Butterfly Counting in Bipartite Networks

CheolHee Jeong

DM Lab

04.04.2024

Seyed-Vahid Sanei-Mehri, Ahmet Erdem Sariyuce, Srikanta Tirthapura

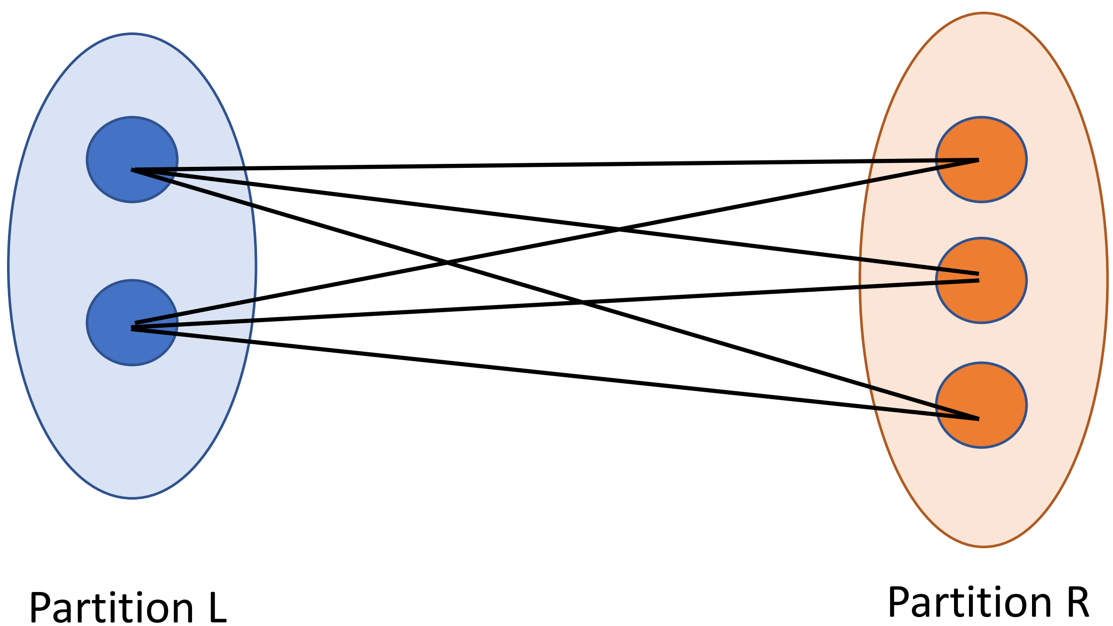
KDD '18

Outline

- Introduction
- Motivation
- Proposed Method
- Conculsion

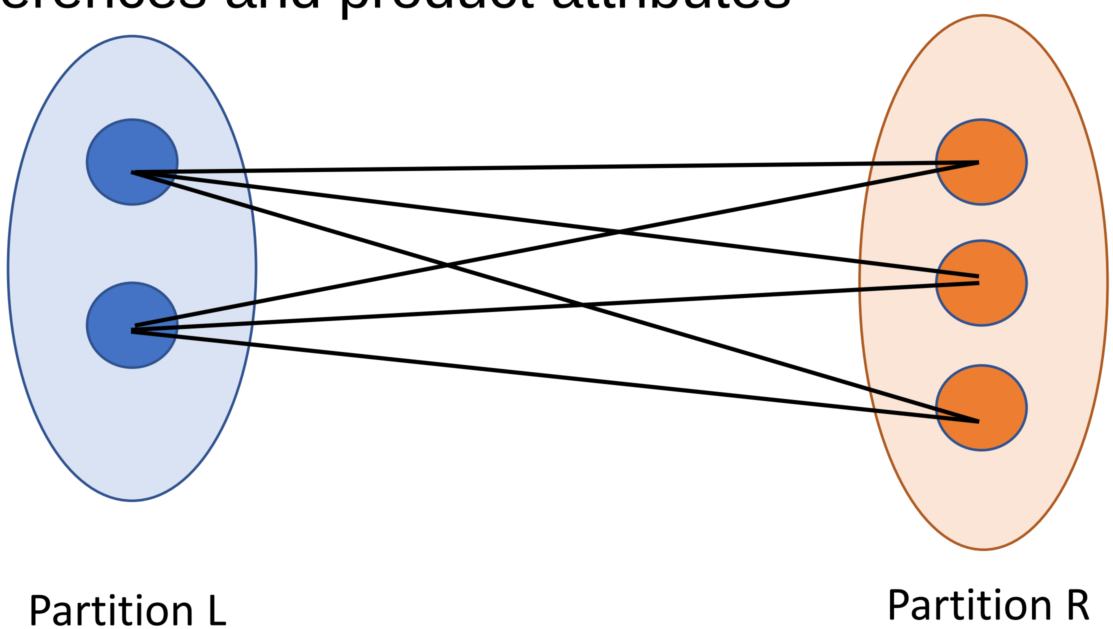
Bipartite Network Definition

- Graph $G = (V, E)$ is called a bipartite graph if:
  - V can be partitioned into two **disjoint** subsets L and R such that $E \subseteq L \times R$, i.e. Every edge connects a vertex in L and a vertex in R



The Importance of Motifs

- **Understanding Network**
  - Butterfly motifs within bipartite graphs are essential for understanding the complexity of networks
- **Data Mining and Recommendation Systems**
  - Utilized for developing more sophisticated recommendation systems by identifying connections between user preferences and product attributes



Outline

- Introduction
- Motivation
- Proposed Method
- Conclusion

Existing Problem

- **기존 모티프 계산 알고리즘은 어떤 문제 점이 있을까?**
  - **계산 복잡도**
    - 기존 모티프 계산 알고리즘은 높은 계산 복잡도를 갖고 있음, 이로 인해 대용량 데이터를 다루는 현대의 응용 분야에서는 실용적이지 못한 경우가 많음
  - **제한된 확장성**
    - 빠르게 증가하는 데이터 크기에 비해, 기존 알고리즘들이 이를 효율적을 처리할 수 있는 확장성 부족

이분 네트워크에서 모티프 계산을 하는데
정확성과 효율성을 동시에 제공하는 방법 필요

Problem Statement

- **Input** : Bipartite graph $G = (V = (L, R), E)$
- **Goal** : To rapidly and accurately compute the number of butterflies in G

Outline

- Introduction
- Motivation
- Proposed Method
- Conclusion

Butterfly Counting

- **Graph 내 나비 숫자 카운팅:**
  - 목표 : 이분 그래프 내 나비 개수 세기
  - 방법 :
    - **ExactBFC** ✓
    - vBFC
    - EBFC

Exact Butterfly Counting

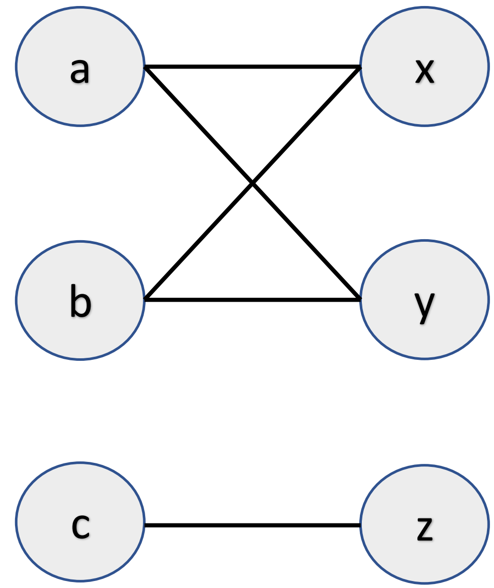
- Exact Butterfly Counting(**ExactBFC**):
  - 목표 : 이분 그래프 내 나비 개수 세기
  - 방법 :
    - 하나의 분할 내 모든 정점 순회
    - 해당 정점을 포함하는 나비 수를 카운트하고 합산
    - 나비를 두 번 세지 않음

Exact Butterfly Counting

Algorithm 1: ExactBFC (V, E): Exact Butterfly Counting

Input : Graph $G = (V = (L, R), E)$
Output: $⧖(G)$

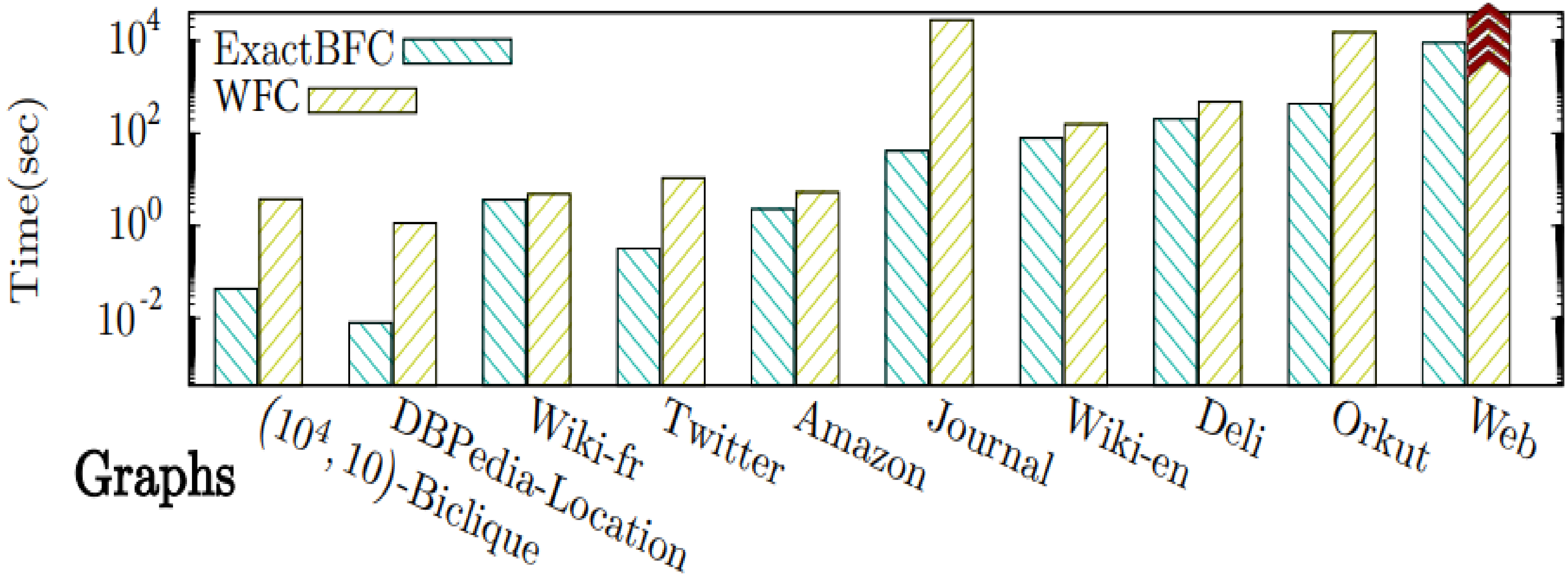
$\mathcal{A} \leftarrow L$, $⧖ \leftarrow 0$
if $\sum_{u \in L} (d_u)^2 < \sum_{v \in R} (d_v)^2$ **then**
  $\mathcal{A} \leftarrow R$
for $v \in \mathcal{A}$ **do**
  $C \leftarrow hashmap$ `// initialized with zero`
  for $u \in \Gamma_v$ **do**
    for $w \in \Gamma_u : w < v$ **do**
      $C[w] \leftarrow C[w] + 1$ `// dist-2 multiplicities`
  for $w \in C : C[w] > 0$ **do**
    $⧖ \leftarrow ⧖ + \binom{C[w]}{2}$
return $⧖/2$ $(⧖)$



Time Complexity of ExactBFC

- **ExactBFC** : $O(\min\{\sum_{v \in L}(d_v)^2 \ , \sum_{v \in R}(d_v)^2 \})$
- **Algorithm by to Wang et** : $O(\sum_{v \in R}(d_v)^2 \ )$

Performance of ExactBFC



L, R 중 어느쪽에서 계산을 진행할지를 선택하기 위한 ExactBFC의 $O(n)$의 사전 처리 단계 효과적

Random Sampling

Counting 알고리즘의 속도를 더 높이고 싶다

- **정말 정확한 나비의 수가 필요할까?**
  - 목표 : 추정치를 이용하여 알고리즘 속도를 향상
  - 방법 :
    - Random Sampling
      - **VSAMP** ✓
      - ESAMP
      - WSAMP

Random Sampling

- **Random Sampling**
  - 샘플링에 의한 근사화 → 나비 수의 추정치 계산
  - 원 그래프보다 작으므로 비용 감소
  - 샘플링 과정은 더 나은 분산을 줄이기 위해 여러 번 반복하여 평균을 내어 사용

Vertex Sampling(VSAMP)

- **Algorithm VSAMP**
  - 무작위로 선택된 한 정점의 거리-2 이웃을 기반으로 샘플링

Algorithm 4: VSAMP (single iteration)

Input: A bipartite graph $G = (V, E)$

Output: An estimate of $⧖(G)$

Choose a vertex v from V uniformly at random.

$⧖_v \leftarrow \text{vBFC}(v, G)$ // Algorithm 2

return $⧖_v \cdot n/4$

VSAMP의 추정치는 비편향, 추정치의 분산은 나비 쌍의 수에 의존

Average time per iteration

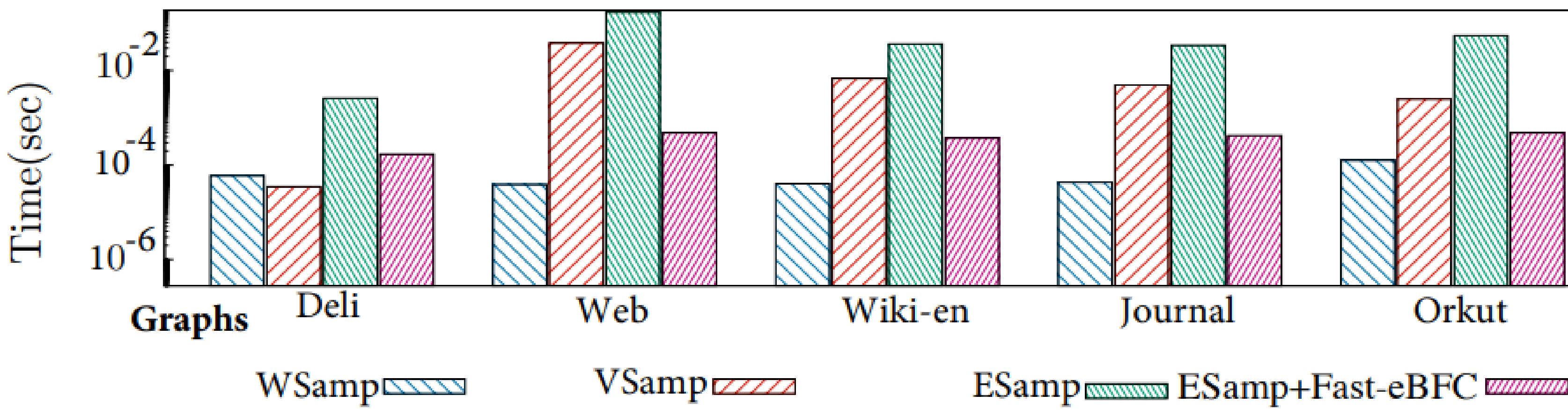


Figure 4: Average time per iteration of sampling algorithms.

WSAMP : 실행 시간이 전반적 가장 짧음
ESAMP : 일관되게 많은 시간 필요함
ESAMP + Fast-eBFC : ESAMP의 시간 크게 단축

Sparsification

- **Sparsification**
  - 그래프를 전역 샘플링 단계를 통해 더 작은 그래프로 축소
  - 각 엣지를 특정 확률로 샘플에 포함
  - 방법 :
    - ESpar
    - ClrSpar

Sparsification

➢ **Espar**

- 각 엣지가 포함될 확률은 다른 엣지와 독립적

Algorithm 8: ESPAR: Edge Sparsification

Input: A bipartite graph $G = (V, E)$, parameter p, $0 < p < 1$

Construct E' by including each edge $e \in E$ independently with probability p

$\beta \leftarrow$ EXACTBFC(V, E') `// Algorithm 1`

return $\beta \cdot p^{-4}$

Sparsification

- **ClrSpar**
  - 특정 구조(밀집된 영역)를 기반으로 확률이 조정
    → **밀집된 영역에 있는 엣지의 확률 높게 설정**

Algorithm 9: ClrSpar

Input: Bipartite graph $G = (V, E)$, number of colors N

Let $f : V \to \{1, \ldots, N\}$ `// map to random colors`

$E' \leftarrow \{(u, v) \in E_G | f(u) = f(v)\}$

$\beta \leftarrow \text{ExactBFC}(V, E')$ `// Algorithm 1`

return $\beta \cdot p^{-3}$ where $p = 1/N$

Sampling or Sparsification

- **Sampling과 Sparsification 중 어떤 것이 좋을까?**

	ESAMP (with FAST-EBFC)	ESPAR
Deli	3.4	2.1
Journal	5.0	1.7
Orkut	3.4	3.4
Web	4.1	3.9
Wiki-en	4.8	2.3

Table 5: Time (in seconds) to obtain 1% relative percent error for the best sampling and sparsification algorithms.

메모리 사용량 : Espar 메모리 사용량 $O(mp)$ Esamp보다 많음
매개변수 설정 : Espar은 샘플링 확률 p를 결정 → 정확도와 실행 시간 사이 trade - off
데이터 접근성 : Espar은 전체 그래프 필요하므로 데이터 접근이 제한적인 환경에서 불리

Outline

- **Introduction**
- **Motivation**
- **Proposed Method**
- **Conclusion**

Conclusion

- **Conclusion:**
  - 간단한 통계를 활용하여 이분 네트워크 내 나비 모티프를 빠르고 정확하게 근사 계산하는 새로운 알고리즘을 제안

- **Strong points**
  - **속도와 정확성** :
    - 나비 모티프의 수를 빠르고 정확하게 추정
  - **정확도 보장** :
    - 정확도에 대한 이론적 알고리즘을 제공
  - **범용성과 적용성** :
    - 다양한 유형의 이분 네트워크 데이터에 적용가능

Conclusion

➢ Weak Points

- 실시간 데이터에 적용 한계
- Vsamp 가정의 명확성 부족 :
  - **독립적 샘플링 가정** :
    - ✓ 실제 네트워크 영역에서는 종속성 있을 수 있음
  - **정점 공유의 균일한 확률 가정** :
    - ✓ 모든 모티프가 그래프 내의 모든 정점과 같은 확률로 연결되어 있다고 가정. 이는 모든 정점이 같은 중요성과 연결성을 가진다는 것을 의미

Thank You

Appendix

Algorithm 2: vBFC (v, G): Per Vertex Butterfly Counting

Input: A vertex $v \in V$ in $G = (V = (L \cup R), E)$

Output: $⧖_v$, number of butterflies in G that contain v

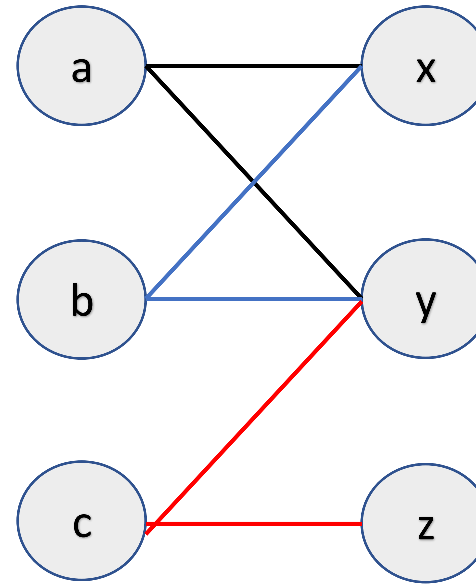
$⧖_v \leftarrow 0, C \leftarrow \mathit{hashmap}$ `// initialized with zero`

for $u \in \Gamma_v$ **do**

  for $w \in \Gamma_u$ **do if** $w \neq v$ **then** $C[w] \leftarrow C[w] + 1$

for $w \in C$ **do** $⧖_v \leftarrow ⧖_v + \binom{C[w]}{2}$

return $⧖_v$



특정 정점을 포함한 나비 수 Counting

Appendix

- **VSAMP 추정치 특성 증명**

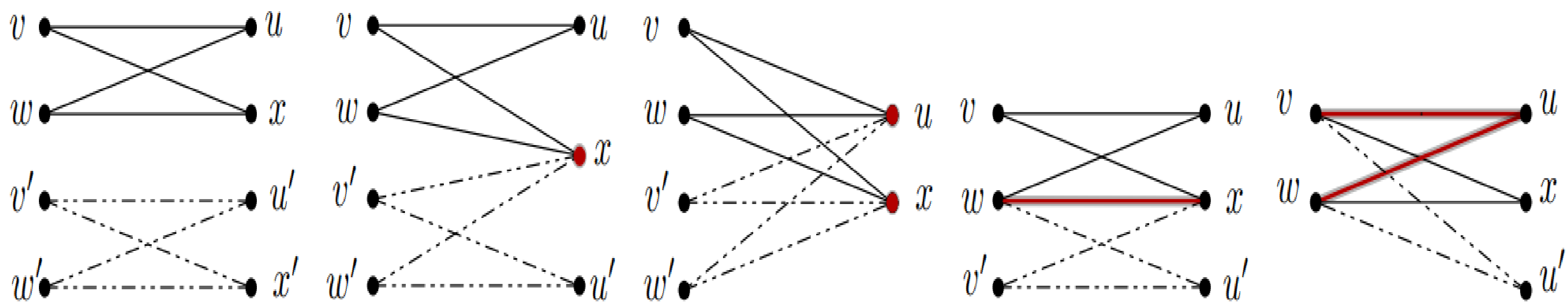
PROOF. Consider that the butterflies in G are numbered from 1 to $\bowtie$. Let $X = \bowtie_v$, the number of butterflies that contain the vertex v, which is sampled uniformly. For $i = 1, \ldots, \bowtie$, let X_i be an indicator random variable equal to 1 if the i^{th} butterfly includes the vertex v. We have $X = \sum_{i=1}^{\bowtie} X_i$. Since each butterfly has four vertices, $\mathbb{E}[X_i] = \Pr[X_i = 1] = 4/n$. Thus, $\mathbb{E}[X] = \sum_{i=1}^{\bowtie} \mathbb{E}[X_i] = \sum_{i=1}^{\bowtie} \Pr[X_i = 1] = \frac{4\bowtie}{n}$. Since $Y_V = X \cdot \frac{n}{4}$, we have $\mathbb{E}[Y_V] = \bowtie$.

$$\mathbb{V}\mathrm{ar}[Y_V] = \mathbb{V}\mathrm{ar}\left[\frac{n}{4}\sum_{i=1}^{\bowtie} X_i\right] = \frac{n^2}{16}\mathbb{V}\mathrm{ar}\left[\sum_{i=1}^{\bowtie} X_i\right]$$

$$= \frac{n^2}{16}\left[\sum_{i=1}^{\bowtie} \mathbb{V}\mathrm{ar}[X_i] + \sum_{i \neq j} \mathbb{C}\mathrm{ov}(X_i, X_j)\right]$$

Appendix

➢ **나비 쌍의 종류**



(a) Type $0v$: No vertex in common

(b) Type $1v$: One vertex in common

(c) Type $2v$: Two vertices in common but no edge

(d) Type $1e$: One edge in common

(e) Type $1w$: One wedge in common

Figure 3: A pair of butterflies in G can be of one of the above five types.