**A Method for Finding the Highly Reliable Densest Subgraph from an Uncertain Weighted Graph**

Duong Quoc Anh Kiet, Thien Nguyen

Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam

Natural Language Processing and Knowledge Discovery Laboratory, Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam

Corresponding author

Email: [dqanhkiet1999@gmail.com](mailto:dqanhkiet1999@gmail.com)

[nguyenchithien@tdtu.edu.vn](mailto:nguyenchithien@tdtu.edu.vn)

# Abstract

Despite being widely used as a crucial modeling tool for a variety of data types, graphs sometimes contain uncertainties because of noise or incompleteness in the data that was collected. This makes mining uncertain subgraphs much more crucial, especially when dealing with dense and reliable graphs. Because of the decreased likelihood of an edge existing, dense subgraphs usually show low predicted density and dependability in uncertain circumstances. Additionally, existing models for uncertain graphs often ignore the graph weights component, a critical omission in real-world applications. To mine subgraphs with high density and dependability, this work presents a unique model for uncertain weighted graphs. We also introduce the idea of a subgraph with a high-pass filter for further in-depth examination. In addition, a greedy technique is designed to make it easier to find subgraphs in the context of an uncertain weighted graph. Finally, we show the efficacy of our method in locating dense, reliable subgraphs in weighted uncertain graphs using experimental validation on protein linked to a dataset.

***Keywords:*** uncertain graph, network reliability, excess average degree, graph mining, weighted graph, uncertain weighted graph, undirected weighted graph

# Introduction

Graph theory, the study of graph properties to represent complex systems, is a fundamental component of computer science. Researchers frequently concentrate on investigating large graphs to discover dense subgraphs within them. One of the most important fundamental concepts in this discipline is the concept of the densest subgraph, which is the subgraph with the highest edge density, or the largest number of edges that may exist in the subgraph [1]. Finding these kinds of dense subgraphs is important for many applications, such as social network research or biology, where identifying dense subgraph structures is important. Nevertheless, there are further complications when uncertainty is included in graph models. For undirected weighted graphs with uncertainty, this is particularly important.

This research focuses on graphs, whose edges do not have a specific direction, creating unique features and challenges. A graph in which the edges are probabilistic and uncertainty is introduced into the edge connections is called an uncertain graph. The notion of weighted uncertain graphs is advanced by assigning weights to edges, with uncertain weights representing simply the existence of an edge. Noise or incomplete data collection is one cause of this uncertainty. As noted by Rual et al., (2005), protein-protein interaction (PPI) networks [3] show that each protein bond can influence structure and biological function, and this uncertainty is evident. Furthermore, the most reliable graph is the graph with the highest probability of edge connections. This is done to balance the reliability and density of the subgraph so that important structures are maintained for different purposes.

A diagram of a tree

Description automatically generated

Figure 1 Protein-Protein Network (PPI)

The goal of this research is to provide an effective strategy for finding dense and dependable subgraphs in uncertain weighted graphs. Our technique takes into account both connection density and edge reliability, resulting in a balance between graph density and reliability. This is important when evaluating Protein-Protein Interaction (PPI) networks and applications that require searching for subgraphs containing relevant data since the graph's density and reliability can have an important impact on the outcomes of these applications. Our technique gives tools for dealing with complicated challenges related to uncertain weighted graphs, and we believe it will make a valuable contribution to the area of graph analysis, as existing methods are insufficient in this context.

# Problem statement

An important challenge in graph theory is to identify dense subgraphs in uncertain weighted graphs. Dense subgraphs with closely connected vertices often contain important information about the properties of the graph. Therefore, extracting dense subgraphs from large graphs is an essential task in graph research. However, advanced analytical techniques are required to derive dense and reliable subgraphs from uncertain environments, especially when each edge has a different probability of existence. Furthermore, equating edge weights often fails to capture the complexity of their interrelationships. Therefore, new methods must be used to incorporate weights, which are measures of the strength of relationships, into graph models. Dense subgraphs of proteins often indicate protein complexity and suggest how proteins interact to perform specific functions. For example, the Fanconi protein complex is involved in DNA repair [4]. Therefore, the use of dense subgraphs in uncertain weighted graphs is essential for identifying and predicting previously unknown protein complexes.

Goldberg (1984) found that the density of a graph with equals . The goal is to find a subset of vertices that has the highest density. [5] first described the expected density of uncertain graphs and then emphasized the problem of finding dense subgraphs in uncertain graphs. In uncertain graphs, edges and weights are probabilistic. This makes finding dense and reliable subgraphs challenging. [6] examined the reliability of uncertain graphs by examining the reliability of subgraphs. Jin et al., 2011 developed the idea of reliability in uncertain graphs and used sampling-based methods for finding high-reliability subgraphs in uncertain graphs. The subgraphs discovered in this way present some concerns. For example, uncertain subgraphs A and B have the same expected density of . Naturally, we cannot compare two graphs based just on expected density. The reliability of a subgraph is used to evaluate the reliability of an uncertain graph. Due to lower edge probabilities, graph B is less reliable than graph A in Figure 2. As a result, a subgraph's reliability is frequently prioritized over its density. [7] developed a beta subgraph method and an optimal subgraph algorithm for finding dense and reliable subgraphs, with positive outcomes for a variety of real-world uses. The present research attempts to deal with the challenge of balancing density and reliability, however, it ignores uncertain weights in graphs because it assumes equal weights.

A close-up of a diagram

Description automatically generated

Figure 2 Density and reliability of graph

The goal of this research is to provide an original method for finding dense and reliable subgraphs in uncertain weighted graphs. Our objective is to develop an approach that can handle both density and reliability while simultaneously providing a reliable and efficient tool for graph analysis in uncertain contexts. Building on known theoretical foundations [8] and recent developments [[6] [9] [7]], we believe that our new method will provide an effective instrument for graph analysis in current practical applications.

# Related works

Researchers are particularly interested in finding dense subgraphs, and they have concentrated on graph models and methods to find dense subgraphs in uncertain contexts. The probable existence of graph edges makes finding dense and reliable subgraphs more challenging. Goldberg's (1984) research on graph density definition, as well as Charikar, (2000) contributions on greedy approximation techniques, give fundamental details about graph density in certain contexts, [10] focused on identifying the densest k-connected subgraphs, demonstrating that research into dense subgraphs also extends to more structured types of graphs. Moreira et al. (2020) developed algorithms to analyze complex hierarchical structures within bipartite graphs, thereby broadening the field and offering perspectives on potential applications in social networks and biological studies, thus extending the scope of graph analysis in various domains. Furthermore, greedy algorithms based on maximal subgraph density, as presented in studies [12] and [13] , increase density by reducing the size of the subgraph, although this is NP-hard. [14] proposed a new dynamic method for identifying the densest sub-hypergraphs within a larger graph framework. Map-Reduce other methods have been used to investigate high-density subgraph issues; however, applying these concepts to unpredictability is far more difficult.

Uncertain graph research began with [9] and has now expanded into a large field of study. These research has presented a variety of methods and models, including Jin et al., (2011) on graph reliability, Z. Zou et al., (2010) on frequent pattern mining, and Kollios et al., (2013) on graph clustering. Particularly, the notion of the largest clique in uncertain graphs, developed in the study of [5], is important. A clique is a subgraph with the highest edge density, and the largest clique is known as a maximal clique, which is thought to be the densest region in the graph. However, finding maximum cliques is NP-hard, and many graphs do not contain cliques, therefore quasi-cliques were developed to handle the NP-hard challenge.

Applications of dense subgraph discovery extend beyond theory to practical applications, such as protein-protein interaction analysis (PPI) by Rual et al., (2005), or social networks and fraud detection [17], these enhancements has deepened our understanding of dense subgraphs within these complex structures, Saha et al. (2022) introduced a new perspective in the field of uncertain graph analysis by providing methods to identify the most likely densest subgraphs, a critical step in understanding complex networks such as social and brain networks. . Furthermore, the development of dense subgraph mining techniques in uncertain environments as proposed by Jin et al., (2011) articulates a new direction in understanding dense subgraphs. Furthermore, Lu et al., (2019) proposed a method to use the beta parameter as a threshold to find dense and reliable subgraphs. However, it assumed that the edges had equal weights, which was a common but incorrect assumption. The work in [19] laid the foundation for dense subgraph detection in uncertain and weighted graphs. Additionally, a study by Cheng et al. 2014 work on shortest-path algorithms also provided insights into uncertain weighted graphs and formed the basis for our research presentation. They provided an advanced research method to identify dense subgraphs in uncertainly weighted graphs. Our goal is to extend these techniques to investigate the interdependence between weights and probabilities that can affect the reliability and density of uncertain graphs. Our work therefore builds on this and extends the problem of finding dense and reliable subgraphs in uncertain weighted graphs.

A diagram of a network

Description automatically generated

Figure 3 Uncertain weighted graph

# Preliminaries and definitions

This section introduces certain notations and definitions. These mathematical formulae will be utilized throughout the article.

|  |  |
| --- | --- |
| Notion | Definition |
| **G** | Uncertain weighted graph |
|  | Set of vertices in the graph |
|  | Set of edges in the graph |
|  | The edge between and in the graph |
|  | Weight of edge E in the graph |
|  | Uncertain probability of the edge E having a weight equal to |
|  | Number of vertices of a graph |
|  | Number of edges of the graph |
|  | Maximum number of edges in the graph |
|  | The average weight of the graph |

Table 1 Notion and definition

**Definition 1** : (Uncertain Weighted Graph) An uncertain weighted graph is represented as a random variable , where is the set of all certain vertices, and the random variable set represents all edges, each being a random variable with a weight following a Bernoulli distribution, specifically or . Let be a set of vertices, and the subgraph is produced from the uncertain weighted graph where . In this study, we assume that edges are pairwise independent random variables.

For example: Figure 3 represents an uncertain weighted graph, with edge having a weight and a probability .

**Definition 2** : (Adjoint Weighted Graph) An adjoint weighted graph is defined by as the set of all certain vertices and as all edges. Each edge is a random variable with of all its edges equal to .

For example: In Figure 3, assume that all edge probabilities are and that edges are connected just by their weights.

**Definition 3** : (Adjoint Logarithmic Reliability) The logarithmic link reliability of an uncertain weighted graph G=(v, E), which is calculated from the adjoint reliability of [7], uses base-10 logarithms for efficient calculation and representation of the product of multiple floating point integers on a computer.

For example, The adjoint logarithmic reliability of the graph in Figure 3 is , calculated as .

**Definition 4**: (Average edge weighted probability) The average edge weighted probability of an uncertain weighted graph is calculated as

For example: Figure 3 shows the graph's average edge-weighted probability as .

**Definition 5** : (Standard Deviation of Edge Weight Probability) The standard deviation of edge weight probability for an uncertain weighted graph is:

For example: Figure 3 shows that the standard deviation of the graph's edge weight probability is

**Definition 6** : (Weighted Expected Edge Density) The weighted expected edge density of a subgraph produced from the uncertain weighted graph is defined as:

For example: Figure 3 shows the weighted expected edge density of the graph as .

**Theorem 1**: An uncertain weighted graph is represented as a random variable , with as the set of all certain vertices and as all edges. Each edge is a random variable with weight following a Bernoulli distribution, particularly or . Let be a set of vertices. The subgraph is produced from the uncertain weighted graph , where . The weighted expected edge density of **G**' will increase as its average edge-weighted probability increases.

**Proof** :

⟺

With the subgraph **G**' having a determined number of edges and vertices, the weighted expected edge density will increase along with the average edge-weighted probability of the graph.

# Propose Methods

We developed two fundamental approaches for finding dense and reliable subgraphs from uncertain weighted graphs: Greedy Uncertain Weighted Dense Subgraph (GreedyUWDS) and Greedy Bound Weighted Dense Subgraph (GreedyBWDS). Each method is specific to an individual aspect of the problem for finding dense and reliable subgraphs, and it is based on a variety of assumptions and evaluation criteria that satisfy the many objectives of graph analysis.

The GreedyUWDS method is designed to resolve edge existence uncertainty while also optimizing subgraph density. GreedyUWDS improves on the GreedyUDS method [7], but modifies the graph to account for unknown weights. GreedyUWDS uses weighted expected density parameters to find high-reliability dense regions in uncertain graphs, providing a deep understanding of the graph's structure and relationships. Similarly, the GreedyBWDS method is intended for finding dense and reliable subgraphs, but it also uses a 'b' parameter to control the subgraph analysis. According to the experimental results, this technique may be applied to a wide range of applications, with a focus on evaluating and selecting edges based on the probability of existence.

In addition to these improved approaches, we discuss a traditional method for finding dense and reliable subgraphs based on weighted expected density, which is similar to the GreedyUWDS algorithm. The brute force technique simply tests all potential possibilities in the graph and chooses the best subgraph based on weighted expected density. While brute force is a useful benchmark to evaluate the performance of complex algorithms such as GreedyUWDS and GreedyBWDS, it is inefficient for large graphs and should not be used for complicated graphs. GreedyUWDS and GreedyBWDS have made significant strides in solving the challenges of finding dense subgraphs in uncertain graph environments. Both techniques provide a better knowledge of the density and reliability of uncertain graphs, providing up new possibilities for future research and application.

## **Algorithm Brute Force**

In attempts to find dense subgraphs from uncertain weighted graphs, the traditional brute force technique is vital and significant. This technique analyzes every potential subset of vertices and edges in the graph and evaluates each subset using weighted expected density, which is a standard based on the overall weight of the edges and their probability of existence. While brute force achieves the best answer by completely considering all options, it has a significant computing complexity limitation. In large graphs, the number of subsets to consider quickly increases and renders runtime impossible. As a result, although being a vital benchmark approach, brute force is often used primarily in theory or as a comparison method in small-sized graph situations, and it is rarely recommended for large and complicated graphs due to performance limitations. As a result, the remaining two algorithms become methods for finding dense and reliable subgraphs in uncertain weighted graphs, with the GreedyBWDS algorithm being capable of controlling the subgraph search process.

## **Algorithm GreedyUWDS**

**Definition 7** : (Weighted Expected Degree) The weighted expected degree of a vertex of the subgraph produced by the graph is defined as

For example: In Figure 3, the weighted expected degree of vertex B is .

**Definition 8** : (Weighted Expected Density) The weighted expected density of the uncertain weighted subgraph produced by the graph is defined as:

For example: In Figure 3, the weighted expected density of the uncertain weighted graph is .

**Definition 9**: An uncertain weighted graph is defined by v as the set of vertices and as the set of edges, where each edge is a random variable with weight that follows a Bernoulli distribution, especially or . Let be a set of vertices. The subgraph is produced by the uncertain weighted graph , where . In this research, we assume that edges are pairwise independent random variables with , which is referred to as the densest uncertain weighted subgraph, or UWDS for short.

The GreedyUWDS method explained in this research is a modification of the notion of the minimal cut in [6] and is based on the Charikar algorithm. It has been modified to reflect the characteristics of uncertain weighted graphs. Algorithm 1 contains a pseudocode that describes the algorithm in full.

**Algorithm 1: GreedyUWDS**

Input: Uncertain Weighted Graph with and for each

Output: Vertex set of the subgraph found with the highest weighted expected density

1. Initialize best\_subgraph and best\_weighted\_expected\_density
2. For i = n, perform the following steps until i = 2:
3. Calculate the current weighted expected density of the vertex set
4. Update best\_subgraph and best\_weighted\_expected\_density if necessary
5. Calculate the weighted expected degree of each vertex and remove the vertex with the smallest weighted expected degree value from the graph
6. End for
7. Return best\_subgraph and best\_weighted\_expected\_density

Throughout the iteration phase, the method evaluates the weighted expected density of the current graph and reduces vertices with the lowest weighted expected degree, remaining just two vertices in the graph. The method produces a dense and reliable subgraph with the maximum weighted expected density. The algorithm has a temporal complexity of , where is the number of vertices and is the number of edges in the uncertain weighted graph.

## **Algorithm GreedyBWDS**

**Definition 10** : (Excess Degree) For a vertex and a set of edges of an uncertain weighted graph , and a value , the excess degree of vertex is defined as the product of the probability of edge weight minus the value b and the weight of edges connected to that vertex.

For example: In Figure 3, assuming b = 0.5, the excess degree of vertex B is .

**Definition 11** : (Excess Average Degree). For an uncertain weighted graph with a value and as a subgraph produced by a subset , the excess average degree of graph is defined as follows:

For example: Assuming b = 0.5, the excess average degree of the graph is:

**Definition 12** : (High-Pass Filtered Subgraph) Given an uncertain weighted graph and a value , is a subgraph building from with subsets such that:

We refer to the subgraph as the High-pass Filtered Subgraph, and the value appears as a filtering threshold.

The GreedyBWDS method is similar to the GreedyUWDS algorithm but uses the parameter for controlling the finding process. It also uses the excess average degree to find dense and reliable subgraphs, and it determines which vertices have the lowest excess degree and should be removed until only two vertices remain in the graph. The method produces the dense, reliable subgraph with the biggest excess average degree. To improve the method, we in addition utilize a min-heap to extract the vertex with the lowest excess degree. However, because Python's min-heap does not enable modifying previously added values, we use a simple method to mark and handle old and new vertex put to the heap. Details of the algorithm are provided in Algorithm 2.

The GreedyBWDS technique, which is central to this study, is based on the parameter, which provides an important limit for finding dense and reliable subgraphs within uncertain weighted graphs. The program effectively handles vertex evaluations by strategically using the excess average degree as an essential variable for subgraph evaluation. It utilizes a min-heap structure to methodically find and remove vertices with the lowest excess degree. This method is repeated iteratively, decreasing the graph's complexity until it reaches a possibly dense and reliable subgraph. This subgraph, defined by its high excess average degree, acts as the algorithm's final output, providing essential insights into the fundamental graph structure.

**Algorithm 2** **GreedyBWDS**

Input: Uncertain Weighted Graph with and for each , and a parameter value

Output: Vertex set of the subgraph with the highest excess average degree

1. Initialize best\_subgraph and best\_excess\_avg\_degree
2. Initialize min\_heap and vertex\_to\_marker
3. For each vertex v in the graph:
4. Calculate excess\_degree of v
5. Add (excess\_degree, v, current marker) to min\_heap
6. Update vertex\_to\_marker[v] = current marker
7. For i = n perform the following steps until i = 2:
8. Calculate the current excess\_avg\_degree of the vertex set
9. Update best\_subgraph and best\_excess\_avg\_degree if necessary
10. For len(minHeap) > 0:
11. Extract (excess\_degree, v\_to\_remove, marker) from min\_heap
12. If the marker for v\_to\_remove matches vertex\_to\_marker[v\_to\_remove]:
13. Remove v\_to\_remove from the vertex set and break the loop
14. End for
15. Collect all adjacent vertices of v\_to\_remove
16. Remove v\_to\_remove from the graph
17. Increase current marker
18. For each adjacent vertex in the graph:
19. If the vertex is in the vertex set:
20. Calculate new excess\_degree and add to min\_heap
21. Update vertex\_to\_marker
22. End for
23. End for
24. Return best\_subgraph and best\_excess\_avg\_degree

## **Algorithm Accuracy**

Suppose we have an uncertain weighted graph , and we find as the subgraph with the largest average excess, suggesting that has the best method.

**Theorem 2**: For an uncertain weighted graphwith any vertex , then .

**Proof** : Because is the best method subgraph, we know that , therefore

⟺

⟺

⟺

**Theorem 3**: Theorem 3: For an uncertain weighted graph and a parameter , is the high-pass filtered subgraph of , the best method, with the largest excess average degree . If is a subgraph near to the high-pass filtered subgraph of produced through the GreedyBWDS algorithm, showing an approximation technique with excess average degree , then.

The excess average degree of produced through the GreedyBWDS method is at least half that of the high-pass filtered subgraph , as proof by the approximation algorithm in [8].

# Implementation Details

This section focuses on the practical implementation of our proposed get conceptual model, with particular focus on the development of two crucial algorithms: greedy\_uwds and greedy\_bwds. These algorithms help find dense and reliable subgraphs within uncertain weighted graphs.

Graph representation: Our method begins with a graph structure declared as “dicttionary[string, list[tuple[string, float]]”. This dictionary converts each vertex into a list of tuples, each containing an adjacent vertex, the edge's weight, and probability. The graph is constructed with the add\_vertex and add\_edge methods. Function add\_vertex adds a new vertex to the graph, whereas add\_edge creates an edge with a specified weight and probability between two vertices.

greedy\_uds: Designed for maximizing weighted expected density, this algorithm iteratively removes the vertex with the lowest weighted expected degree, recalculating the density at each iteration. It provides an effective approach for solving edge existence uncertainty while focusing on the most densely connected regions. The function results in a set of the best subgraphs found by mining, together based on their weighted expected density.

greedy\_bwds: This method is distinguished because it uses a bound parameter as a control mechanism. It evaluates the excess degree of vertices and iteratively removes those with the lowest value, focusing on subgraphs with an optimal balance of density and reliability. A notable feature of this method is the implementation of a min-heap structure, enhancing efficiency in identifying and processing vertices during the algorithm's execution. The function results in a set of the best subgraphs found by mining, together based on their excess average degree.

brute\_force: Our brute force technique goes for all possible subgraphs to find the one with the highest weighted expected edge density. This accurate technique tests every possible combination of vertices and edges, determining the weighted expected density for each subgraph. The function results in a set of the best subgraphs found by mining, together based on their weighted expected density.

The methods' computational design depends strongly on the function's get\_excess\_degree, get\_excess\_average\_degree, get\_weighted\_expected\_density, and get\_weighted\_expected\_degree.

* get\_excess\_degree: Calculates a vertex's excess degree, which is an important metric for choosing subgraphs with higher density and reliability in which there are of uncertain weights.
* get\_excess\_average\_degree: Calculates a subgraph's excess average degree, which allows for the optimum balance of density and reliability.
* get\_weighted\_expected\_density: Calculates a subgraph's expected density, which is important to assess its overall connectedness and relevance.
* get\_weighted\_expected\_degree: Calculates the weighted expected degree of a vertex, which is necessary for determining the importance of each vertex within the graph's structure.

The remove\_vertex method in the program is crucial for modifying the graph's structure. It effectively removes the chosen vertex and associated edges from the graph, which is critical for reducing finding it for dense subgraphs. This function iterates through the adjacency list, removing all edges associated to the removed vertex, maintaining the graph's integrity and accuracy.

Furthermore, evaluation metrics such as adjoint\_logarithmic\_reliability, average\_edge\_weighted\_probability, weighted\_ expected\_edge\_density, and std\_edge\_weight\_probability provide a thorough examination of the subgraphs' atrribute. These measures evaluate not just the subgraphs' reliability and density, but also the graph's overall quality of structure.

Create a function identified print\_summarize\_graph that displays details about the graph, such as its vertices, edges, and key evaluation metrics. This function is critical for demonstrating a full overview of the graph's structure and evaluating results, allowing for an understanding of the graph's attributes as well as the effectiveness of the algorithms applied.

Following the development of our algorithms, we added a class designed for testing on a protein dataset. To make this easier, a function has been built for reading data files and producing a graph including properties related to protein interactions. This phase is critical for using our algorithms in a practical, real-world setting. After defining a structure with protein-related properties, we do an extensive analysis on the protein data. Using our get\_uwds\_algorithm or get\_bwds\_algorithm methods provides complete results that demonstrate the methods' effectiveness. These functions not only return the best subgraph found throughout the mining process, but also provide a complete evaluation of parameters such as execution time. This systematic approach enables a clear and exact evaluation of the algorithms' performance in processing complex datasets, such as protein interactions. The results of these functions, particularly the highlighted best subgraph and related metrics, provide useful information, demonstrating the applicability and efficiency of our computational methods in real-world analysis of information.

A diagram of a company

Description automatically generated

Figure 4 Flow-chart for mining highly reliable densest subgraph

# Experiment Setup

The protein data used in this section comes from the STRING-DB database (http://string-db.org), namely protein. However, because the data lacks weighted edges, we will utilize the data's attribute columns to weight protein connections. The first two columns in the data show protein vertices and their connections, with "combine\_scored" denoting probabilities, which represent the uncertain weight of protein a connection. The remaining data columns will provide weights for relevant edges, with zeros replaced by a constant of calculated from protein biology data. The algorithms' run times are computed using the average of their separate run times. Decimal values will be rounded to the third decimal place, and the default parameter for comparing the GreedyBWDS method is . Because the same protein dataset is used with varying weights given to edges by different attribute columns, the number of vertices is always and the number of edges is .

To find subgraphs that have excellent density, reliability, and efficiency, we compare the GreedyUWDS and GreedyBWDS algorithms' performance. We then examine the performance of various values in the GreedyBWDS algorithm to show that this parameter may be changed according to various applications. This research used all algorithms and was run on Python on a test PC equipped with an AMD Ryzen H GHz CPU, GB of RAM, and Windows Home Single Language.

|  |  |  |
| --- | --- | --- |
| Feature | Average edge-weighted probability | The standard deviation of edge weight probability |
| neighborhood | 36.721 | 219.572 |
| fusion | 14.684 | 137.995 |
| cooccurrence | 53.865 | 184.996 |
| coexpression | 56.314 | 362.262 |
| experimental | 55.889 | 355.513 |
| database | 26.28 | 333.74 |
| textmining | 35.231 | 91.835 |

Table 2 Overview of the Dataset Used for Experimentation

**Results and discussion**

In this part, we will compare and evaluate the results of experiments using a protein dataset. This comparison is divided into two parts: first, we compare the GreedyUWDS and GreedyBWDS algorithms; and second, we look at the GreedyBWDS algorithm's results when different values are used.

|  |  |  |
| --- | --- | --- |
| Feature | Execution time | |
| UWDS | BWDS |
| neighborhood | 133.731 | 55.951 |
| fusion | 155.252 | 55.381 |
| cooccurrence | 158.06 | 68.385 |
| coexpression | 160.907 | 67.853 |
| experimental | 160.892 | 67.039 |
| database | 157.174 | 69.746 |
| text mining | 165.555 | 62.023 |

Table 3 Comparison of execution time between the two algorithms

Figure 5 Graph comparing the execution time of UWDS and BWDS

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Feature | Weighted Expected Edge Density | | Average Edge-Weighted Probability | | The Standard Deviation of Edge Weight Probability | | Adjoint Logarithmic Reliability | |
| UWDS | BWDS | UWDS | BWDS | UWDS | BWDS | UWDS | BWDS |
| neighborhood | 388.321 | 616.659 | 438.329 | 575.923 | 215.412 | 176.195 | -209.163 | -8.518 |
| fusion | 35.123 | 640.159 | 32.665 | 654.708 | 279.519 | 143.502 | -4822.897 | -1.874 |
| cooccurrence | 508.576 | 1016.148 | 278.939 | 508.074 | 175.252 | 162.742 | -2298.398 | -19.189 |
| coexpression | 1401.044 | 1547.738 | 714.772 | 776.682 | 204.707 | 167.679 | -447.093 | -174.462 |
| experimental | 1445.787 | 1548.561 | 749.406 | 790.524 | 134.945 | 107.097 | -350.654 | -218.437 |
| database | 1350.766 | 1341.696 | 695.726 | 670.848 | 70.619 | 102.233 | -3.826 | -0.648 |
| textmining | 339.654 | 401.997 | 176.414 | 201.095 | 72.708 | 52.356 | -603.928 | -77.534 |

Table 4 Comparison of Results of GreedyUWDS and GreedyBWDS Algorithms

Figure 6 Graph comparing the weighted expected edge density of UWDS and BWDS

**Evaluation Indicators**

We discover many important parameters to evaluate algorithm performance on weighted uncertain graph data. The weighted expected edge density is the first metric, which helps understand the density of the structure within the subgraph identified by the algorithm. The next metric is adjoint logarithmic reliability, which measures the reliability of relationships throughout the subgraph. In addition, we look at the average edge-weight probability and the standard deviation of edge weight probability to get a more comprehensive and in-depth understanding of the results in subgraphs.

The experimental results show a positive correlation between the specified evaluation parameters: as the weighted expected edge density increases, so does the average edge weight probability, the standard deviation decreases, and the adjoint logarithmic reliability, resulting in a subgraph that is both highly reliable and dense.

## **Model Comparison**

When the GreedyUWDS and GreedyBWDS algorithms are applied to the same dataset with the parameter set to the default value of , GreedyBWDS outperforms GreedyUWDS in two important standards: weighted expected edge density and adjoint logarithmic reliability. GreedyBWDS, in particular, results in subgraphs with more density while simultaneously ensuring better reliability, proving the ability to find dense and reliable subgraphs from uncertain data. Besides these evaluation measures, we evaluated the execution times of both methods. Table 3 shows that GreedyBWDS has a significantly faster execution time than GreedyUWDS, nearly times faster. This shows that GreedyBWDS is not only excellent at finding dense, reliable subgraphs, but it also performs much more quickly than GreedyUWDS.

Detailed experimental analysis resulted in:

1. The GreedyBWDS algorithm outperforms GreedyUWDS in terms of density. Table 3 shows that the average weighted expected edge density for GreedyUWDS is , whereas the average for GreedyBWDS is . As a result, GreedyBWDS outperforms GreedyUWDS in finding dense subgraphs due to its higher density.
2. GreedyBWDS outperforms GreedyUWDS in terms of adjoint logarithmic reliability. Table 3 shows that GreedyBWDS outperforms GreedyUWDS in terms of adjoint logarithmic reliability over the entire dataset.

## **Parameter Selection**

In this section, we use two features, namely "experimental" and "text mining," to test and evaluate how different parameter values impact the results of discovering dense and dependable subgraphs. The experimental procedure was developed to offer a thorough knowledge of how different variables might improve search performance and influence the quality of produced subgraphs. This not only provides a better understanding of the parameter's impact, but it also provides the foundation to determine if the GreedyBWDS algorithm is the best method and how parameter setting may vary across different real-life scenarios.

We gained valuable insights into the GreedyBWDS algorithm by experimenting with different values of the parameter . In particular, the parameter is a useful tool for modifying and controlling the algorithm's output. For example, when the value is set to , the GreedyBWDS outputs are extremely similar to those produced by the GreedyUWDS method, indicating a level of density but a low degree of reliability. However, raising the value of results in a significant improvement not only in the density of the subgraph but also in the average edge weight probability and reliability. Furthermore, execution time while utilizing different values is essential; as the value of grows, the algorithm's execution time becomes faster. This demonstrates GreedyBWDS' ability to adapt to different demands in real-world applications, allowing for flexible algorithm development based on the unique requirements of each dataset and research purpose.

Although this work provides deep knowledge and a new method for identifying dense and reliable subgraphs from weighted uncertain graphs, there are still limitations that need to be addressed. First, although our algorithm is effective on protein assay datasets, its ability to handle other types of data has not yet been confirmed. Although our study mainly focuses on undirected uncertain weighted graphs, it may not be directly applicable to other types of graphs or directed graphs, which limits the completeness of our method.

Figure 7 Graph showing weighted expected edge density with different parameters

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Feature | b | Number of Edges | Weighted Expected Edge Density | Average Edge-Weighted Probability | The Standard Deviation of Edge Weight Probability | Adjoint Logarithmic Reliability | Execution Time |
| experimental | 0.1 | 2972 | 1445.787 | 749.406 | 134.945 | -350.654 | 120.982 |
| 0.2 | 2906 | 1463.720 | 756.289 | 128.665 | -320.305 | 142.639 |
| 0.3 | 2906 | 1463.720 | 756.289 | 128.665 | -320.305 | 95.102 |
| 0.4 | 2906 | 1463.720 | 756.289 | 128.665 | -320.305 | 77.279 |
| 0.5 | 2771 | 1498.980 | 770.858 | 119.400 | -283.647 | 64.667 |
| 0.6 | 2574 | 1548.561 | 790.524 | 107.097 | -218.437 | 60.415 |
| 0.8 | 1950 | 1692.501 | 847.553 | 77.576 | -64.973 | 55.393 |
| text mining | 0.1 | 2935 | 356.219 | 182.236 | 69.523 | -446.475 | 137.015 |
| 0.2 | 2613 | 377.517 | 189.842 | 65.475 | -254.904 | 134.338 |
| 0.3 | 2477 | 387.107 | 194.179 | 63.047 | -192.838 | 120.862 |
| 0.4 | 2409 | 391.308 | 196.141 | 61.961 | -170.589 | 101.346 |
| 0.5 | 2340 | 395.420 | 198.217 | 60.821 | -150.769 | 82.937 |
| 0.6 | 2079 | 401.997 | 201.095 | 52.356 | -77.534 | 72.327 |
| 0.8 | 1595 | 424.933 | 212.600 | 44.862 | -21.541 | 68.831 |

Table 5 Comparison of different parameters

Figure 8 Compare execution time with different parameters b

# Conclusion

In this paper, we explore a methodology to extract dense and reliable subgraphs from uncertain weighted graphs by developing a new greedy algorithm. The algorithm uses parameter as a threshold tuning tool to determine the optimal subgraph that satisfies the condition. Through practical experiments, we show that the subgraphs identified by this algorithm outperform previous models in terms of both density and reliability. This development opens up a wide range of applications in the analysis of systems with complex graph structures and represents a significant advance in the study and application of uncertain weighted graphs. In future research, we plan to extend the application of this algorithm to different datasets and understand its applicability in more areas. Additionally, we plan to develop a method to identify multiple subgraphs from large graphs with high density and reliability.

# References

[1] R. Dondi, M. M. Hosseinzadeh, G. Mauri, and I. Zoppis, “Top-k overlapping densest subgraphs: approximation algorithms and computational complexity,” *J Comb Optim*, vol. 41, no. 1, 2021, doi: 10.1007/s10878-020-00664-3.

[2] J. F. Rual *et al.*, “Towards a proteome-scale map of the human protein-protein interaction network,” *Nature*, vol. 437, no. 7062, 2005, doi: 10.1038/nature04209.

[3] D. Szklarczyk *et al.*, “STRING v10: Protein-protein interaction networks, integrated over the tree of life,” *Nucleic Acids Res*, vol. 43, no. D1, 2015, doi: 10.1093/nar/gku1003.

[4] W. Wang, "Emergence of a DNA-damage response network consisting of Fanconi anemia and BRCA proteins," *Nature Reviews Genetics*, vol. 8, no. 10. 2007. doi: 10.1038/nrg2159.

[5] Z. N. Zou and R. Zhu, “Mining top-k maximal cliques from large uncertain graphs,” *Jisuanji Xuebao/Chinese Journal of Computers*, vol. 36, no. 10, 2013, doi: 10.3724/SP.J.1016.2013.02146.

[6] R. Jin, L. Liu, and C. C. Aggarwal, “Discovering highly reliable subgraphs in uncertain graphs,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011. doi: 10.1145/2020408.2020569.

[7] Y. Lu, R. Huang, and D. Huang, “Mining highly reliable dense subgraphs from uncertain graphs,” *KSII Transactions on Internet and Information Systems*, vol. 13, no. 6, 2019, doi: 10.3837/tiis.2019.06.012.

[8] M. Charikar, “Greedy approximation algorithms for finding dense components in a graph,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2000. doi: 10.1007/3-540-44436-x\_10.

[9] X. Gao and Y. Gao, “CONnectedness index of uncertain graph,” *International Journal of Uncertainty, Fuzziness and Knowldege-Based Systems*, vol. 21, no. 1, 2013, doi: 10.1142/S0218488513500074.

[10] F. Bonchi, D. García-Soriano, A. Miyauchi, and C. E. Tsourakakis, “Finding densest k-connected subgraphs,” *Discrete Appl Math (1979)*, vol. 305, 2021, doi: 10.1016/j.dam.2021.08.032.

[11] E. Moreira, G. Oliveira Campos, and W. Meira Jr., “Weighted Linking Decomposition: Mining Denser and More Compact Hierarchies for Bipartite Graphs,” *Journal of Information and Data Management*, vol. 11, no. 1, 2020, doi: 10.5753/jidm.2020.2031.

[12] S. Khuller and B. Saha, “On finding dense subgraphs,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009. doi: 10.1007/978-3-642-02927-1\_50.

[13] R. Andersen and K. Chellapilla, “Finding dense subgraphs with size bounds,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009. doi: 10.1007/978-3-540-95995-3\_3.

[14] S. K. Bera, S. Bhattacharya, J. Choudhari, and P. Ghosh, “A New Dynamic Algorithm for Densest Subhypergraphs,” in *WWW 2022 - Proceedings of the ACM Web Conference 2022*, 2022. doi: 10.1145/3485447.3512158.

[15] Z. Zou, J. Li, H. Gao, and S. Zhang, “Mining frequent subgraph patterns from uncertain graph data,” *IEEE Trans Knowl Data Eng*, vol. 22, no. 9, 2010, doi: 10.1109/TKDE.2010.80.

[16] G. Kollios, M. Potamias, and E. Terzi, “Clustering large probabilistic graphs,” *IEEE Trans Knowl Data Eng*, vol. 25, no. 2, 2013, doi: 10.1109/TKDE.2011.243.

[17] C. Ma, Y. Fang, R. Cheng, L. V. S. Lakshmanan, W. Zhang, and X. Lin, “On Directed Densest Subgraph Discovery,” *ACM Transactions on Database Systems*, vol. 46, no. 4, 2021, doi: 10.1145/3483940.

[18] A. Saha, X. Ke, A. Khan, and C. Long, “Most Probable Densest Subgraphs,” in *Proceedings - International Conference on Data Engineering*, 2023. doi: 10.1109/ICDE55515.2023.00115.

[19] A. Miyauchi and A. Takeda, “Robust Densest Subgraph Discovery,” in *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2018. doi: 10.1109/ICDM.2018.00157.

[20] Y. Cheng, Y. Yuan, G. Wang, B. Qiao, and Z. Wang, “Efficient sampling methods for shortest path query over uncertain graphs,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014. doi: 10.1007/978-3-319-05813-9\_9.