzy graphs and Fuzzy Granular Social Networks can be used to model uncertainty in networks [9]. However, uncertain graphs use probabilities to represent uncertainty, whereas Fuzzy Graphs use membership functions, and Fuzzy Granular Social Networks leverage granules.

A. Community Mining a.k.a. Clustering

Potamias et al. [21] and Liu et al. [17] laid the groundwork for uncertain graph partitioning, with subsequent studies employing various approaches. For their part, Liu et al. [17] introduced the concepts of "Purity" and "Size Balance" in an information theoretic framework to propose the method coded-k-means for accurate clustering, but the method proved ineffective for small-world graphs and needed prior knowledge about cluster numbers. Hu et al. [13] introduced URGE (Uncertain Graph Embedding), a method of embedding uncertain graphs into a set of low-dimensional vectors that carry important information about the nodes' proximity. Li et al. [16] proposed AUG-I and AUG-U for uncertain graphs with node attributes. These attributes are incorporated to reduce edge uncertainty, and distance metric learning is used to transform the uncertain attributed graph into a deterministic weighted graph. AUG-I and AUG-U showed better performance in comparison to the coded-k-means algorithm [17].

Ceccarello et al. [3] proposed MIN-PARTIAL, an algorithm to partition an uncertain graph into k clusters. The main idea is that nodes within a cluster have higher connectivity than nodes between clusters. MIN-PARTIAL iteratively produces a partial k-clustering to cover maximal subsets of nodes, given a threshold on the minimum connection probability of a node to its cluster center, eventually creating full k-clusterings. The algorithm's effectiveness is compared with other contenders (MCL [24] and GMM [8]). As MIN-PARTIAL [3] provided weak approximation guarantees and required significant computational overhead, Han et al. [11] proposed an improved version that addresses these points; however they only compare its performance with Ceccarello's work. Martin et al. [19] proposed a maximum-likelihood method for inferring community structure in uncertain networks, which involves modeling the probability of observing each edge in the network as a function of the underlying community structure. The base assumption is that the probability of an edge between two nodes is higher if they belong to the same community.

All the methods above, target the uncertain network as a whole and may not be scalable when the network is large and does not fit in main memory.

B. Local Community Search

Unlike Community Mining, which focuses on partitioning the entire graph, Community Search is more targeted and looks for a specific community that includes the user's query node. Community Search is sometimes referred as Local Community Detection [1]. These methods, cited below, are closely related to ours, and they follow the same greedy expansion methodology of *USIWO*.

Halim et al. [10] proposed a density-based approach, called DBCLPG (Density-Based Clustering of Large Probabilistic Graphs), that starts with a single query node, well chosen, as a cluster. It then greedily adds nodes from the shell set (called the cluster neighbors) to the current community, based on a condition to only add "reliable" nodes. The reliability of a node is determined by a threshold edge weight (T_w). For each candidate node v, the number of common neighbors between v and the cluster are computed and added to the sum of edges' probabilities (which is the expected degree) between the cluster and v. If this value (called the CP or cluster periphery of the node) is higher than T_w, the node is added to the cluster. The process is repeated until no node in the shell set satisfies the condition. The cluster is then removed from the graph and the procedure is repeated for the remaining graph. This approach is effective for identifying dense communities in uncertain networks, but it may not perform well on networks with complex structures, due to the assumption of high local density for cluster formation. Moreover, the method differs from a typical community search as the process starts from a well chosen node with the aim of partitioning the graph. In addition, to choose an optimal node to start with, the approach needs the full adjacency matrix, making it not local.

Zhang and Zaïane [27] propose an algorithm called $UR{+}K$, which starts with a single node and expands the community by adding neighboring nodes that are considered to be part of the community, using the metrics UR and K. These neighboring nodes that have the potential to be inside the community are referred to as candidate nodes. K measures the closeness of the relationship between a candidate node and an existing community. A high K value indicates that the candidate node is closely related to the community and is more likely to be part of it. K is used in the first few steps of the local community detection algorithm to help choose which neighboring node should be added to the community. UR, on the other hand, stands for the uncertain version of modularity R, introduced by Clauset in [4], is proposed for evaluating local communities in uncertain networks. UR is calculated based on the expected number of edges within the community and the expected number of edges connecting the community to the rest of the network. A high UR value indicates a sharp boundary between the community and the rest of the network, meaning that the community is well-defined. The algorithm starts by placing the start node in the community and its neighbors in the shell set. At each step, candidate nodes in the shell set are sorted based on their metric (K or UR) values, and the first node that can increase the community's metric is added to the community. This process continues until there are no remaining nodes in the shell node set that can increase the community's metric, and the resulting set of nodes is considered to be a local community. The hyper-parameter λ determines how many steps K is considered as the main metric before switching to UR. The paper shows that $UR + K$ outperforms other local community detection algorithms (including ULouvain, the weighted variant of Louvain [2] considering edge probabilities as weights) on real-world and synthetic networks.

III. Preliminaries

We now lay the groundwork for the concepts and terminologies that will be frequently referenced throughout this paper. We begin by explaining the notion of "Edge Strength" or "Link Strength" that is introduced in [6], [26]. Each edge is assigned a strength value in the range of $(-1, 1)$, with stronger edges corresponding to larger weights. The strength of the connection between two nodes is calculated based on the number of neighbors they share, which is referred to as their support value. More specifically, the support of an edge between two nodes is the number of triangles, or size-three cliques, that include both nodes:

Definition 1: **(Support)** Given a graph $G = (V, E)$, the support of the edge $e_{u,v}$ is the number of mutual neighbors of u and v or the number of triangles that $e_{u,v}$ belongs to, and it is defined as follows:

$$sup(u, v) = |w \in V, e_{u,w}, e_{v,w} \in E|$$

The strength of the link between two nodes is then calculated based on their support value and the maximum support value of any link involving either node:

Definition 2: **(Link's strength)** Given a graph G=(V, E) and a vertex $u \in V$, the strength $s_{u,v}$ of the link connecting u to a node $v \in V_N(u)$, where $V_N(u)$ is the set of neighbors of u, is defined as follows:

$$s_{u,v} = sup(u, v)(\frac{1}{sup_{u,max}} + \frac{1}{sup_{v,max}}) - 1$$

where $sup_{u,max}$ is the maximum support of u and any node in $V_N(u)$: $sup_{u,max} = \max_{w \in V_N(u)}\{sup(u, w)\}$.

The concept of Link Strength, which leads to the *SIWO* objective function, forms the foundation of the work presented in [6]. This approach enables community discovery in a network through a greedy optimization process, which iteratively performs two main phases until a local maximum of the SIWO measure is reached. However, approaches that take the whole graph into account are often impractical in real-world scenarios where the input graph is too large to fit into the main memory. To address this, we adapt these concepts to be applicable to uncertain graphs and propose a method that utilizes the new definitions in an ego-centric manner. The result is a local community search algorithm for uncertain graphs, which we refer to as *USIWO*.

We now turn our attention to the formal definition of uncertain graphs. The simplest and most widely used definition of uncertain graphs, introduced in [21], is akin to:

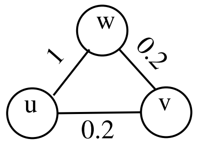
Definition 3: An uncertain graph $\mathcal{G} = (V, E, p)$ where $p : E \to (0, 1]$ can be viewed as a probability space whose outcomes are sub-graphs $G^{'} = (V, E^{'})$ of $\mathcal{G}$ where any edge $e \in E$ is included in E' with probability $p(e)$ independently. These possible outcomes are also referred to as "Possible Worlds", each representing a potential realization of the uncertain graph [22].

In *USIWO*, we start by putting a query node inside a community. We then identify the shell set of this community. The shell of a community C, denoted by $shell(C)$, is the set of nodes that are not in the community but are connected to at least one node in the community. Formally, it is defined as:

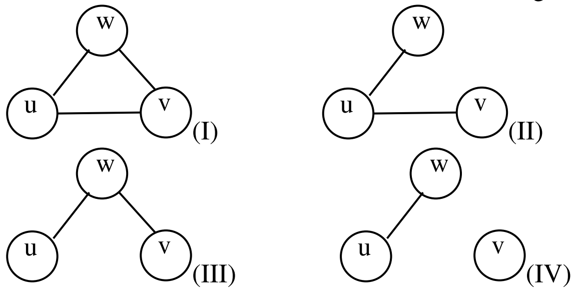
Definition 4: (Shell set) Given an uncertain graph $G = (V, E, p)$, the shell of the community C is defined by:

$$shell(C) = \{v \in V, v \notin C \text{ s.t. } \exists u \in C, p(e_{u,v}) > 0\}$$

We then proceed by finding the best candidate node inside the shell set. We expand the community by adding the node that has the strongest connection to the current community. We iteratively repeat this process until the addition of the best candidate node is considered to be too insignificant for the current community. To determine how strong a connection is, we adopt a similar definition of link strengths and support values. The support of an edge between two nodes in a deterministic graph is the number of triangles, or size-three cliques. However, in an uncertain graph, the process of counting these cliques becomes more complex. Rather than counting the shared neighbors, we must compute their expected count. To illustrate how we can modify the support value formula to account for probabilistic cases, consider the following probabilistic structure:



To count the expected number of triangles between u and v, we need to check all possible worlds. In all possible worlds, there is an edge between u and w, so we only need to consider the other two edges. There are 4 possible worlds corresponding to the existence and non-existence of these two edges:



We can observe that among these four possible worlds, only one of them (I) contains a triangle between the three nodes. With u, v, w there are a maximum of $2^3 = 8$ possible worlds regardless of their edge probability, and only one possible world will have the 3 edges occurring (i.e. a triangle). The probability of that possible world occurring is the product of the probabilities of the 3 edges: $P = 1 \times 0.2 \times 0.2 = 0.04$. Since only one possible world will have the 3 edges occurring (i.e. a triangle), the expected number of triangles $E_w[N_{u,v}]$ between nodes u and v related to w is defined as follows:

$$E_w[N_{u,v}] = (p_{u,w} \times p_{w,v} \times p_{u,v})$$

Where $p_{i,j}$ is shorthand for $p(e_{i,j})$, or the probability assigned to the edge between i and j in the graph. Having defined this notion, we can now define the probabilistic support. In a probabilistic graph, the support value of an edge, corresponding to the expected number of triangles between the two linked nodes, is defined as follows:

Definition 5: **(Probabilistic Support)** Given an uncertain graph $\mathcal{G} = (V, E, p)$, the support of the edge $e_{u,v}$ is the expected number of mutual neighbors of u and v or the number of triangles that $e_{u,v}$ belongs to, and it is defined as follows:

$$sup(u, v) = \sum_{w \in V_N(u) \cap V_N(v), w \notin \{u,v\}} E_w[N_{u,v}]$$

Where $V_N(u)$ denotes the neighborhood of the node u defined as the nodes in the graph for which there is a non-zero probability of an edge between them and u. It is important to note that the nodes w contributing to the summation must be part of the shared neighborhood of both nodes u and v. This is because if a node w is not a mutual neighbor of u and v, either $p_{u,w}$ or $p_{v,w}$ would be zero, rendering the entire term zero in the summation. Therefore, only those nodes w that are mutual neighbors of u and v contribute non-zero terms to the summation, reflecting their role in the formation of potential triangles involving u and v. The strength of the link between two nodes is calculated similarly to the deterministic case based on their support value and the maximum support value of any link involving either node. The link strengths are scaled in such a way that they take a value in $[0, 1]$:

Definition 6: **(Probabilistic Strength)** Given an uncertain graph $\mathcal{G} = (V, E, p)$, and a vertex $u \in V$, the strength of the link connecting u to a node $v \in V_N(u)$ is defined as follows:

$$s_{u,v} = \frac{sup(u, v)}{2} \left(\frac{1}{sup_{u,max}} + \frac{1}{sup_{v,max}} \right)$$

Where $sup_{u,max} = \max_{w \in V_N(u)} \{sup(u, w)\}$ is the maximum support of u and any node in $V_N(u)$.

Another important concept is the notion of Peripheral Nodes. In the context of community detection, these nodes often represent outliers or unique entities that are only loosely connected to a community. Since our method involves counting triangles, peripheral nodes that are not a part of any triangle should be considered at the last stage of our algorithm for potential addition to the found community.

Definition 7: **(Peripheral Nodes)** In a deterministic or uncertain graph, peripheral nodes are nodes that are connected to the rest of the graph through a single edge.

By utilizing these definitions, we can follow a greedy approach by starting with a query node, calculating the support and strength values for edges between the community (initially the query node) and nodes in the shell set (initially consisting of the neighbors of the query node), adding the best node from the shell set, and repeating until a stopping condition is met.

The distinctive approach of our algorithm lies in its utilization of strength values on edges to normalize the number of triangles containing that edge. This normalization, achieved by using $sup_{u,max}$ as the denominator in Definitions 2 and 6, aims to offer a consistent relative measure of strength throughout the graph, regardless of the nodes' degree. For instance, a node with a high degree might have a number of mutual neighbors with another node, but the strength value may not be as significant when calculated using Definition 6. Conversely, for nodes with a smaller degree, even a few mutual neighbors can be of significant importance.

IV. Methodology

In this section, we propose the detailed methodology of our proposed *USIWO* algorithm for community search in uncertain graphs. Building upon the foundational concepts and definitions of support and strength values established in the previous section, we outline the step-by-step process of our approach. This includes the initialization phase, the iterative process of node expansion, and the stopping condition.

USIWO begins by placing the query node in an empty community. It then locally explores the network to identify the most suitable node in the community's neighborhood to expand the community by one node at a time. This iterative process continues until the desired community around the given query node is found. The method involves five key steps, as outlined in Algorithm 1:

Update Shell Set: The shell set, initially empty, is updated after placing the query node u in the community C. All nodes connected to u are added to the shell S. If the query node is a peripheral node, its single neighbor replaces the query node. The shell set is updated in subsequent rounds by removing the node v that joined C in the previous round and adding nodes directly connected to v that are not already in C. The nodes in the shell set are called "candidate nodes", since they represent potential candidates for new nodes that may be added to the community.

Assign Strength Values: Following Definition 6, the strength value $s_{u,v}$ is computed for each pair of nodes (u, v) where $u \in C$ and $v \in S$. This process is performed locally, meaning it does not require access to the entire network. To avoid recalculation, we can also make sure we only calculate what is needed in each step and store the calculated strength values in a data structure.

Select Best Candidate Node: After edge strengths are determined, the strength that each potential candidate node v can bring to the current community is computed. This value, denoted by $s(C, v)$, is the sum of the strength values of all edges between the community and v:

$$s(C, v) = \sum_{u \in C} s_{u,v}$$

The node corresponding to the largest $s(C, v)$ is declared the best candidate.

Expand Community: The community is expanded by adding the selected node, but only if its addition will result in a significant increase in the overall strength of the community (calculated by adding the strength values of all the edges inside the community). This "significance" is determined by the stopping threshold δ, which is a threshold that can be set depending on how large we want the resulting communities to be. Thus, for the best candidate node v, we check if $s(C, v) > \delta$. In that case, the algorithm returns to the first step to update the shell set. If no new node can be added, (i.e., there is no node v such that the sum of the strength values between v and the nodes inside the community would exceed δ), the algorithm proceeds to the final step.

Reform Community: The final step of the algorithm is to add any peripheral node that has a neighbor in the current community. Without this step, these nodes do not have the chance to join the community, because the edge that connects them to the rest of the graph cannot be a part of any triangle.

The proposed method of greedy expansion ensures that the final detected community is a connected sub-graph and that a node joins C only if it improves its strength. The process repeats until an optimal community structure is obtained. Although this greedy approach provides us with a community "search" algorithm with the objective of finding the best community around a given query node, it can be extended to community mining by repeatedly applying the community search process to random query nodes and removing the found community from the network, defining it as a distinct community (or cluster). This strategy offers a practical approach to community detection in uncertain graphs, particularly when dealing with large graph structures.

Below is a summary of the proposed algorithm in the form of a pseudo-code:

Algorithm 1 *USIWO*: A local community search algorithm

```
Input: Uncertain Graph $G = (V, E, p)$, query node(s) $\{q\}$, and the stopping threshold $\delta$
Output: The community $C$ of the query node(s) $\{q\}$

$C = \{q\}$, $S = V_N(q)$
while $S \neq \emptyset$ do
  Calculate and store $s(C, v) = \sum_{u \in C} s_{u,v}$ for all $v \in S$
  Find the node $u \in S$ which maximizes $s(C, u)$
  if $s(C, u) > \delta$ then
    $C = C \cup \{u\}$
    $S = S \cup (V_N(u) - C) - \{u\}$
  else
    break
  end if
end while
$C = C \cup P$ where $P$ is the set of peripheral nodes that are connected to a node in $C$
return $C$
```

Furthermore, to make the algorithm truly local, we use an efficient data structure to store the necessary parts of the graph in memory. Most algorithms in the literature that are designed for community detection or community search, load the entire network into main memory in their first step, which is not efficient for large networks. To address this, *USIWO* only loads the necessary edges and calculates their strengths. The original input file for uncertain graphs is reformatted to facilitate this approach. Each line in the reformatted file stores the neighborhood information of a node along with their probabilities, making it easier to access the necessary nodes.

The selection of an appropriate stopping condition, combined with the probabilistic support values, can yield optimal results, provided a suitable threshold value is chosen. In the experiments section, we provide evidence supporting this claim and detail how we determined an appropriate threshold value through a series of experiments.

V. Experiments

The aim of the experiments is to assess the performance of *USIWO* in comparison with other existing methods.

A. Datasets

Due to the scarcity of probabilistic networks with ground truth available online, we have employed a method to convert deterministic graphs into uncertain graphs. Our method is a variation of the network generator proposed in [27]. This generator was chosen for its ability to create large synthetic uncertain graphs, allowing us to compare the output with the ground truth. In particular, we use this method to convert LFR [15] networks and real-world networks with ground truth to uncertain graphs. The generation process is outlined below:

Inputs: The algorithm takes a deterministic network G with the ground-truth communities as input. It also takes two parameters: p_{intra} and p_{inter}. These parameters define the range of possible probability values for intra-community links and inter-community links respectively. Both p_{intra} and p_{inter} are bounded between 0 and 1, where p_{intra} defines the lower bound for intra-community link probabilities and p_{inter} defines the lower bound for inter-community link probabilities.

Generating Probabilities: The process of converting deterministic networks into uncertain networks begins by identifying the links as either intra-community links or inter-community links based on the ground-truth communities. For each intra-community link, a probability value is generated using a uniform distribution, ranging between the p_{intra} value and 1. Similarly, for each inter-community link, a probability value is generated using the same uniform distribution, but between the p_{inter} value and 1.

Assigning Probabilities: The generated probability values are then assigned to their respective links, turning the deterministic network into an uncertain network while preserving the community structure from the original deterministic network and does not compromise the preset ground truth.

For the purpose of our experiments, we also introduce the concept of "complexity" to describe different variations of added uncertainty. The complexity of an uncertain graph is determined by the parameters p_{intra} and p_{inter}, as well as the method used to assign probability values to the edges. We define four levels of complexity:

Complexity 1: Probability values are first generated uniformly at random in [0,1]. These values are then sorted in descending order and assigned to intra-community edges and inter-community edges, respectively. This ensures that intra-community edges are always stronger and the ground truth is not compromised.

Complexity 2: The lower bound for intra-community link probabilities (p_{intra}) is set to 0.6, while the lower bound for inter-community link probabilities (p_{inter}) is set to 0. This creates a scenario where intra-community links (which are given probabilities in range $[0.6, 1]$) are generally stronger than inter-community links (with probabilities in range $[0, 1]$).

Complexity 3: Both p_{intra} and p_{inter} are set to 0.6, creating a scenario where both intra-community and inter-community links can be strong.

Complexity 4: In addition to the settings of Complexity 3, new probabilistic edges are added between communities with a low probability (0.01). This introduces additional uncertainty and can potentially make community detection more challenging.

The complexity levels were designed to progressively increase the difficulty of the community detection task, and create multiple variations of uncertain networks from a given deterministic network. Complexity 1 represents the simplest scenario, where intra-community edges are always stronger than inter-community ones. Complexity 2 introduces some uncertainty by allowing inter-community edges to be potentially as strong as intra-community edges. Complexity 3 further increases the uncertainty by making the strength of intra- and inter-community edges indistinguishable on average. Finally, Complexity 4 represents the most challenging scenario, where new low-probability edges are added between communities, further blurring the community boundaries. This way, we are able to examine the performance of community detection algorithms under different uncertainty scenarios.

1) Real-world Networks: For the experiments on the real-world graphs, we consider three well-known networks with a known community structure, namely, the Karate [25], Football [7] and Dolphins [18] networks. These networks were also used as potential graphs for experiments in works of [5] and [27]. Each of these deterministic networks are then converted to four uncertain networks using the explained method.

2) Synthetic Networks: To further evaluate the algorithms on a network with ground truth, we used our conversion method to convert an LFR network [15] with 2500 nodes into a probabilistic network. The parameters for generating the LFR network were set as follows: the power law exponent for the degree distribution τ_1 was set to 3, the power law exponent for the community size distribution τ_2 was set to 1.5. These were selected in such a way that the network would have a realistic degree distribution and community size distribution, respectively. The fraction of inter-community edges incident to each node μ was set to 0.2, and the average and maximum degree of nodes were set to 10 and 30 respectively, to cause the network to have a moderate density. Additionally, to test scalability, we used the same parameters to generate a much larger LFR network with 1 million nodes and 5,774,105 edges. This time, we chose 1 percent of all edges and sampled their intra-community edge probabilities and inter-community edge probabilities from [0.7, 1.0] and [0.0, 0.3], respectively.

B. Optimal Threshold Value

Before testing the effectiveness of our method on the converted graphs, we conducted a series of experiments to determine the optimal threshold value for the stopping condition. These experiments were performed in two stages. In the initial stage of our experiments, we explored a range of threshold values, incrementing by 0.1 from 0.1 to 1.5. We applied the algorithm to several query nodes and evaluated the average F_1 measure against the threshold for three real-world graphs: Football, Karate, and Dolphins. For each graph, one query node was selected from each community. These experiments suggested that an effective threshold value consistently fell within the range of 0.8 to 0.9, regardless of the input graph or the specific nodes chosen. To further refine our threshold selection, we conducted a an additional set of experiments, adjusting the threshold between 0.8 and 0.9 in increments of 0.01. The outcomes of these experiments led us to identify an optimal threshold value of 0.82 within this range. This value was then applied as the stopping condition for the *USIWO* algorithm in all subsequent experiments.

C. Algorithms

When it comes to community search in uncertain graphs, the number of direct competitors is limited. The most notable method in this category is the UR+K method proposed by Zhang and Zaïane [27]. We ran the community search algorithms, namely *USIWO* and UR+K, for each query node in the uncertain graphs. The resulting communities were then compared with the ground truth communities. We computed the usual precision, recall, and F_1 measures for the results of each query node. The final reported metrics for these algorithms are the averages of these individual results.

Given the scarcity of direct competitors in the field of community search, we have also included in our comparison other algorithms that are primarily used for community mining. This approach allows us to provide a more comprehensive evaluation of our method. We first considered the DBCLPG method proposed by Halim et al. [10], which has shown promising results in detecting communities in uncertain graphs. Furthermore, we also included the Leiden algorithm [23] in our comparison. The Leiden algorithm is widely recognized as one of the most effective algorithms for community detection in deterministic graphs, and it has been shown to outperform the Louvain algorithm [2]. We included both the weighted and unweighted versions of the Leiden algorithm in our comparison. Even though the Leiden algorithm is originally designed for deterministic graphs, for the purpose of our experiments, we adapted it to work with uncertain graphs. For the weighted version of the Leiden algorithm, we treated the edge probabilities as weights. For the unweighted version, we ignored the edge probabilities altogether and treated all edges as if they do not have a weight. This approach allowed us to apply the Leiden algorithm to uncertain graphs and include it in our comparison.

For the community mining algorithms, which include the two versions of the Leiden algorithm and DBCLPG, and for each community found by these algorithms, we associated each discovered community with all its contained query nodes. We then proceeded as before, computing the precision, recall, and F_1 measure for each query node, then averaged these metrics to obtain the final results. Note that choosing bad starting nodes, such as those on the boundaries of a community, could potentially lead to poor results. By attributing the communities found via the optimal starting node to all nodes inside that community instead, this starting node issue is mitigated, which gives these methods an advantage over *USIWO* and UR+K.

D. Results and Discussion

To evaluate the performance of our algorithm, we ran *USIWO* on each node of the three real-world networks (Karate, Football, and Dolphins), and on 100 nodes of the synthetic network. We then computed and recorded the precision, recall, and F_1 measure, and computed the average of these measures across the nodes. Figures 1 and 2 plot the average F_1

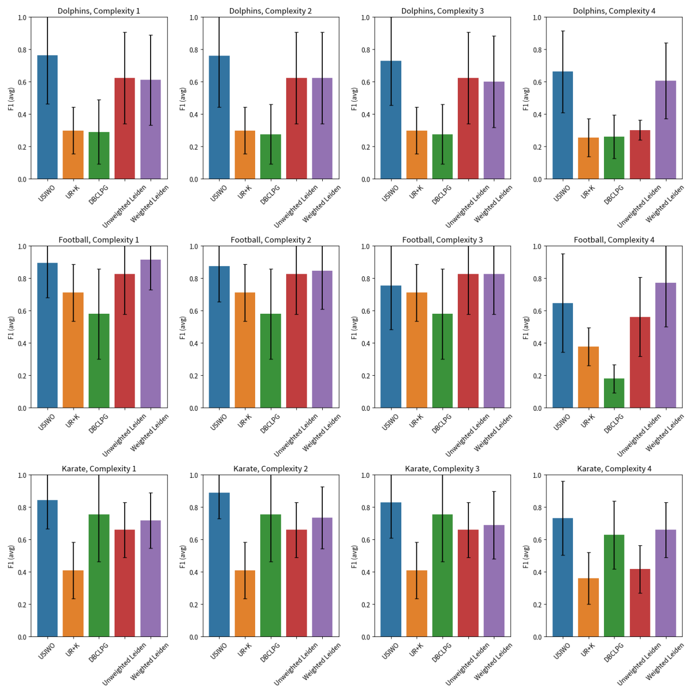


Fig. 1. Community search: F_1 average scores computed over all nodes used as query nodes on real world networks

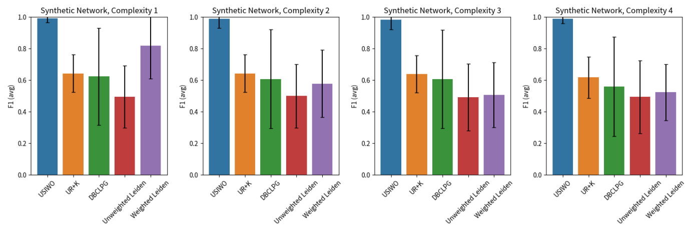


Fig. 2. Community search: F_1 average scores computed over 100 nodes used as query nodes on a synthetic network

scores for each algorithm on each uncertain graph, and their corresponding standard deviations. The average results over existing graphs are shown in Table I. In terms of execution time, all contenders are equivalent not withstanding the fact that Leiden does not consider possible worlds. *USIWO* requires three times the time needed for UR+K for a given query node and the execution time depends on the size of the considered community and not the network, typically not exceeding 1.5 seconds, when ran on a commodity laptop.

For the real-world networks, the *USIWO* algorithm outperforms other algorithms in terms of Recall, and F_1 measure on average and it is second in terms of Precision as shown in Table I. However, in specific cases such as the Football network at complexity level 4, the Weighted Leiden algorithm surpasses *USIWO* with an F_1 score of approximately 0.77, compared to *USIWO*'s score of 0.65. Despite this, *USIWO* shows its strength in consistently achieving high scores across a range of networks and complexity levels.

For the small-scale synthetic network, *USIWO* demonstrates outstanding performance, achieving an average precision of nearly 1.00, a recall of 0.98, and an F_1 score of 0.99. The low standard deviation for the F_1 score, 0.04, suggests that *USIWO*'s performance is stable and reliable across different query nodes. A scalability experiment is done on the 1M node synthetic network. On this network, *USIWO* averaged 1.605 seconds per node (for 100 searches) with precision, recall, and F1 scores of 1.000, 0.988, and 0.994, using only 200 MB of RAM. In comparison, UR+K took 19.03 seconds per node, achieving scores of 0.706, 0.555, and 0.598 in precision, recall, and F1, respectively, while consuming around 5 GB of RAM. Furthermore, Weighted and Unweighted Leiden, as well as DBCLPG, faced memory errors and could not operate on this network, mainly due to the way they handle the input graph.

Our results experimentally validate the fact that the strength values calculated using Definition 6 effectively capture a node's relative importance relative to its other connections, providing a perspective that is both consistent and reflective of broader connectivity patterns of the involved nodes.

TABLE I
PERFORMANCE ON REAL-WORLD AND SYNTHETIC NETWORKS

Algorithm	Precision (avg)	Recall (avg)	F1 (avg ± std)
Real-World networks			
DBCLPG	0.78	0.46	0.49 ± 0.23
UR+K	0.75	0.42	0.44 ± 0.16
USIWO	0.83	**0.80**	**0.78** ± 0.25
Unweighted Leiden	0.82	0.60	0.63 ± 0.21
Weighted Leiden	**0.92**	0.66	0.72 ± 0.23
Small-scale Synthetic Network			
DBCLPG	0.71	0.56	0.60 ± 0.31
UR+K	0.72	0.60	0.64 ± 0.12
USIWO	**1.00**	0.98	**0.99 ± 0.04**
Unweighted Leiden	0.36	1.00	0.50 ± 0.21
Weighted Leiden	0.48	1.00	0.61 ± 0.20

VI. CONCLUSION AND FUTURE WORK

Complex networks are ubiquitous and important in many applications. Many effective and efficient approaches have been designed to better understand the structure of these networks. Yet most target deterministic graphs and very little has been done for probabilistic graphs despite their increasing prevalence and importance. In this paper we present a community search approach to identify the community a node belongs to when dealing with networks with edges tagged with existential probabilities. We demonstrate that, on most dataset configurations, our approach *USIWO* outperforms others in terms of accuracy. *USIWO* is also local, not requiring the whole network, making it practical for large networks.

We aim to extend our research in several directions. First, we plan to explore the applicability of our algorithm, *USIWO*, to other types of uncertain graphs, such as weighted uncertain graphs and attributed uncertain graphs, as well as graphs with uncertainties on nodes in addition to edges. Second, while our approach is local and by definition only explores locally without loading the whole graph, we intend to further compare the efficiency of our approach to others when dealing with very large probabilistic networks. This may include developing more efficient heuristics and more effective implementations to handle the computational challenges posed by large-scale uncertain graphs. Third, we intend to investigate further the theoretical aspects of our approach, seeking to provide more rigorous proofs and analyses of its performance and properties. We believe these directions will significantly contribute to the field of uncertain graph analysis and community search.

REFERENCES

[1] G. Baltsou, K. Christopoulos, and K. Tsichlas, "Local community detection: A survey," *IEEE Access*, vol. 10, pp. 110 701–110 726, 2022.

[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, p. P10008, 2008.

[3] M. Ceccarello, C. Fantozzi, A. Pietracaprina, G. Pucci, and F. Vandin, "Clustering uncertain graphs," *VLDB*, p. 472–484, 2017.

[4] A. Clauset, "Finding local community structure in networks," *Physical Review E*, vol. 72, no. 2, p. 026132, Aug. 2005.

[5] J. Dahlin and P. Svenson, "A method for community detection in uncertain networks," in *EISIC*, 2011, pp. 155 – 162.

[6] S. Z. Gharaghooshi, O. R. Zaïane, C. Largeron, M. Zafarmand, and C. Liu, "Addressing the resolution limit and the field of view limit in community mining," in *IDA*, 2020, pp. 210–222.

[7] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *PNAS*, pp. 7821–7826, 2002.

[8] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.

[9] I. Gutiérrez, D. Gómez, J. Castro, and R. Espínola, "From fuzzy information to community detection: An approach to social networks analysis with soft information," *Mathematics*, vol. 10, no. 4348, 2022.

[10] Z. Halim and J. H. Khattak, "Density-based clustering of big probabilistic graphs," *Evolving systems*, vol. 10, no. 3, pp. 333–350, 2019.

[11] K. Han, F. Gui, X. Xiao, J. Tang, Y. He, Z. Cao, and H. Huang, "Efficient and effective algorithms for clustering uncertain graphs," *VLDB*, vol. 12, no. 6, 2019.

[12] P. Hintsanen and H. Toivonen, "Finding reliable subgraphs from large probabilistic graphs," *DMKD*, pp. 3–23, 2008.

[13] J. Hu, R. Cheng, Z. Huang, Y. Fang, and S. Luo, "On embedding uncertain graphs," in *CIKM*, 2017, p. 157–166.

[14] X. Huang, L. V. S. Lakshmanan, and J. Xu, *Community Search Over Big Graphs*. Springer, 2019.

[15] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E*, 2008.

[16] Y. Li, X. Kong, C. Jia, and J. Li, "On clustering uncertain graphs with node attributes," in *ACML*, 2018.

[17] L. Liu, R. Jin, C. Aggarwal, and Y. Shen, "Reliable clustering on uncertain graphs," in *ICDM*, 2012.

[18] D. Lusseau, "The emergent properties of a dolphin social network," *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 2003.

[19] T. Martin, B. Ball, and M. E. J. Newman, "Structural inference for uncertain networks," *Phys. Rev. E*, 2016.

[20] A. Nilsson, "Implementing and evaluating clustering methods for large probabilistic graphs," Uppsala University, Sweden, Tech. Rep., 2021. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1605722/FULLTEXT01.pdf

[21] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios, "K-nearest neighbors in uncertain graphs," *Proc. VLDB Endow.*, vol. 3, no. 1–2, 2010.

[22] D. Suciu, "Probabilistic databases," in *Encyclopedia of Database Systems*, L. LIU and M. ÖZSU, Eds., 2009.

[23] V. A. Traag, L. Waltman, and N. J. van Eck, "From louvain to leiden: guaranteeing well-connected communities," *Scientific Reports*, 2019.

[24] S. Van Dongen, "Graph clustering via a discrete uncoupling process," *SIAM Journal on Matrix Analysis and Applications*, pp. 121–141, 2008.

[25] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, pp. 452–473, 1977.

[26] M. Zafarmand, Y. Talebirad, E. Austin, C. Largeron, and O. Zaïane, "Fast local community discovery relying on the strength of links," *Social Network Analysis and Mining*, vol. 13, 09 2023.

[27] C. Zhang and O. R. Zaiane, "Detecting Local Communities in Networks with Edge Uncertainty," in *ASONAM*, 2018, pp. 9–16.