Most Probable Maximum Weighted Butterfly Search

Yu Shao [1], Peng Cheng [1], Longbin Lai [2], Long Yuan [3], Wangze Ni [4*], Xuemin Lin [5]
[1]*East China Normal University, Shanghai, China;* [2]*Alibaba Group, Hangzhou, China;*
[3]*Nanjing University of Science and Technology, Nanjing, China;* [4]*Zhejiang University, Hangzhou, China;*
[5]*Shanghai Jiaotong University, Shanghai, China*
yushao@stu.ecnu.edu.cn; pcheng@sei.ecnu.edu.cn; longbin.lailb@alibaba-inc.com; longyuan@njust.edu.cn
niwangze@zju.edu.cn; xuemin.lin@gmail.com

***Abstract*—Uncertain butterflies are fundamental and popular graphlet motifs within uncertain bipartite networks, serving as a crucial metric in structural analysis. Despite extensive research have studied butterflies sufficiently on deterministic networks, few of works explore uncertain butterflies. In this paper, we introduce the Most Probable Maximum Weighted Butterfly (MPMB), which holds the highest probability of becoming a maximum weighted butterfly on an uncertain bipartite network. Proved that searching MPMBs is NP-Hard, we then proposed two sampling-based methods, namely Ordering Sampling (OS), and Ordering-Listing Sampling (OLS). The OS method is suitable for single-trial sampling, while the OLS method is optimized for multiple trials, which first finds candidate butterflies in rough before searching MPMBs. Our experimental results indicate that our basic method (OS) performs 1000x faster than the baseline and the optimized method (OLS) achieves another 180x speedup.**

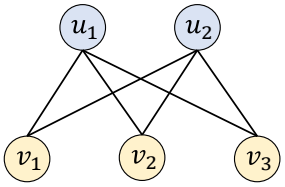
I. INTRODUCTION

Uncertain networks are graphs where the existence of each edge is a probability event (i.e., uncertainty) [1], [2], [3]. Recently, uncertainty connections have gained great interest in modeling real-world relations, such as connection reliability in routing networks [4], [5], [6], traffic awareness in road networks [7], [8], [9], and confidence in recommendation networks [10]. Recent studies have extended some traditional graph algorithms on deterministic networks to uncertain networks, e.g., shortest path query [11], [12], maximal clique search [13], [14], and nearest neighbor search [15], [16].

As a widely studied motif in graph researches, *butterfly* is a 4-cycle subgraph (also known as rectangle [17] or (2,2)-biclique [18]) on a bipartite network. Bipartite networks are useful in many applications, e.g., in brain networks [19], [20] and task matching networks [21], [22], [23], vertices represent left-right regions and worker-task entities, respectively. If edges are weighted, a butterfly with the largest weight is called the *maximum weighted butterfly*, representing a strong correlation between two pairs of vertices.

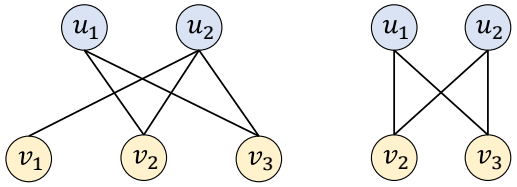
For example, Figure 1(a) shows an uncertain bipartite networks with two partitions $\{u_1, u_2\}$ and $\{v_1, v_2, v_3\}$. Each edge is associated with a specific weight and probability. Figure 1(b) is a possible world of the uncertain bipartite network and a butterfly on the possible world. The probability of this possible world is $(1-0.5)*0.6*0.8*0.3*0.4*0.7 = 0.02016$. The butterfly with four vertices (u_1, u_2, v_2, v_3) has weight 7.

*Wangze Ni is also with The State Key Laboratory of Blockchain and Data Security; Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security.



Edge	*Weight*	*Probability*
(u_1, v_1)	2	0.5
(u_1, v_2)	2	0.6
(u_1, v_3)	1	0.8
(u_2, v_1)	3	0.3
(u_2, v_2)	3	0.4
(u_2, v_3)	1	0.7

(a) An uncertain bipartite network with edge weights and probabilities.



(b) A possible world with probability 0.02016 and a butterfly with weight 7.

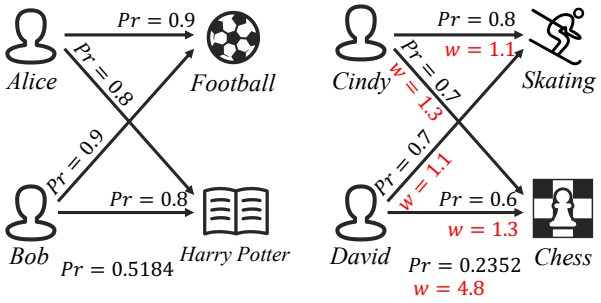
Fig. 1: An example of an uncertain bipartite network, a possible world, and a butterfly. A possible world is a sample of the uncertain network.

Weight and probability are two fundamental indicators of the relationship between entities. The weight of a butterfly reflects the relationship strength on the whole network, while the high probability indicates the existence possibility of the relationship. Intuitively, we want to find a butterfly that owns both a relatively high weight and probability.

In this paper, we consider these two indicators and study to find the Most Probable Maximum Weighted Butterfly (MPMB) that has the highest accumulated probability to be the maximum weighted butterfly among all the possible worlds on an uncertain network.

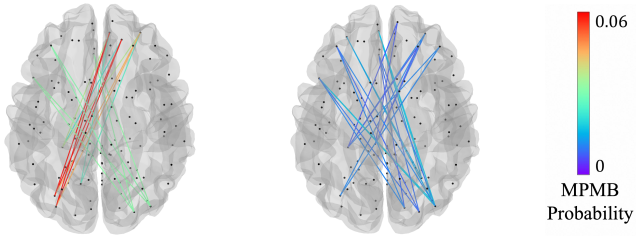
Diversity of MPMB. MPMB can reveal various important regions in a bipartite network. Suppose a dense region contains multiple butterflies, MPMB only counts one butterfly in each possible world. Instead of mining massive but similar butterflies [24], MPMB has the flexibility to return a suitable number of butterflies for the scattered visualization. We show the importance and usefulness of MPMBs with the following use cases.

Use case 1: Recommendation Systems. A user-item network is a bi-partite network where two vertex sets are *users* and *items*, respectively, and edges indicate behaviors such as *liking* and *buying* [25]. The User-based Collaborative Filtering (UserCF) technique [26], [27] is one of the most



(a) Most Probable Butterfly (b) Most Probable Maximum Weighted Butterfly

Fig. 2: Same interests can be used for recommendation. To avoid the impact of hot items, MPMB adds a weight on each edge to improve the quality of recommendations.



(a) MPMBs in TC. (b) MPMBs in ASD.

Fig. 3: MPMBs results in the Typical Controls (TC) group and the Autism Spectrum Disorder (ASD) group.

popular recommending methods, which assumes that similar users have similar likes/dislikes. Since a recommendation can be modeled as a biclique [28], [29], MPMBs represent the expected most valuable recommendations.

The UserCF often produces similar recommendations. For instance in Figure 2(a), Alice and Bob both like football and Harry Potter, forming a butterfly pattern with a high probability $\Pr = 0.5184$. However, there may be millions of other users also interested in these two hot items, which is a common phenomenon and worthless to recommend.

Instead, two users who are both interested in the unpopular items have a more obvious similarity. According to this idea, some optimized UserCF [30], [31] add a reward weight on the edge connected to the cold items. Take Figure 2(b) as an example, skating and chess are relatively unpopular. We assign higher weights to these edges. This butterfly has a relatively lower probability ($\Pr = 0.2352$) but a higher weight ($w = 4.8$), enhancing the diversity of recommendations [32].

Use case 2: Brain Network. A brain can be modeled as an uncertain network, where vertices are different *Regions of Interest* (ROIs) and edges are region connections. Edge weight and probability represent the distance and the correlation between two ROIs, respectively. We use *Automated Anatomical Labeling* (AAL) atlas [33] to locate ROIs and *Autism Brain Imaging Data Exchange* (ABIDE) [34] dataset to compute ROI-pairs probabilities. For hemisphere-crossing activities, edges between the left and right hemispheres are extracted to form a bipartite uncertain network. The dataset has the Typical Controls (TC) group and the Autism Spectrum Disorder (ASD) group. People in the TC group have more active connections between far regions, while ASD patients are lacking in long connections.

We calculate the top-10 MPMBs on the datasets in a TC brain and an ASD brain, as shown in Figure 3(a) and Figure 3(b). These butterflies are concentrated in four clusters, indicating more strong activities among these regions. Moreover, the color represents activation intensity, which is on average twice as high in TC compared to ASD, since patients generally have weak connections between long regions.

MPMB is an effective measurement in uncertain bipartite network analysis, but hard to compute directly. *The problem is that given the exponential possible worlds, we cannot afford the computation cost to enumerate all of them and find the butterfly with the maximum weight and the highest existing possibility.* Simply enumerating an exponential number of possible worlds and then finding the one with the maximum weight and the highest existing possibility is intractable and thus unacceptable. The most practical approach up to now is based on sampling, such as the Monte-Carlo Sampling [35] and Karp-Luby Sampling [36], evaluating with finite and little trials to approximate the exact result. Despite these sampling-based methods are already applied in a wide range of problems, there are few related optimizations towards the sampling process. We optimize existing solutions in the entire workload on calculating MPMBs, including the sampling process and computing process. Our contributions are as follows:

- We define the MPMB problem and prove its hardness in Section III. A baseline method MC-VP is introduced in Section IV.
- In Section V, we propose a method called Ordering Sampling (OS) with some ordering optimization based on MC-VP, which is efficient for single-trial sampling.
- In Section VI, we propose Ordering-Listing Sampling (OLS), which uses OS to list some candidate butterflies first and then computes probabilities. We also compare our optimized sampling algorithm to the popular Karp-Luby algorithm with some theoretical analysis.
- In Section VII, we extend the definition of the MPMB problem and claim that our methods can generate multiple MPMBs, called the top-k MPMBs.
- We implement our methods and report experimental results in Section VIII.

II. Related Work

Threshold-based methods. Threshold-based methods aim to mine all instances based on a given pattern and probability threshold, since an instance with a low probability is considered meaningless. Depending on the density of instances, these methods can either enumerate all instances [37], [38], [39], [40], or simply count the number [41], [42]. One general framework is to list all instances on the backbone network (i.e., a possible world containing all edges) first, and then compute the probability of each instance [43], [44]. Therefore, strategies used on deterministic networks can also be applied to the

backbone network. To optimize this process, many studies approximate the probability of an instance during the initial listing process, discarding instances with low probabilities early on [42], [40].

Distribution-based methods. Rather than merely counting the numbers based on a single threshold, distribution-based methods aim to count instances across all possible worlds, thereby generating a distribution of count numbers. This approach is more meaningful as it operates in real-world scenarios and considers the correlations between instances. However, it is intractable to enumerate all possible worlds. Recent works focus on sampling methods for estimating the mean [45], the variance [46], and the probability mass function [44].

Probable-based methods. Distribution-based methods can be thought of as *counting over all possible worlds*, while probability-based methods are akin to *enumerating over all possible worlds*. These methods seek the instance with the specific target (e.g., the shortest path [12] or the densest subgraph [47]) in each possible world and aggregate their probabilities. Similar to distributed-based methods, it's impractical to enumerate all possible worlds. Instead, most implementations use a sampling process to select potential worlds based on probability. The main advantage of probability-based methods is reporting real instances with high probabilities.

As both distribution-based and probability-based methods utilize a sampling process to generate possible scenarios, it's imperative to ensure the sampling efficiency and accuracy.

Monte-Carlo Sampling [35] is a fundamental and well-known sampling method. It uses a limited number of repeated random trials to approximate real distribution. For uncertain networks, Monte-Carlo sampling can generate a series of limited possible words, which can be seen as a close estimation of the precise result [44], [47]. *Karp-Luby Sampling* [36], [48] is a Monte Carlo sampling variant specifically designed for sampling the union of multiple variables, particularly when the probability is small. The main idea of Karp-Luby sampling involves applying reject sampling over the union area and discarding repeated parts. Regardless of how small the union probability might be, Karp-Luby sampling directly samples within the union area, thus achieving higher accuracy [12], [49].

Our methods. Our proposed methods follow some probable-based methods that apply both Monte-Carlo and Karp-Luby sampling. Instead of directly comparing the time complexity using a fixed trial number, we analysis the approximation guarantees and compute trial number lower bound. Therefore, we confirm that all methods have the same accuracy and make the comparison more reasonable.

III. PRELIMINARIES

This section gives the formal definition and the hardness proof of our problem. Some key notations are presented in Table I.

TABLE I: Notations.

Notation	Definition
G	uncertain bipartite network
H	backbone graph
W_i	possible world
B	butterfly
$Pr[\cdot]$	probability
$P(B)$	the probability of B being maximum
S_{MB}	maximum butterflies set
C_{MB}	maximum butterflies candidate set
$\angle$	angle
$A(\cdot)$	angle set
N	trial number

A. Problem Definition

Definition 1. (Uncertain Bipartite Weighted Network) An uncertain bipartite network is a graph $G = (V = (L, R), E, p, w)$, where V is the vertex set consisting of two disjoint parts L, R. $L \cap R = \varnothing$, $E \subseteq L \times R$ is the edge set, and $p : E \to [0, 1]$, $w : E \to \mathbb{R}$ maps each edge to a real number as the probability and weight, respectively.

We also define the backbone graph of G as $H = (V = (L, R), E, w)$, which is a deterministic network with the same structure.

Definition 2. (Possible World) A possible world $W_i = (V = (L, R), E_i, w)$ is a subgraph of H by sampling each edge $e \in E$ with probability $p(e)$ randomly and independently. Thus, the probability of this possible world is:

$$Pr(W_i) = \prod_{e \in E_i} p(e) \prod_{e \notin E_i} (1 - p(e)) \tag{1}$$

Since each edge has two possible sampling results (exists or not), there will be up to $2^{|E|}$ possible worlds, denoted by W, i.e., $W = (W_1, W_2, ..., W_{2^{|E|}})$.

Definition 3. (Angle) An angle $\angle(u, v, w)$ contains two edges (u, v) and (v, w) with a middle-vertex v. An angle can be regarded as a path with 3 vertices. Therefore, u and w must belong to one part of L or R, and v belongs to the other.

Definition 4. (Butterfly) A butterfly B in a possible world $W_i = (V = (L, R), E_i, w)$ consists of two vertices $u_1, u_2 \in L$ and $v_1, v_2 \in R$ with four edges $(u_1, v_1), (u_1, v_2), (u_2, v_1), (u_2, v_2) \in E_i$, denoted by $B(u_1, u_2, v_1, v_2)$. The weight of a butterfly is defined as the summation of all four edges:

$$w(B) = w(u_1, v_1) + w(u_1, v_2) + w(u_2, v_1) + w(u_2, v_2) \tag{2}$$

A possible world W_i may contain many butterflies with the same maximum weight. Those butterflies with the maximum weight are denoted by a set $S_{MB}(W_i)$:

$$S_{MB}(W_i) = \{B \mid \forall B_j \sqsubseteq W_i, w(B) \geq w(B_j)\} \tag{3}$$

The maximum weighted butterfly B in worlds W_i and W_j may be different. Specifically, the cumulative probability of a butterfly being maximum weighted among all the possible worlds is:

$$P(B) = \sum_{W_i \in W} Pr(W_i) \times \mathbf{1}[B \in S_{MB}(W_i)] \tag{4}$$

Finally, there comes the problem we studied in this paper:

Definition 5. (Most Probable Maximum Weighted Butterfly (MPMB)) Given an uncertain weighted network $G = (V = (L, R), E, p, w)$, a most probable maximum weighted butterfly is:

$$MPMB(G) = \arg\max_{B} P(B) \tag{5}$$

B. Hardness of the MPMB Problem

To present the hardness of the MPMB problem, we first define the MPMB decision problem based on Definition 5:

Definition 6. (MPMB Decision Problem) Given a probability p, whether there exists a butterfly B that $P(B) \geq p$.

The MPMB decision problem is even not an NP problem since we cannot verify a solution in polynomial time, i.e., compute the MPMB probability $P(B)$ for a given butterfly B and check whether $P(B) \geq p$. Therefore, the MPMB decision problem is an NP-Hard problem.

Lemma III.1. *Given an uncertain weighted network $G = (V = (L, R), E, p, w)$ and a butterfly $B \sqsubseteq G$, the computation of the probability of B being a maximum weighted butterfly of G is #P-Hard.*

Proof. We prove the #P-Hardness by a reduction from Monotone #2-SAT. Monotone #2-SAT is a 2-SAT counting problem that each literal is positive, i.e.,

$$\begin{aligned} &|\{x \mid F(x) = 1\}| \\ where \quad &F = c_1 \wedge c_2 \wedge ... \wedge c_r,\ c_i = (y_{i1} \vee y_{i2}) \end{aligned} \tag{6}$$

Given such a #2-SAT instance with n variables $y_1, y_2, ..., y_n$, we first construct a related uncertain weighted bipartite network $G_\# = (V_\# = (L_\#, R_\#), E_\#, p_\#, w_\#)$. Here $L_\# = \{u_0, u_1, u_2, ..., u_n, u_{n+1}, u_{n+2}\}$ and $R_\# = \{v_0, v_1, v_2, ..., v_n, v_{n+1}, v_{n+2}\}$. Each variables y_i corresponds to $u_i \in L_\#$ and $v_i \in R_\#$, while $u_0, u_{n+1}, u_{n+2}, v_0, v_{n+1}, v_{n+2}$ are six extra vertices. Edge set $E_\#$ includes four parts, (i) for each variable y_i, add an edge (u_i, v_i) with probability $p = 0.5$ and weight $w = 1$, (ii) for each clause $c_i = (y_{i1} \vee y_{i2}), i1 \neq i2$, add two edges $(u_{i1}, v_{i2}), (u_{i2}, v_{i1})$ with probability $p = 1$ and weight $w = 1$, (iii) for each clause $c_i = (y_{i1} \vee y_{i2}), i1 = i2$, we can regard it as $c_i = y_{i1} = y_{i1} \vee 1$ and use u_0 and v_0 to represent the constant true value and add two edges $(u_{i1}, v_0), (u_0, v_{i1})$ with probability $p = 1$ and weight $w = 1$, and (iv) let $B(u_{n+1}, u_{n+2}, v_{n+1}, v_{n+2})$ form a independent butterfly with four edges which have probability $p = 1$ and weight $w = 0.5$.

Note that only edge (u_i, v_i) has uncertainty. Let y_i be true if (u_i, v_i) does not exist and vice versa. Thus, each interpretation of $y_1, y_2, ..., y_n$ corresponds to a possible world, and each clause c_i corresponds to a butterfly $B_i(u_{i1}, u_{i2}, v_{i1}, v_{i2})$.

Now we try to compute the probability of $B(u_{n+1}, u_{n+2}, v_{n+1}, v_{n+2})$ being a maximum weighted butterfly. Since $w(B_i) = 4$ but $w(B) = 2$, B is maximum means all B_i are incomplete. Thus, (u_{i1}, v_{i1}) and (u_{i2}, v_{i2}) cannot exist at the same time, leading to c_i = true and F is also true. On the other hand, if B is not maximum, there must be one B_k that exists. So c_k will be false and F is false. Finally, we have:

$$\begin{aligned} P(B) &= \sum_{W_i \in W} Pr(W_i) \times \mathbf{1}[B \in S_{MB}(W_i)] \\ &= \frac{1}{2^n} \cdot \sum_{W_i \in W} \mathbf{1}[B \in S_{MB}(W_i)] \\ &= \frac{1}{2^n} \cdot |\{x \mid F(x) = 1\}| \end{aligned}$$

If $P(B)$ can be computed, the #2-SAT problem can also be solved in polynomial reduction. However, #2-SAT is a #P-Hard problem, which implies the computation of the probability of B being a maximum weighted butterfly of G is at least #P-Hard. □

IV. Baseline: Monte-Carlo with Vertex Priority

Due to MPMB being an NP-hard problem, it is intractable to solve it efficiently and accurately. In practice, to solve such kind of hard problem, Monte-Carlo sampling is a good choice. With the inspiration of the state-of-the-art butterfly enumerating algorithm, ButterFly Counting with Vertex Priority (BFC-VP) [50], we propose the baseline algorithm Monte-Carlo with Vertex Priority (MC-VP) to solve MPMB as shown in Algorithm 1.

MC-VP assigns each vertex u a priority order $o(u)$ based on their degrees. A vertex with a larger degree will have a larger priority order. Then in each trial, we randomly choose a possible world W_i from W and compute its maximum butterfly set S_{MB} with MC-VP, which generates angles first with priority order and combines them into butterflies. The only difference is that the Monte-Carlo sampling only consists of N_{mc} rounds. When N_{mc} is sufficiently large, the probability (i.e., $P(B)$) can be approximated by the mean of independent samples. Recall the theorem [51] about the guarantee of the trial number N:

Theorem IV.1. *Let μ and $\hat{\mu}$ be the target and the estimated probability, respectively. If $N \geq (1/\mu) \cdot (4\ln(2/\delta)/\epsilon^2)$, the Monte-Carlo Sampling can achieve ϵ−δ approximation, i.e., $Pr(|\hat{\mu} - \mu| > \epsilon\mu) \leq \delta$ [51].*

For example, if $P(B) = 0.01, \epsilon = 0.1, \delta = 0.01$, i.e., the probability of the related error larger than 10% is no more than 0.01, N should be larger than around $2 \cdot 10^5$.

According to Theorem IV.1, an important observation is that the speed of MC-VP is not related to the size of all possible worlds. Even when the network G is very large, we can still use a small N to sample.

Lemma IV.1. *The time and space complexity of each trial in Algorithm 1 are both $O(\sum_{e(u,v) \in E} F_{deg}(u, v) + |S_{MB}|)$ where $|S_{MB}|$ is the number of maximum butterflies and the degree function F_{deg} is defined as:*

$$F_{deg}(u, v) = \begin{cases} \bar{d}(v) & o(u) > o(v) \\ \bar{d}(u) & o(u) < o(v) \end{cases}$$

where $\bar{d}(u)$ is the expected degree of u, i.e., $\bar{d}(u) = \sum_{e(u,v) \in E} p(e)$.

Algorithm 1: Monte-Carlo with Vertex Priority

Input: trial number N_{mc}, an uncertain bipartite weighted network $G = (V = (L, R), E, p, w)$
Output: the most probable maximum butterfly B

1 $S_{MB} \leftarrow \varnothing$
2 compute priority order $o(u)$ for each vertex $u \in V$
3 **foreach** $r = 1$ *to* N_{mc} **do**
4 randomly choose W_i from W
5 **for** $u_i \in V$ **do**
6 $A \leftarrow \varnothing$
7 **for** $e_a = (u_i, u_j) \in N_{W_i}(u_i), o(u_i) > o(u_j)$ **do**
8 **for** $e_b = (u_j, u_k) \in N_{W_i}(u_j), o(u_i) > o(u_k)$ **do**
9 $\angle_{new} \leftarrow e_a \oplus e_b$
10 $A(u_k) \leftarrow A(u_k) \cup \angle_{new}$
11 **foreach** $A(u_k)$ **do**
12 **foreach** $\angle_x, \angle_y \in A(u_k), x \neq y$ **do**
13 $B \leftarrow \angle_x \oplus \angle_y$
14 **if** $w(B) > w(S_{MB})$ **then**
15 $S_{MB} \leftarrow \{B\}$
16 **else if** $w(B) = w(S_{MB})$ **then**
17 $S_{MB} \leftarrow S_{MB} \cup \{B\}$
18 **foreach** $B \in S_{MB}$ **do**
19 $P(B) \leftarrow P(B) + 1/N_{mc}$
20 **return** $\arg\max_B P(B)$

Proof. MC-VP will cost $O(\sum_{e(u,v)\in E} \min\{deg(u), deg(v)\})$ to generate all angles due to the priority order [50]. Since each possible worlds contain few edges, the real cost is the expected degree instead of the real degree. So we use the degree function $F_{deg}(u, v)$ to compute the expected degree of a vertex with lower priority order (i.e., the middle-vertex). Finally, the number of butterflies could be very large and requires to be considered. The space complexity is the same since all angles and butterflies might be saved. □

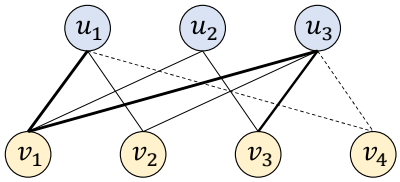
V. Ordering Sampling

The MC-VP method is faster than direct enumeration methods because it only samples a subset of all potential worlds. However, its efficiency is not guaranteed when searching for maximum weighted butterflies. This process has three main bottlenecks: (1) it lists all edges, (2) it generates and stores all angles, and (3) it creates all butterflies. Our first method, Ordering Sampling (OS), addresses these bottlenecks through three optimizations: Edge Ordering, Angle Ordering, and Fast Butterfly Creating, respectively.

A. Ordering Sampling Framework

Algorithm 2 shows our first method, Ordering Sampling (OS), and the illustration process is shown in Figure 4. It also consists of N_{os} rounds, in which all edges will be sampled independently, denoted by $\hat{E}$ in Line 5. A possible world example is shown in Figure 4(a).

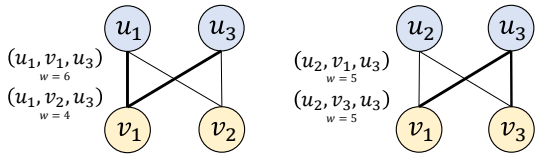
The main difference is that OS uses edge weights as the priority order. All edges are added by the weight order in Line 8 and generate all angles in Line 11, shown in Figure 4(b).



(a) A possible world. Thick lines, thin lines, and dash lines have weight 3, 2, 1, respectively.

(u_i, u_k)	$A_1(u_i, u_k)$	$A_2(u_i, u_k)$	ignored
(u_1, u_2)	(u_1, v_1, u_2) $_{w=5}$		
(u_1, u_3)	(u_1, v_1, u_3) $_{w=6}$	(u_1, v_2, u_3) $_{w=4}$	(u_1, v_4, u_3) $_{w=2}$
(u_2, u_3)	(u_2, v_1, u_3) $_{w=5}$ (u_2, v_3, u_3)		

(b) The index of Ordering Sampling Method.



(c) S_{MB} with two maximum butterflies of 4(a).

Fig. 4: The framework of the Ordering Sampling Method. There are 6 angles in Figure 4(a), classified by their endpoints (v_i, v_j) and sorted by their weights. In Figure 4(b), only five angles will be saved (marked with orange blocks) and the rest angles will be ignored. Finally, two maximum butterflies are created from $A_1(v_i, v_j)$ and $A_2(v_i, v_j)$ in Figure 4(c).

We also use w_{max} to save the weight of current maximum butterflies in Line 13. Therefore, all butterflies whose weight is w_{max} will be created in Line 15-19 and added into S_{MB} in Line 20, shown in Figure 4(c).

B. Edge Ordering

The Ordering Sampling method uses edge ordering to prevent the need to enumerate all edges. The order in which edges are added does not have any impact on the final result. Therefore, the first optimization is to sort the edges according to their weights, as indicated in Line 1 of Algorithm 2. If the upcoming edge has a lower weight, the corresponding angle and butterfly will also have a relatively lower weight. Consequently, it's acceptable to skip these lower-weight edges. Specifically, as demonstrated in Lines 1 and 9, if the current edge weight, $w(e_i)$, plus the three largest weights $\overline{w} = w(e_1) + w(e_2) + w(e_3)$, is still less than the weight of the current maximum butterfly w_{max}, all subsequent edges can be pruned.

C. Angle Ordering

MC-VP generates and stores all angles to create butterflies. However, it is possible to only consider the maximum weighted butterfly, disregarding some lower-weight angles, without affecting the result. Specifically, storing the two largest weights is sufficient. Based on this idea, Algorithm 2 uses $A_1(u_i, u_k)$ and $A_2(u_i, u_k)$ to store the largest and second-largest weighted angles, respectively.

We prove the correctness by contradiction. Suppose there is a maximum butterfly includes an angle $\angle_c$ which is ignored,

Algorithm 2: Ordering Sampling

Input: trial number N_{os}, an uncertain bipartite weighted network $G = (V = (L, R), E, p, w)$
Output: the most probable maximum butterfly B

1 sort edges $e_i \in E$ by the edge weight $w(e_i)$
2 $\overline{w} \leftarrow w(e_1) + w(e_2) + w(e_3)$ ▷ Section V-B
3 **for** $r = 1$ *to* N_{os} **do**
4 $S_{MB} \leftarrow \varnothing$
5 $\hat{E} \leftarrow$ Sample e_i with probability $p(e_i)$ independently
6 $\hat{N}_E \leftarrow \varnothing, w_{max} \leftarrow 0$
7 $A_1, A_2 \leftarrow \varnothing$
8 **foreach** $e_a = (u_i, v_j) \in \hat{E}, u_i \in L, v_j \in R$ **do**
9 **if** $w(e_a) + \overline{w} < w_{max}$ **then break** ▷ Section V-B
10 **foreach** $e_b = (v_j, u_k) \in \hat{N}_E(v_j)$ **do**
11 $\angle_{new} \leftarrow e_a \oplus e_b$
12 update $A_1(u_i, u_k), A_2(u_i, u_k)$ ▷ Section V-C
13 update w_{max} ▷ Section V-D
14 $\hat{N}_E(v_j) \leftarrow \hat{N}_E(v_j) \cup \{e_i\}$
15 **foreach** $u_i, u_k \in L, u_i \neq u_k$ **do** ▷ Section V-D
16 **if** $2 * w(A_1(u_i, u_k)) = w_{max}$ **then**
17 $B \leftarrow \angle_a \oplus \angle_b \quad \angle_a, \angle_b \in A_1(u_i, u_k), \angle_a \neq \angle_b$
18 **else if** $w(A_1(u_i, u_k)) + w(A_2(u_i, u_k)) = w_{max}$ **then**
19 $B \leftarrow \angle_a \oplus \angle_b \quad \angle_a \in A_1(u_i, u_k), \angle_b \in A_2(u_i, u_k)$
20 $S_{MB} \leftarrow S_{MB} \cup B$
21 **foreach** $B \in S_{MB}$ **do**
22 $P(B) \leftarrow P(B) + 1/N_{os}$
23 **return** $\arg\max_B P(B)$

i.e., $A_1(u_i, u_k)$ and $A_2(u_i, u_k)$ are not empty but $\angle_c$ not belongs to any one. However, we can always use $\angle_a \in A_1(u_i, u_k)$ and $\angle_b \in A_2(u_i, u_k)$ to form a larger butterfly, which contradicts the condition.

Table II shows the updating process in Line 12 with different cases. $w(\angle_{new})$ is the weight of $\angle_{new}$. $w(A_1)$ and $w(A_2)$ denote the weight of angles saved in $A_1(u_i, u_k)$ and $A_2(u_i, u_k)$, respectively.

TABLE II: Updating cases.

condition	A_1	A_2
$w(\angle_{new}) > w(A_1)$	$\{\angle_{new}\}$	A_1
$w(\angle_{new}) = w(A_1)$	$A_1 \cup \{\angle_{new}\}$	A_2
$w(\angle_{new}) > w(A_2)$	A_1	$\{\angle_{new}\}$
$w(\angle_{new}) = w(A_2)$	A_1	$A_2 \cup \{\angle_{new}\}$
$w(\angle_{new}) < w(A_2)$	A_1	A_2

D. Fast Butterfly Creating

Unlike the MC-VP method, which creates all possible butterflies before selecting the maximum weighted ones, Algorithm 2 directly creates maximum weighted butterflies based on $A_1(u_i, u_k)$ and $A_2(u_i, u_k)$. There are two specific cases. If $|A_1(u_i, u_k)| > 1$, each pair of angles in $A_1(u_i, u_k)$ can create a butterfly with a weight of $2 \cdot w(A_1(u_i, u_k))$. If $|A_1(u_i, u_k)| = 1$, the only angle in $A_1(u_i, u_k)$ should match each angle in $A_2(u_i, u_k)$, and the weight is $w(A_1(u_i, u_k)) + w(A_2(u_i, u_k))$. Consequently, in line 13 of Algorithm 2, the weight of the current maximum weighted butterflies, w_{max}, can be updated in real-time for optimization, as described in Section V-B. Furthermore, in lines 16-19 of Algorithm 2, the true maximum weighted butterflies set, S_{MB}, can be created.

Lemma V.1. *The time and space complexity of each trial in Algorithm 2 are both* $O(\min\{\sum_{u \in L} \overline{d^2}(u), \sum_{v \in R} \overline{d^2}(v)\} + |S_{MB}|)$*, where* $\overline{d^2}(u)$ *is the expected square of* u*'s degree.*

Proof. The complexity of Lines 8-14 on the backbone graph is $O(\sum_{v \in R} deg^2(v))$, since there are $O(deg^2(v))$ angles whose middle vertex is v, and each angle will be added into A_1 and A_2 at most once. Same as Lemma IV.1, the degree in a possible world is relatively lower. Then, the average time complexity is expected of $deg^2(v)$, i.e., $\overline{d^2}(v)$. Note two disjoint parts are symmetrical [52], the time complexity is $O(\min\{\sum_{u \in L} \overline{d^2}(u), \sum_{v \in R} \overline{d^2}(v)\})$. In Lines 15-20, only maximum weighted butterflies will be created, which costs $O(|S_{MB}|)$. Therefore, the total time complexity is $O(\min\{\sum_{u \in L} \overline{d^2}(u), \sum_{v \in R} \overline{d^2}(v)\} + |S_{MB}|)$. The space complexity is the same to save all angles and maximum weighted butterflies. □

In fact, with the optimization of Edge Ordering, Algorithm 2 requires handling fewer edges and significantly reduces the actual execution time. The only issue is that $\overline{d^2}(u)$ may not seem intuitive to compute directly. An alternative is to use the square of the expected degree of u, i.e., $\bar{d}^2(u) = (\bar{d}(u))^2$. While this approximation might incur an additional cost, it's still negligible given the Edge Ordering pruning.

Lemma V.2. *To achieve* $\epsilon - \delta$ *approximation, the trial number of Algorithm 2 should at least* $N_{os} \geq (1/P(B)) \cdot (4\ln(2/\delta)/\epsilon^2)$.

Proof. Algorithm 2 directly estimates $P(B)$, so let $\mu = P(B)$ in Lemma IV.1, the lowerbound is $(1/P(B)) \cdot (4\ln(2/\delta)/\epsilon^2)$. □

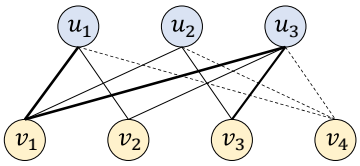
VI. Ordering-Listing Sampling

The OS Method demonstrates high efficiency and achieves a low time complexity in a single iteration. However, this Monte-Carlo-based algorithm requires numerous trials to ensure accuracy. Our second method, called Ordering-Listing Sampling (OLS), is specialized for multiple trials to avoid sampling the whole network.

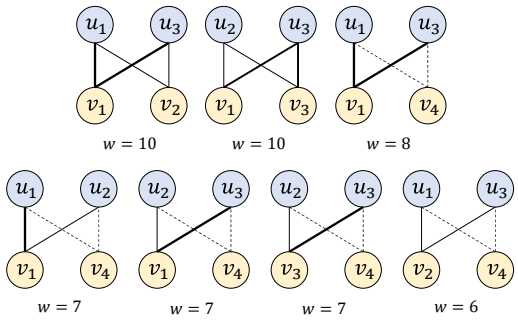
OLS involves an additional preparing phase to list some candidate butterflies first, and then samples on these candidate butterflies directly, ignoring the backbone network. Specifically, OLS can invoke OS for a few iterations to list some candidate butterflies as a candidate set C_{MB}, and then apply some efficient sampling algorithms over C_{MB}.

A. Ordering-Listing Sampling Framework

Figure 5 presents an example of the OLS framework. The candidate maximum weighted butterflies set, C_{MB}, of the backbone graph contains at most 7 butterflies, which are sorted by their weights. After obtaining the candidate set C_{MB}, we can directly sample butterflies on C_{MB} by the weight order, instead of sampling the whole network. Upon finding



(a) A possible world. Thick lines, thin lines, and dash lines have weight 3, 2, 1, respectively.



(b) All 7 potential candidate butterflies.

Fig. 5: The backbone graph and candidate butterflies sorted by weight.

Algorithm 3: Ordering-Listing Sampling

Input: trial number N_{os}, N_{ls}, an uncertain bipartite weighted network $G = (V = (L, R), E, p, w)$
Output: the most probable maximum butterfly B

1 $C_{MB} \leftarrow \varnothing$
2 **for** $r = 1$ *to* N_{os} **do**
3 $\quad S_{MB} \leftarrow$ the maximum butterflies set by Algorithm 2
4 $\quad C_{MB} \leftarrow C_{MB} \cup S_{MB}$
5 $P(\cdot) \leftarrow$ ProbabilityEstimating(C_{MB}, N_{ls})
6 **return** $\arg\max_B P(B)$

a butterfly, other butterflies with the same weight should also be sampled, and subsequent butterflies will be skipped.

The OLS method, as shown in Algorithm 3, consists of two phases: the preparing phase (Lines 2-4) and the sampling phase (Line 5). The preparing phase employs the OS method to list some candidate butterflies. Unlike conventional methods that require numerous trials and the accumulation of probabilities for precision, OLS uses fewer trials to identify maximum butterflies S_{MB} and stores them in the candidate set C_{MB}. Then the sampling phase samples on a small candidate set C_{MB} with some efficient sampling algorithms.

In short, the preparing phase will filter possible butterflies and the sampling phase is responsible for precise probabilities.

B. Preparing Phase

Instead of reporting the exact probability $P(B_i)$ of each butterfly B_i, the preparing phase only requires the existence of each butterfly. This approximate method can retain high-probabilities butterflies while ignoring other butterflies. Specifically, if we run the OS method for more iterations, more butterflies will be added to the candidate set. To make sure that the target MPMB does not be filtered out, we have:

Lemma VI.1. *Suppose B^* is the true answer to the MPMB problem, then with a high probability, $B^* \in C_{MB}$.*

Algorithm 4: ProbabilityEstimating (Karp-Luby [48])

Input: the candidate set C_{MB}, trial number N_{kl}
Output: the estimated probability of each butterfly $P(\cdot)$

1 **foreach** $B_i \in C_{MB}$ **do**
2 $\quad Cnt_i \leftarrow 0$
3 $\quad L(i) \leftarrow$ the largest index that $L(i) < i, w(B_{L(i)}) > w(B_i)$
4 $\quad S_i \leftarrow \sum_{j \in [1, L(i)]} Pr[E(B_j \backslash B_i)]$
5 $\quad$ **for** $r = 1$ *to* N_{kl} **do**
6 $\quad\quad$ Sample $j \in [1, L(i)]$ with probablity $\frac{Pr[E(B_j \backslash B_i)]}{S_i}$
7 $\quad\quad$ Sample a possible world W such that $B_j \backslash B_i \subseteq E_W$
8 $\quad\quad$ **if** $\forall k < j, B_k \backslash B_i \not\subseteq E_W$ **then**
9 $\quad\quad\quad Cnt_i \leftarrow Cnt_i + 1$
10 $\quad P(B_i) \leftarrow (1 - \frac{Cnt_i}{N_{kl}} \times S_i) \times Pr[E(B_i)]$
11 **return** $P(B_i)$ *for all* $B_i \in C_{MB}$

Proof. In each trial of the preparing phase, a butterfly B is in the set S_{MB} if and only if B is the maximum butterfly in the possible world. Therefore, $Pr[B \in S_{MB}] = P(B)$.

The preparing phase has N_{os} trials. As long as $B \in S_{MB}$ holds once, the butterfly B will appear in the candidate set C_{MB}, where the probability is $1-(1-P(B))^{N_{os}}$. Even when $P(B) = 0.1$ and $N_{ls} = 20$, the probability is nearly 90%. □

C. Sampling Phase

Now the probability estimating function only requires to compute probabilities for candidate butterflies. We first sort these candidate butterflies by the weight order and estimate the probability. To achieve high precision with fewer sampling trials, we introduce two efficient sampling algorithms, namely Karp-Luby sampling and our optimized sampling.

Karp-Luby sampling [48], [12]. Algorithm 4 is the Karp-Luby sampling algorithm. The KL sampling is developed to estimate the probability that at least one event exists if these events are not independent. If we directly summate the probability of each event up, the result could be relatively large since one possible world may contain multiple events. The major idea of the KL sampling is to define the priority that only one event can contribute to each possible world.

To make a butterfly B_i being maximum, (1) the butterfly B_i should exist, and (2) no larger butterfly $B_j (j \leq L(i))$ exists, where $L(i)$ indicates the last butterfly with a larger weight, i.e., $\forall j \in [1, L(i)], w(B_j) > w(B_i)$. Therefore, the KL sampling can be applied to estimate the opposite case that at least one butterfly $B_j (j \leq L(i))$ exists.

Suppose B_i exists, we only consider other edges $B_j \backslash B_i$ for other butterflies B_j. At first in Line 4, we compute the total summation probability S_i. Then the KL sampling iteratively chooses a possible world in S_i that contains at least one butterfly, saying B_j in Lines 6-7. Here the priority is defined that only the butterfly with the minimum index can be counted. Therefore, in Lines 8-9, we have to check whether another butterfly with a lower index k exists. If not, B_j can contribute to the result.

Then, we have the analyses of the time and space complexity of Algorithm 4 in the following lemma:

Lemma VI.2. *The time complexity of Algorithm 4 with Karp-Luby sampling is $O(N_{kl} \cdot |C_{MB}|^2)$ and the space complexity is $O(|C_{MB}|)$.*

Proof. Each butterfly requires N_{kl} trials, where each trial costs $O(|C_{MB}|)$ in the worst case. So the total time complexity with N_{kl} trials is $O(N_{kl} \cdot |C_{MB}|^2)$. The KL sampling requires no extra memory so the space complexity is $O(|C_{MB}|)$. □

Optimized sampling. Algorithm 4 applies the Karp-Luby algorithm to increase accuracy and reduce the trial number for each butterfly. However, the sampling process between butterflies is independent, thus every butterfly requires N_{kl} trials.

Algorithm 5 is our optimized probability estimating method, which samples butterflies by the weight order, and all butterflies share each trial. When a butterfly exists, the sampling process will not terminate but save the weight w_{max} (Lines 8-10), continuing sampling until the current butterfly weight is less than w_{max} (Lines 5-6).

Lemma VI.3. *The time complexity of Algorithm 5 with our optimized sampling is $O(N_{op} \cdot |C_{MB}|)$ and the space complexity is $O(|C_{MB}|)$.*

Proof. Algorithm 5 will enumerate at most $O(|C_{MB}|)$ edges in each trial. Thus, the total time complexity using the optimized sampling algorithm is $O(N_{op} \cdot |C_{MB}|)$. The space complexity is still $O(|C_{MB}|)$ with no extra memory required. □

D. Trial Number Lowerbound

Next, we explain why Algorithm 5 is much more efficient than Algorithm 4. To achieve the same approximation, two sampling algorithms may require different trial numbers N_{kl} and N_{op}. Therefore, we cannot directly compare their time complexity.

Recalling that the trial lower-bound for Monte-Carlo-based sampling is $N \geq (1/\mu) \cdot (4\ln(2/\delta)/\epsilon^2)$. However, different sampling processes will affect realistic trial numbers. Specifically, we have:

Lemma VI.4. *Let μ and $\hat{\mu}$ is the target and the estimated probablity of a butterfly B_i, i.e., $\mu = P(B_i)$. To achieve $\epsilon - \delta$ approximation using Algorithm 4 and Algorithm 5, i.e., $Pr(|\hat{\mu} - \mu| > \epsilon\mu) \leq \delta$, the trial number lower-bound should respectively be:*

$$N_{op} \geq (1/\mu) \cdot (4\ln(2/\delta)/\epsilon^2)$$
$$N_{kl} \geq Pr[E(B_i)] \times S_i \times \left(\frac{Pr[E(B_i)]}{\mu} - 1\right)(1/\mu) \cdot (4\ln(2/\delta)/\epsilon^2)$$

where $Pr[E(B_i)]$ is the existing probability of the butterfly B_i, and S_i is as shown in Algorithm 4.

Proof. Our optimized sampling algorithm directly estimates $P(B_i)$, and thus $N_{op} \geq (1/\mu) \cdot (4\ln(2/\delta)/\epsilon^2)$. We prove the remaining part by analyzing the ratio of two trial numbers N_{kl} and N_{op}.

Since Algorithm 4 estimates $\frac{Cnt_i}{N_{kl}}$ in Line 9 instead of $P(B_i)$, we have to convert the error and confidence. The following equation shows the transform between $\frac{Cnt}{N_{kl}}$ and $P(B_i)$:

$$P(B_i) = \left(1 - \frac{Cnt_i}{N_{kl}} \times S_i\right) \times Pr[E(B_i)]$$

Suppose Algorithm 4 satisfies the $\epsilon - \delta$ condition on $\frac{Cnt_i}{N_{kl}}$, the result $P(B_i)$ is $\epsilon \cdot \frac{Cnt_i}{N_{kl}} \times S_i \times Pr[E(B_i)]/P(B_i) - \delta$ approximation. Thus, the ratio of trial numbers of the two methods is:

$$\begin{aligned}\frac{N_{kl}}{N_{op}} &= \frac{\mu_{op}\epsilon_{op}^2}{\mu_{kl}\epsilon_{kl}^2} = \frac{P(B_i) \cdot \epsilon^2 \cdot (\frac{Cnt_i}{N_{kl}} \times S_i \times Pr[E(B_i)])^2}{P(B_i)^2 \cdot \frac{Cnt_i}{N_{kl}} \cdot \epsilon^2} \\ &= \frac{Cnt_i}{N_{kl}} \cdot S_i^2 \cdot \frac{Pr[E(B_i)]^2}{P(B_i)} \\ &= Pr[E(B_i)] \times S_i \times \left(\frac{Pr[E(B_i)]}{P(B_i)} - 1\right)\end{aligned}$$

Here $P(B_i)$ is the target estimated probability and can be replaced by μ. So we have:

$$\frac{N_{kl}}{N_{op}} = Pr[E(B_i)] \times S_i \times \left(\frac{Pr[E(B_i)]}{\mu} - 1\right) \tag{8}$$

With $N_{op} \geq (1/\mu) \cdot (4\ln(2/\delta)/\epsilon^2)$, the lemma is proved. □

Using Lemma VI.4, two sampling algorithms can precisely set a unique trial number for each butterfly. Moreover, we can analyze the time complexity to report the result with the same $\epsilon - \delta$ approximation. Considering the time complexity of two methods presented in Lemma VI.2 and Lemma VI.3, when the time complexities of the two methods are the same, we have:

$$\frac{N_{kl}}{N_{op}} = \frac{1}{O(|C_{MB}|)} \tag{9}$$

Considering that $|C_{MB}|$ and S_i could be very large, Equation 8 is generally exceed $\frac{1}{O(|C_{MB}|)}$, indicating the huge advantage of our optimized sampling algorithm in Algorithm 5.

E. Approximation Guarantees

A remaining problem is that the preparing phase may miss some butterflies with low probabilities. These butterflies themselves might not be the answer but could affect the computation of other candidate butterflies. In the following, we will show that the approximation error of $P(B_i)$ is bounded.

Algorithm 5: ProbabilityEstimating (optimized)

Input: the candidate set C_{MB}, the trial number N_{op}
Output: the estimated probability of each butterfly $P(\cdot)$

```
1  for r = 1 to N_op do
2      S_MB ← ∅
3      w_max ← 0
4      foreach B_k(u_i, u_j, v_i, v_j) ∈ C_MB do
5          if w(B_k) < w_max then
6              break
7          sampling each edge in B_k if not sampled previously
8          if B_k exists then
9              S_MB ← S_MB ∪ {B_k}
10             w_max ← w(B_k)
11     foreach B ∈ S_MB do
12         P(B) ← P(B) + 1/N_op
13 return P(B_i) for all B_i ∈ C_MB
```

Lemma VI.5. *Suppose Algorithm 3 returns the estimated probability $\hat{P}(B_i)$ and its exact probability is $P(B_i)$, the absolute error on estimating $P(B_i)$ from skipping some butterflies in the preparing phase is bounded, i.e.,*

$$\hat{P}(B_i) - P(B_i) \leq \sum_{\substack{1 \leq j \leq L(i) \\ B_j \notin C_{MB}}} P(B_j)$$

where $L(i)$ indicates the largest index: $w(B_{L(i)}) > w(B_i)$.

Proof. To make a butterfly B_i being maximum, (1) the butterfly B_i should exists, and (2) no larger buttefly $B_j (j \leq L(i))$ exists:

$$P(B_i) = \Pr\left[\left(\bigcap_{j=1}^{L(i)} \overline{E(B_j)}\right) \cap E(B_i)\right]$$

Considering edges that are both in B_i and B_j, we have:

$$\overline{E(B_j)} \cap E(B_i) = \overline{E(B_j \backslash B_i)} \cap E(B_i)$$

Now $\overline{E(B_j \backslash B_i)}$ and $E(B_i)$ are independent. Hence:

$$\begin{aligned} P(B_i) &= \Pr[E(B_i)] \cdot \Pr\left[\bigcap_{j=1}^{L(i)} \overline{E(B_j \backslash B_i)}\right] \\ &= \Pr[E(B_i)] \cdot \left(1 - \Pr\left[\bigcup_{j=1}^{L(i)} E(B_j \backslash B_i)\right]\right) \end{aligned}$$

Let $B_j^* = B_j \backslash B_i$, the following transformation holds:

$$\begin{aligned} &\bigcup_{j=1}^{L(i)} E(B_j^*) \\ =&E(B_1^*) \cup \left(\overline{E(B_1^*)} \cap E(B_2^*)\right) \cup \left(\overline{E(B_1^*)} \cap \overline{E(B_2^*)} \cap E(B_3^*)\right) ... \\ =&\bigcup_{j=1}^{L(i)} \left(\bigcap_{t=1}^{j-1} \overline{E(B_t^*)} \cap E(B_j^*)\right) \end{aligned}$$

Since all $\cap_{t=1}^{j-1} \overline{E(B_t^*)} \cap E(B_j^*)$ are independent, the principle of additivity is applied:

$$P(B_i) = \Pr[E(B_i)] \cdot \left(1 - \sum_{j=1}^{L(i)} \Pr\left[\bigcap_{t=1}^{j-1} \overline{E(B_t^*)} \cap E(B_j^*)\right]\right)$$

The estimated probability, $\hat{P}(B)$, is similar except missing some butterflies out of the candidate set:

$$\begin{aligned} \hat{P}(B_i) &= \Pr[E(B_i)] \cdot \left(1 - \sum_{\substack{1 \leq j \leq L(i) \\ B_j \in \mathcal{C}}} \Pr\left[\bigcap_{\substack{1 \leq t \leq j-1 \\ B_t \in \mathcal{C}}} \overline{E(B_t^*)} \cap E(B_j^*)\right]\right) \\ &\leq \Pr[E(B_i)] \cdot \left(1 - \sum_{\substack{1 \leq j \leq L(i) \\ B_j \in \mathcal{C}}} \Pr\left[\bigcap_{t=1}^{j-1} \overline{E(B_t^*)} \cap E(B_j^*)\right]\right) \end{aligned}$$

Therefore, we have:

$$\begin{aligned} \hat{P}(B_i) - P(B_i) &\leq \Pr[E(B_i)] \cdot \left(\sum_{\substack{1 \leq j \leq L(i) \\ B_j \notin \mathcal{C}}} \Pr\left[\bigcap_{t=1}^{j-1} \overline{E(B_t^*)} \cap E(B_j^*)\right]\right) \\ &= \sum_{\substack{1 \leq j \leq L(i) \\ B_j \notin C_{MB}}} \Pr\left[\bigcap_{t=1}^{j-1} \overline{E(B_t^*)} \cap E(B_j^*) \cap E(B_i)\right] \\ &\leq \sum_{\substack{1 \leq j \leq L(i) \\ B_j \notin C_{MB}}} \Pr\left[\bigcap_{t=1}^{L(j)} \overline{E(B_t)} \cap E(B_j) \cap E(B_i)\right] \end{aligned}$$

Omitting $E(B_i)$, the approximation error is bounded by:

$$\hat{P}(B_i) - P(B_i) \leq \sum_{\substack{1 \leq j \leq L(i) \\ B_j \notin C_{MB}}} P(B_j)$$

□

To explain it, the approximation error happens when B_j could have been a maximum butterfly but is missed in the candidate set C_{MB}. Followed by the Lemma VI.2, a missing butterfly B_j in the preparing phase may have a small $P(B_j)$ with a high probability. Moreover, the derivation includes many inequality scales, thus the real error can be much lower.

VII. Multiple MPMB Solutions

To address the limitation that MPMB only return one individual butterfly and enhance the scalability in large-scale network, we propose an extension to the MPMB problem: top-k MPMB problem. This approach allows for the detection of multiple significant butterflies, providing a more comprehensive view of important areas.

For MC-VP and OS: After estimating the probability $P(B_i)$ for all butterflies B_i, instead of selecting only the maximum, we can sort the butterflies S_{MB} in descending order of probability and select the top-k butterflies.

For OLS: We can apply a similar operation to the butterflies in the candidate set C_{MB}. By sorting all candidates butterflies, we can identify the top-k butterflies in the candidate set. This operation is supported by Lemma VI.1, which demonstrates that high-probability butterflies are likely to be included in the candidate set.

VIII. Experiment

In this section, we evaluate the efficiency of the proposed methods, OS and OLS, in comparison to the baseline method, Monte-Carlo sampling with the BFC-VP algorithm (MC-VP). We also implement the Karp-Luby algorithm on OLS, referred to as OLS-KL. All methods are implemented in C++17 and compiled using -O3 optimization. All experiments are conducted on an Intel Core i5-13500HX CPU @ 2.5GHz with 32 GB of memory.

The experiment includes two main aspects, efficiency and effectiveness. For efficiency, we evaluate two major metrics, i.e., the executing time and the sampling trial numbers. For effectiveness, we present the convergence trend with increasing trial numbers to verify our proposed lower bound.

A. Datasets

Table III provides details for the four datasets used in various experiments. The ABIDE [34] dataset presents brain network data, with vertices representing different Regions of Interest (ROIs) and edges indicating ROI connections. Edge weight and probability denote the distance and correlation between two ROIs, respectively.

MovieLens [53] and Jester [54] are rating datasets. Here, the left vertices represent users and the right vertices represent entities (movies or jokes). The edge weight signifies the rating point, while edge probability shows the reliability of this

TABLE III: Dataset Details.

Datasets	$\|E\|$	$\|L\|$	$\|R\|$	weight	probability
ABIDE	3,364	58	58	physical distance	correlation
MovieLens	100,836	610	9,724	rating	reliability
Jester	4,136,360	100	73,421	rating	reliability
Protein	39,471,870	186,773	186,772	interaction	Normal(0.5,0.2)

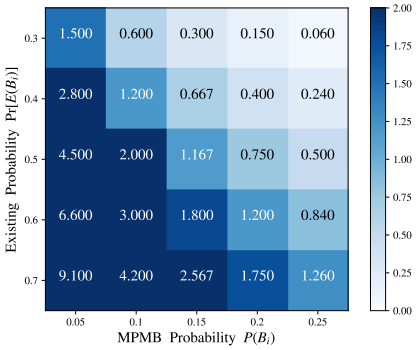


Fig. 6: Matrix of trial number ratio N_{kl}/N_{op} when $S_i = 1$ computed by Equation 8.

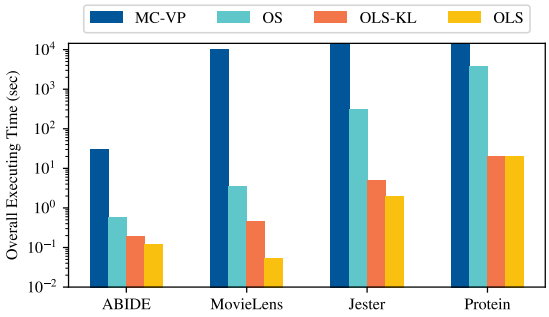


Fig. 7: Overall executing time on four datasets.

rating, defined as the relative difference between the user rating and the average rating.

Protein[1] is a dataset where vertices are proteins and edges are interaction strength between two proteins. As the original network is a deterministic non-bipartite graph, we preprocessed this dataset to randomly generate probabilities with normal distribution and divided vertices equally by their odd and even IDs.

B. Parameters

The trial number N should be carefully chosen with the same accuracy. For MC-VP and OS, we set $N_{mc} = N_{os} = 2 \times 10^4$ by default suppose $\mu = 0.05$ and $\epsilon = \delta = 0.1$ in Theorem IV.1, i.e., the probability of the relative error on $P(B) = 0.05$ exceeding 10% is no more than 10%. For OLS-KL and OLS, we set $N_{os} = 100$ for the preparing phase to make sure that the missing probability of a butterfly with $P(B) = 0.05$ is no more than 0.5%. As for the sampling phase, OLS-KL follows Equation 8 to dynamically compute the trial number N_{kl} for each butterfly, while OLS also has $N_{op} = 2 \times 10^4$.

The Karp-Luby algorithm uses dynamic trial numbers computed during the sampling process. Figure 6 shows some general combinations of MPMB probability $P(B_i)$ and exsiting probability $Pr[E(B_i)]$ when $S_i = 1$. A larger ratio (or darker block) means more trials for the Karp-Luby algorithm. Since (1) S_i is incremental and very likely to exceed 1 and (2) this ratio has to compare with $\frac{1}{C_{MB}}$, which is much lower than 1, our method show a significant advantage, especially for a precise result and a large candidate set.

TABLE IV: Trial numbers of four methods in different phases.

Sampling Methods	Preparing Phase	Sampling Phase
MC-VP	-	20,000
OS	-	20,000
OLS-KL	100	dynamic
OLS	100	20,000

[1]Available at https://cn.string-db.org/

C. Sampling Efficiency

Overall executing time. The first experiment is about sampling efficiency. All methods apply the default parameter setting declared in Section VIII-B and the time limitation is 14,400 sec (4 hours). Figure 7 presents the overall executing time for MC-VP, OS, OLS-KL, and OLS methods on four datasets. MC-VP is too time-consuming and cannot finish the process on two larger datasets within 4 hours so we omit it in further experiments.

The result shows that both OS and OLS methods obtain significant improvement, while the baseline method. Compared with MC-VP, OS has at least 10^3x speedup due to the pruning optimizations. Compared with OS, OLS-KL and OLS methods are more efficient with at most around 180x speedup.

The Comparison between OLS-KL and OLS is twofold. On smaller datasets (ABIDE, MovieLens, and Jester), our optimized probability estimating process is more efficient and requires lower trial numbers (discussed in Section VI-D and visualized in Figure 6), achieving over 8x speedup. However, this leading advantage is almost negligible on the MouseGene dataset.

Varying N in the sampling phase. Figure 8 further presents the preparing time (when N=0%, only for OLS-KL and OLS) and sampling time (N varying in $25\%, 50\%, 75\%, 100\%$). For two smaller datasets (ABIDE and MovieLens), the preparing phase is fast, while in two larger dataset (Jester and Protein), the preparing time is the dominant cost. As a result, OLS-KL/OLS is up to 180x faster than OS especially in Jester and Protein, since OLS-KL and OLS only require only $1/200$ trials in the preparing phase. Even in ABIDE and MovieLens, OLS-KL and OLS still perform better than OS due to the simple sampling process. Compared to OLS-KL, OLS with our optimized sampling algorithm further achieves around $3 \sim 8$x speedup, which is quite evident in ABIDE and MovieLens.

Scalability. We also evaluate the scalability of these algorithms by randomly choosing $25\%, 50\%, 75\%, 100\%$ of vertices to form a new dataset. Figure 9 shows a better scalability of OS, whose time complexity is directly related to degrees.

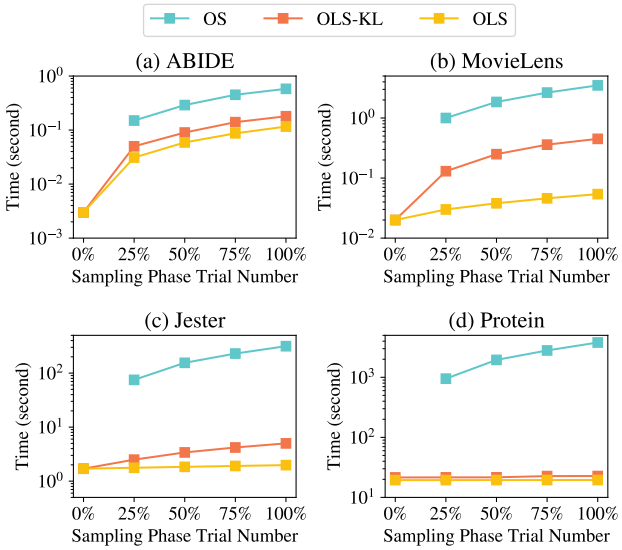


Fig. 8: Specific executing time with different trial numbers.

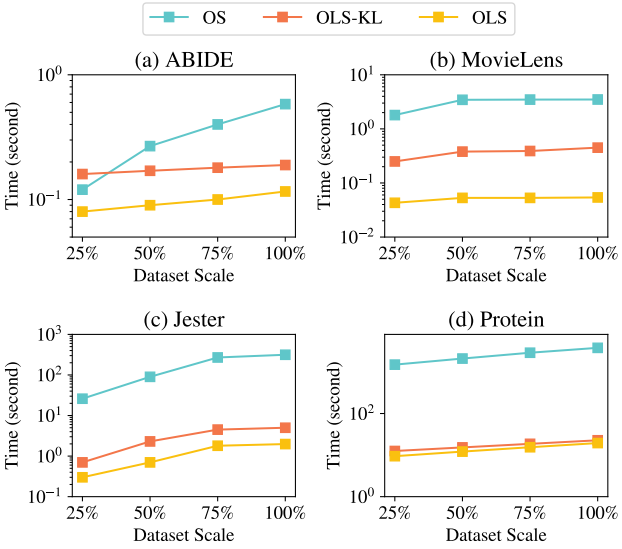


Fig. 9: Specific executing time with different dataset scales.

On the contrary, the sampling phase in OLS-KL and OLS has a fixed size of the candidate set so the scalability in ABIDE and MovieLens is not evident. However, the preparing phase has a dominant time cost on Jester and Protein datasets, showing a similar scalability compared with OS.

Trial numbers. Figure 10 visualizes the trial number for each candidate butterfly in detail. Specifically, each bar indicates a candidate butterfly and the height is the trial number ratio. The ratio value N_{kl}/N_{op} is computed by Equation 8 and $\mu = 0.1$. In general, the average ratio changes from 0.1 to 2, which is consistent with Figure 6.

This ratio requires to be compared with $\frac{1}{C_{MB}}$ according to Equation 9. Therefore, we also use a red line to indicate the value of $\frac{1}{C_{MB}}$, where most bars significantly exceed this balanced value. Though these theoretical analyses are just an upper bound and the actual sampling process is faster with some early terminations, this experiment can still confirm that our optimized sampling method is more efficient.

In addition, Figure 10(c) shows many same ratios. The

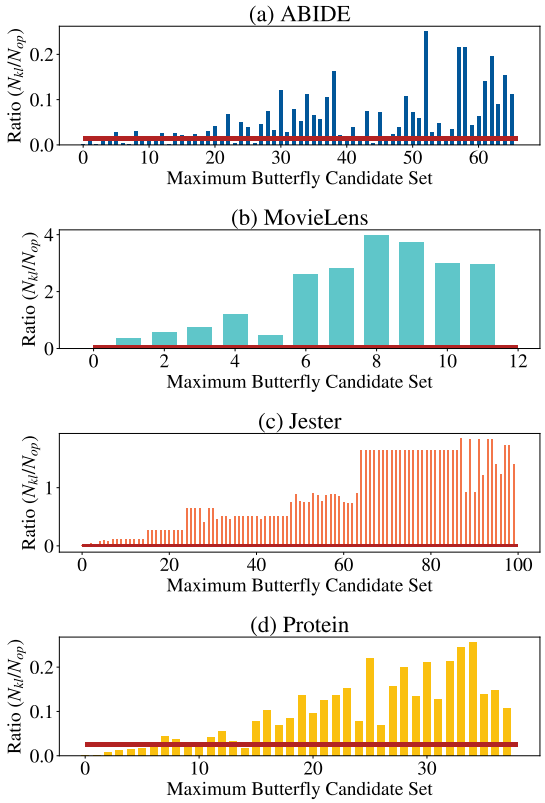


Fig. 10: The ratio of the required trial numbers in the Karp-Luby algorithm and ours, i.e., N_{kl}/N_{op}. The red line indicates balancing value $1/C_{MB}$.

reason is that some people give the same rating to one joker, leading to many butterflies with the same weights and probabilities. In fact, this experiment also confirms Lemma VI.5 especially when some candidate butterflies have the same weight.

D. Trials Effectiveness

To examine the trial number declared in Section VIII-B in experiments, we trace the probability convergence trend of a butterfly with $P(B_i)\approx 0.05$ using all three methods.

Sampling Phase Trial Number. Figure 11 shows the result with increasing trial numbers in the sampling phase with twice the trial number. The strip zone indicates the error bound, whose width is twice the relative error i.e., $2\epsilon = 0.01$.

The probability trends of all three methods have some fluctuation only in the initial 50% trials and then start convergence until becoming completely stable after 100% trials (or even earlier), which can be bounded into 2ϵ error zone. Although we cannot find the exact MPMB probability, we can still imply the correctness of three different sampling methods, since the basic OS method has a full theoretical guarantee.

Comparing OS and OLS, the latter method misses some butterflies with low MPMB probabilities in the candidate set. The result verifies that the 100 trials for the ordering phase are enough, and the error given in Lemma VI.5 will not affect the final result too much.

Comparing OLS and OLS-KL, the result verifies the sufficiency of the trial number lower-bound of the Karp-Luby

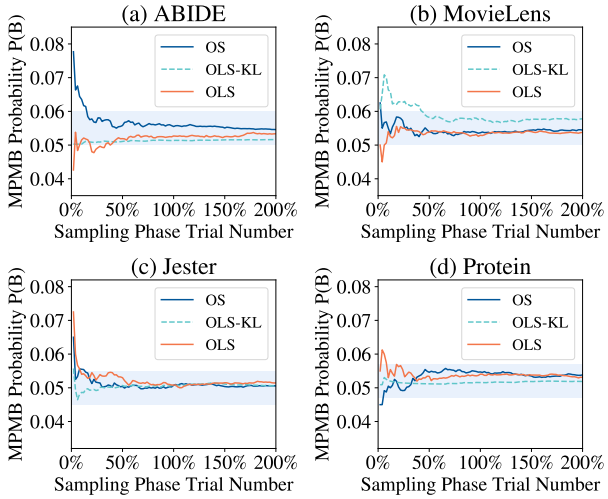


Fig. 11: MPMB probability convergence trend with twice the trial number of the listing phase over four datasets.

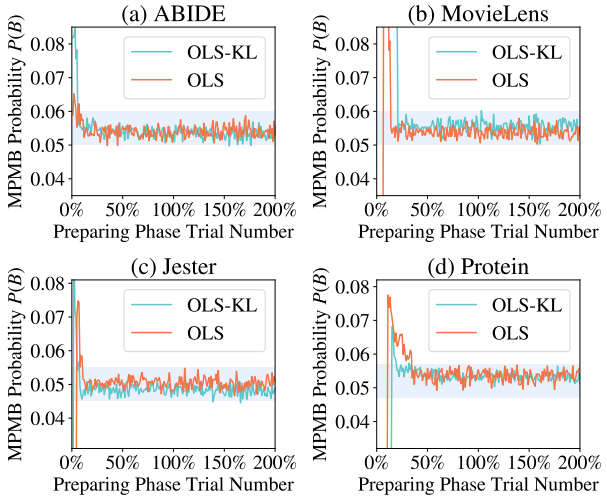


Fig. 12: MPMB probability trend with twice the trial number of the ordering phase over four datasets.

algorithm shown in Lemma VI.4, which means the above comparison between OLS and OLS-KL is effective since they have the same approximation guarantee with the trial number.

Preparing Phase Trial Number. Moreover, we want to check whether the trial number in the preparing phase is enough. Figure 12 plots the result also with twice the trial number and a 2ϵ error zone. Note that each experiment is conducted independently so the trend is not convergent but fluctuant.

Initially, the probabilities is either zero (i.e., not included in the candidate set) or too large (i.e., the candidate set is too small). After around 50% trials, the probabilities start to stable and almost within the error zone, which means a small number of trials is sufficient to find candidate butterflies.

E. Memory Consumption

Figure 13 presents the memory usage situation of four methods, MC-VP, OS, OLS-KL, and OLS. In a smaller dataset ABIDE, few butterflies exist so all four methods use similar memory to save the network information. While the scale of the network becomes larger, the number of angles and butterflies that are required to be saved is also increasing. However, OS, OLS-KL, and OLS all apply optimizations mentioned in Section V to ignore some worse solutions. As a result, the difference between these three methods is not notable since the extra space is far less than the network itself, while the MC-VP method requires another 50% memory consumption to save all temporary information.

F. Experimental Results Summary

In summary, the OS method method surpasses the baseline method by a factor of 10^3, demonstrating a significant improvement in efficiency. The OLS method delivers an additional speedup of 180x compared to the OS method. The sampling technique proposed in the OLS method is highly efficient, necessitating fewer trials and thus outperforming the widely-used Karp-Luby algorithm with up to 8x speedup, given the same precision.

In terms of effectiveness, all experimental results start to converge before reaching half the theoretical trial number,

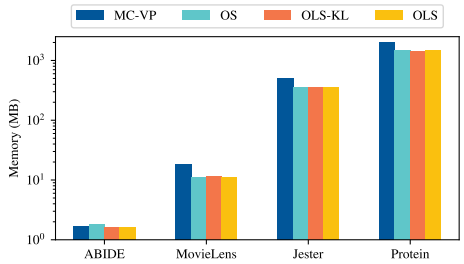


Fig. 13: Memory consumption on four datasets.

thereby substantiating our theoretical trial numbers as adequate. Additionally, the results from all methods are very close and bounded in an acceptable error, showing the correctness of our optimized sampling process.

The peak memory usage is approximating 2GB, wherein the majority of memory is allocated for the original network and the index size remains tiny.

IX. Conclusion

In this paper, we focus on uncertain butterflies on the uncertain bipartite network and introduce the Most Probable Maximum Weighted Butterfly (MPMB) problem. We prove the NP-Hardness of searching for MPMBs and introduced two methods: Ordering Sampling (OS) and Ordering-Listing Sampling (OLS). The OS method involves randomly sampling possible worlds to search for MPMBs, while the OLS method first use OS to generate candidate MPMBs initially, and then calculates MPMB probabilities over the relatively small candidate set. We present theoretical trial number bounds and approximation guarantees to meet various precision requirements. Our experimental results confirmed the efficiency and effectiveness of our methods.

Acknowledgment

Peng Cheng is supported by NSFC under Grant No. 62102149. Long Yuan is supported by NSFC under Grant No. 62472225. Xuemin Lin is supported by NSFC U2241211 and U20B2046. Corresponding Author: Peng Cheng.

REFERENCES

[1] P. Parchas, F. Gullo, D. Papadias, and F. Bonchi, "Uncertain graph processing through representative instances," *ACM Transactions on Database Systems (TODS)*, vol. 40, no. 3, pp. 1–39, 2015.

[2] A. Khan and L. Chen, "On uncertain graphs modeling and queries," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 2042–2043, 2015.

[3] A. Khan, Y. Ye, L. Chen, and H. Jagadish, *On uncertain graphs*. Springer, 2018.

[4] X. Ke, A. Khan, M. Al Hasan, and R. Rezvansangsari, "Reliability maximization in uncertain graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 2, pp. 894–913, 2020.

[5] J. Tang, L. Fu, S. Liang, F. Long, L. Zhou, X. Wang, and C. Zhou, "Flowercast: Efficient time-sensitive multicast in wireless sensor networks with link uncertainty," *ACM Transactions on Sensor Networks*, vol. 20, no. 1, pp. 1–32, 2023.

[6] S. Yang and F. A. Kuipers, "Traffic uncertainty models in network planning," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 172–177, 2014.

[7] M. Hua and J. Pei, "Probabilistic path queries in road networks: traffic uncertainty aware path selection," in *Proceedings of the 13th International Conference on Extending Database Technology*, pp. 347–358, 2010.

[8] X. Lian and L. Chen, "Trip planner over probabilistic time-dependent road networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 2058–2071, 2013.

[9] B. Y. Chen, W. H. Lam, A. Sumalee, Q. Li, H. Shao, and Z. Fang, "Finding reliable shortest paths in road networks under uncertainty," *Networks and spatial economics*, vol. 13, pp. 123–148, 2013.

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, 2001.

[11] A. Khan, F. Bonchi, F. Gullo, and A. Nufer, "Conditional reliability in uncertain graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 11, pp. 2078–2092, 2018.

[12] A. Saha, R. Brokkelkamp, Y. Velaj, A. Khan, and F. Bonchi, "Shortest paths and centrality in uncertain networks," *Proceedings of the VLDB Endowment*, vol. 14, no. 7, pp. 1188–1201, 2021.

[13] R.-H. Li, Q. Dai, G. Wang, Z. Ming, L. Qin, and J. X. Yu, "Improved algorithms for maximal clique search in uncertain networks," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 1178–1189, IEEE, 2019.

[14] A. P. Mukherjee, P. Xu, and S. Tirthapura, "Mining maximal cliques from an uncertain graph," in *2015 IEEE 31st international conference on data engineering*, pp. 243–254, IEEE, 2015.

[15] J. Niedermayer, A. Züfle, T. Emrich, M. Renz, N. Mamoulis, L. Chen, and H.-P. Kriegel, "Probabilistic nearest neighbor queries on uncertain moving object trajectories," *arXiv preprint arXiv:1305.3407*, 2013.

[16] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios, "K-nearest neighbors in uncertain graphs," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 997–1008, 2010.

[17] J. Wang, A. W.-C. Fu, and J. Cheng, "Rectangle counting in large bipartite graphs," in *2014 IEEE International Congress on Big Data*, pp. 17–24, IEEE, 2014.

[18] Y. Zhang, C. A. Phillips, G. L. Rogers, E. J. Baker, E. J. Chesler, and M. A. Langston, "On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types," *BMC bioinformatics*, vol. 15, pp. 1–18, 2014.

[19] Q. Yu, Y. Du, J. Chen, J. Sui, T. Adalē, G. D. Pearlson, and V. D. Calhoun, "Application of graph theory to assess static and dynamic brain connectivity: Approaches for building brain graphs," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 886–906, 2018.

[20] Q. Yu, J. Chen, Y. Du, J. Sui, E. Damaraju, J. A. Turner, T. G. van Erp, F. Macciardi, A. Belger, J. M. Ford, *et al.*, "A method for building a genome-connectome bipartite graph model," *Journal of neuroscience methods*, vol. 320, pp. 64–71, 2019.

[21] P. Cheng, X. Lian, L. Chen, J. Han, and J. Zhao, "Task assignment on multi-skill oriented spatial crowdsourcing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 8, pp. 2201–2215, 2016.

[22] G. Ye, Y. Zhao, X. Chen, and K. Zheng, "Task allocation with geographic partition in spatial crowdsourcing," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 2404–2413, 2021.

[23] Y. Yang, Y. Cheng, Y. Yang, Y. Yuan, and G. Wang, "Batch-based cooperative task assignment in spatial crowdsourcing," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pp. 1180–1192, IEEE, 2023.

[24] X. Cai, X. Ke, K. Wang, L. Chen, T. Zhang, Q. Liu, and Y. Gao, "Efficient temporal butterfly counting and enumeration on temporal bipartite graphs," *arXiv preprint arXiv:2306.00893*, 2023.

[25] X. Li and H. Chen, "Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach," *Decision Support Systems*, vol. 54, no. 2, pp. 880–890, 2013.

[26] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, no. 1, p. 421425, 2009.

[27] J. L. Herlocker, J. A. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pp. 241–250, 2000.

[28] C. Maier and D. Simovici, "Bipartite graphs and recommendation systems," *Journal of Advances in Information Technology-in print*, 2022.

[29] C. Maier and D. Simovici, "On biclique connectivity in bipartite graphs and recommendation systems," in *Proceedings of the 2021 5th International Conference on Information System and Data Mining*, pp. 151–156, 2021.

[30] J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert, "A new approach to collaborative filtering: Operator estimation with spectral regularization.," *Journal of Machine Learning Research*, vol. 10, no. 3, 2009.

[31] W. Pan, E. Xiang, and Q. Yang, "Transfer learning in collaborative filtering with uncertain ratings," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, pp. 662–668, 2012.

[32] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou, "Recommender systems," *Physics reports*, vol. 519, no. 1, pp. 1–49, 2012.

[33] E. T. Rolls, C.-C. Huang, C.-P. Lin, J. Feng, and M. Joliot, "Automated anatomical labelling atlas 3," *Neuroimage*, vol. 206, p. 116189, 2020.

[34] A. Di Martino, C.-G. Yan, Q. Li, E. Denio, F. X. Castellanos, K. Alaerts, J. S. Anderson, M. Assaf, S. Y. Bookheimer, M. Dapretto, *et al.*, "The autism brain imaging data exchange: towards a large-scale evaluation of the intrinsic brain architecture in autism," *Molecular psychiatry*, vol. 19, no. 6, pp. 659–667, 2014.

[35] N. Metropolis and S. Ulam, "The monte carlo method," *Journal of the American statistical association*, vol. 44, no. 247, pp. 335–341, 1949.

[36] R. M. Karp and M. Luby, "Monte-carlo algorithms for enumeration and reliability problems," in *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pp. 56–64, IEEE Computer Society, 1983.

[37] A. P. Mukherjee, P. Xu, and S. Tirthapura, "Enumeration of maximal cliques from an uncertain graph," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 3, pp. 543–555, 2016.

[38] Q. Dai, R.-H. Li, M. Liao, H. Chen, and G. Wang, "Fast maximal clique enumeration on uncertain graphs: A pivot-based approach," in *Proceedings of the 2022 International Conference on Management of Data*, pp. 2034–2047, 2022.

[39] J. Wang, J. Yang, Z. Ma, C. Zhang, S. Yang, and W. Zhang, "Efficient maximal biclique enumeration on large uncertain bipartite graphs," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[40] X. Li, K. Hao, Z. Yang, X. Cao, and W. Zhang, "Hop-constrained s-t simple path enumeration in large uncertain graphs," in *Australasian Database Conference*, pp. 115–127, Springer, 2022.

[41] A. Zhou, Y. Wang, and L. Chen, "Butterfly counting on uncertain bipartite graphs," *Proceedings of the VLDB Endowment*, vol. 15, no. 2, pp. 211–223, 2021.

[42] A. Zhou, Y. Wang, and L. Chen, "Butterfly counting and bitruss decomposition on uncertain bipartite graphs," *The VLDB Journal*, vol. 32, no. 5, pp. 1013–1036, 2023.

[43] Y. Chen, X. Zhao, X. Lin, Y. Wang, and D. Guo, "Efficient mining of frequent patterns on uncertain graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 2, pp. 287–300, 2018.

[44] C. Ma, R. Cheng, L. V. Lakshmanan, T. Grubenmann, Y. Fang, and X. Li, "Linc: a motif counting algorithm for uncertain graphs," *Proceedings of the VLDB Endowment*, vol. 13, no. 2, pp. 155–168, 2019.

[45] N. H. Tran, K. P. Choi, and L. Zhang, "Counting motifs in the human interactome," *Nature communications*, vol. 4, no. 1, p. 2241, 2013.

[46] A. Todor, A. Dobra, and T. Kahveci, "Counting motifs in probabilistic biological networks," in *Proceedings of the 6th ACM Conference*

on Bioinformatics, Computational Biology and Health Informatics, pp. 116–125, 2015.

[47] A. Saha, X. Ke, A. Khan, and C. Long, "Most probable densest subgraphs," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pp. 1447–1460, IEEE, 2023.

[48] R. M. Karp and M. G. Luby, *A new monte-carlo method for estimating the failure probability of an n-component system*. University of California at Berkeley, 1983.

[49] L. Zou, P. Peng, and D. Zhao, "Top-k possible shortest path query over a large uncertain graph," in *International Conference on Web Information Systems Engineering*, pp. 72–86, Springer, 2011.

[50] K. Wang, X. Lin, L. Qin, W. Zhang, and Y. Zhang, "Vertex priority based butterfly counting for large-scale bipartite networks.," *PVLDB*, 2019.

[51] R. M. Karp, M. Luby, and N. Madras, "Monte-carlo approximation algorithms for enumeration problems," *Journal of algorithms*, vol. 10, no. 3, pp. 429–448, 1989.

[52] S.-V. Sanei-Mehri, A. E. Sariyuce, and S. Tirthapura, "Butterfly counting in bipartite networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2150–2159, 2018.

[53] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[54] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *information retrieval*, vol. 4, pp. 133–151, 2001.