Nucleus Decomposition in Probabilistic Graphs: Hardness and Algorithms

Fatemeh Esfahani, Venkatesh Srinivasan, Alex Thomo, and Kui Wu

***Abstract*—Finding dense components in graphs is of great importance in analysing the structure of networks. Popular frameworks for discovering dense subgraphs are core and truss decompositions. Recently, Sarıyüce et al. introduced nucleus decomposition, which uses r-cliques contained in s-cliques, where $s > r$, as the basis for defining dense subgraphs. Nucleus decomposition can reveal interesting subgraphs that can be missed by core and truss decompositions.**

In this paper, we present *nucleus decomposition in probabilistic graphs*. The major questions we address are: How to define meaningfully nucleus decomposition in probabilistic graphs? How hard is computing nucleus decomposition in probabilistic graphs? Can we devise efficient algorithms for exact or approximate nucleus decomposition in large graphs?

We present three natural definitions of nucleus decomposition in probabilistic graphs: *local*, *global*, and *weakly-global*. We show that the local version is in PTIME, whereas global and weakly-global are #P-hard and NP-hard, respectively. We present an efficient and exact dynamic programming approach for the local case. Further, we present statistical approximations that can scale to bigger datasets without much loss of accuracy. For global and weakly-global decompositions we complement our intractability results by proposing efficient algorithms that give approximate solutions based on search space pruning and Monte-Carlo sampling. Extensive experiments show the scalability and efficiency of our algorithms. Compared to probabilistic core and truss decompositions, nucleus decomposition significantly outperforms in terms of density and clustering metrics.

***Index Terms*—Probabilistic Graphs, Dense Subgraphs, Nucleus Decomposition**

I. Introduction

Probabilistic graphs are graphs where each edge has a probability of existence (cf. [1]–[8]). Many real-world graphs, such as social, trust, and biological networks are associated with intrinsic uncertainty. For instance, in social and trust networks, an edge can be weighted by the probability of influence or trust between two users that the edge connects [9]–[11]. In biological networks of protein-protein interactions (cf. [12]) an edge can be assigned a probability value representing the strength of prediction that a pair of proteins will interact in a living organism [13]–[15].

Mining dense subgraphs and discovering hierarchical relations among them is a fundamental problem in graph analysis tasks. For instance, it can be used for visualizing complex networks [16], finding correlated genes and motifs in biological networks [17], [18], detecting communities in social and web graphs [19], [20], summarizing text [21], and revealing new research subjects in citation networks [22]. Core and truss decompositions are popular tools for finding dense subgraphs. A k-core is a maximal subgraph in which each vertex has at least k neighbors, and a k-truss is a maximal subgraph whose edges are contained in at least k triangles. Core and truss decompositions have been extensively studied for deterministic as well as probabilistic graphs (cf. [1], [23]–[27]).

A recent notion of dense subgraphs is *nucleus* introduced by Sarıyüce et al. [28], [29]. Nucleus decomposition is a generalization of core and truss decompositions that uses higher-order structures to detect dense regions. It can reveal interesting subgraphs that can be missed by core and truss decompositions. In a nutshell, a k-(r, s)-nucleus is a maximal subgraph whose r-cliques are contained in at least k of s-cliques, where $s > r$. For $r = 1, s = 2$ and $r = 2, s = 3$ we obtain the notions of k-core and k-truss, respectively. For $r = 3, s = 4$, r-cliques are *triangles*, s-cliques are 4-*cliques*, and k-$(3, 4)$-nucleus is strictly stronger than k-truss and k-core. Sarıyüce et al. in [28], [29] observed that, in practice, k-$(3, 4)$-nucleus is the most interesting in terms of the quality of subgraphs produced for a large variety of graphs. As such, in this paper we also focus on this decomposition. To the best of our knowledge, *nucleus decomposition over probabilistic graphs* has not been studied yet.

As pointed out by [28], [29], nucleus decomposition can uncover a finer grained structure of dense groups not possible using other dense subgraph mining methods; as such, nucleus decomposition can be beneficial for a large variety of applications, e.g. community structure discovery [30], mining dense regions in internet of things [31], financial fraud detection [32], extracting brain connectome subgraph hierarchy [33], detection of complexes in biological networks [34], etc. All these applications of nucleus decomposition extend naturally to the probabilistic networks.

A. Contributions

We are the first to study nucleus decomposition in probabilistic graphs. The major questions we address are: How to define meaningfully nucleus decomposition in probabilistic graphs? How hard is computing nucleus decomposition in probabilistic graphs? Can we devise efficient algorithms for exact or approximate nucleus decomposition in large graphs?

Definitions. We start by introducing three natural notions of probabilistic nucleus decomposition (Section III). They are based on the concept of *possible worlds* (PW's), which are instantiations of a probabilistic graph obtained by flipping a biased coin for each edge independently, according to its

F. Esfahani,V. Srinivasan, A. Thomo and K. Wu are with the Department of Computer Science, University of Victoria, Victoria, B.C.
E-mail: esfahani,srinivas,thomo,wkui@uvic.ca.

probability. We define *local*, *global*, and *weakly-global* notions of nucleus as a maximal probabilistic subgraph $\mathcal{H}$ satisfying different structural conditions for each case.

In the local case, we require a good number of PW's of $\mathcal{H}$ to satisfy a high level of density around each triangle (in terms of 4-cliques containing it) in $\mathcal{H}$. This is local in nature because the triangles are considered independently of each other. To contrast this, we introduce the global notion, where we request the PW's themselves be deterministic nuclei. This way, not only do we achieve density around each triangle but also ensure the same is achieved for all the triangles of $\mathcal{H}$ simultaneously. Finally, we relax this strict requirement for the weakly-global case by requiring that PW's only contain a deterministic nucleus that includes the triangles of $\mathcal{H}$.

Global and Weakly-Global Cases. We show that computing global and weakly-global decompositions are intractable, namely #P-hard and NP-hard, resp. We complement these results with efficient algorithms for these two cases that give approximate solutions based on search space pruning combined with Monte-Carlo sampling (Section V).

Local Case. We show that local nucleus decomposition is in PTIME (Section IV). The main challenge is to compute the probability of each triangle to be contained in k 4-cliques. We present a dynamic programming (DP) solution for this task, which combined with a triangle peeling approach, solves the problem of local nucleus decomposition efficiently. While this is welcome result, we further propose statistical methods to speed-up the computation. Namely, we provide a framework where well-known distributions, such as Poisson, Normal, and Binomial, can be employed to approximate the DP results. This hybrid approach speeds-up the computation significantly and is able to handle datasets, which DP alone cannot.

Experiments. We present extensive experiments which show that our DP method for local nucleus decomposition is efficient and can handle large datasets; when combined with our statistical approximations, the process is significantly sped-up and can handle much larger datasets. We demonstrate the importance of nucleus decomposition by comparing it to probabilistic core and truss decomposition using density and clustering metrics. Comprehensive use cases show that our notions of nucleus decomposition can detect subgraphs with nice properties which are missed by other notions.

II. Deterministic Nuclei

Let $G = (V, E)$ be an undirected graph, where V is a set of vertices, and E is a set of edges. For a vertex $v \in V$, let $N(v)$ be the set of v's neighbors: $N(v) = \{u : (u, v) \in E\}$. The (deterministic) degree of v in G, is equal to $|N(v)|$.

Nucleus decomposition in deterministic graphs. Nucleus decomposition is a generalization of core and truss decompositions [28], [29]. Each nucleus is a subgraph which contains a dense cluster of cliques. The formal definitions are as follows.

Let r, s with $r < s$ be positive integers. We call cliques of size r, *r-cliques*, and denote them by R, R', etc. Analogously, we call cliques of size s, *s-cliques*, and denote them by S, S'.

Definition 1: The *s-support* of an r-clique R in G, denoted $s\text{-}supp_G(R)$, is the number of s-cliques in G that contain R.

Definition 2: Two r-cliques R and R' in G, are *s-connected*, if there exists a sequence $R = R_1, R_2, \cdots, R_k = R'$ of r-cliques in G such that for each i, there exists some s-clique in G that contains $R_i \cup R_{i+1}$.

Now nucleus decomposition is as follows.

Definition 3: Let k be a positive integer. A ***k*-(r, s)-nucleus** is a maximal subgraph H of G with the following properties.

1) H is a union of s-cliques: every edge in H is part of an s-clique in H.
2) $s\text{-}supp_H(R) \geq k$ for each r-clique R in H.
3) Each pair R, R' of r-cliques in H is s-connected in H.

For simplicity, whenever clear from the context, we will drop the use of prefix s from the definition of support and connectedness.

When $r = 1, s = 2$, r-cliques are nodes, s-cliques are edges, and k-$(1, 2)$-nucleus is the well-known notion of k-core. When $r = 2, s = 3$, r-cliques are edges, s-cliques are triangles, and k-$(2, 3)$-nucleus is the well-known notion of k-truss. [28] shows that k-$(3, 4)$-nucleus, where we consider triangles contained in 4-cliques, provides much more interesting insights compared to k-core and k-truss in terms of density and hierarchical structure. As such, in this paper, we also focus on the $r = 3, s = 4$ case. For simplicity, we will drop using r and s and assume them to be 3 and 4, respectively. In particular, we will refer to k-$(3, 4)$-nucleus as simply k-nucleus.

III. Probabilistic Nuclei

Probabilistic Graphs. A probabilistic graph is a triple $\mathcal{G} = (V, E, p)$, where V and E are as before and $p : E \to (0, 1]$ is a function that maps each edge $e \in E$ to its existence probability p_e. In the most common probabilistic model (cf. [1], [3], [4]), the existence probability of each edge is assumed to be independent of other edges.

In order to analyze probabilistic graphs, we use the concept of *possible worlds* that are deterministic graph instances of $\mathcal{G}$ in which only a subset of edges appears. Conceptually, the possible worlds are obtained by flipping a biased coin for each edge independently, according to its probability. We write $G \sqsubseteq \mathcal{G}$ to say that G is possible world for $\mathcal{G}$. The probability of a possible world $G = (V, E_G) \sqsubseteq \mathcal{G}$ is as follows: $\Pr[G \mid \mathcal{G}] = \prod_{e \in E_G} p_e \prod_{e \in E \setminus E_G} (1 - p_e)$.

We will use $\mathcal{G}$, $\mathcal{G}'$, $\mathcal{H}$, $\mathcal{H}'$ to denote probabilistic graphs.

Nucleus decomposition in probabilistic graphs. We now define three variants of nucleus decomposition in probabilistic graphs which are based on Definitions 4 and 5 we give below. These variants relate to the nature of nucleus and we refer to them as **local (ℓ)**, **global (g)**, and **weakly-global (w)**.

Definition 4: Let $\mathcal{H}$ be a probabilistic graph, $\triangle$ a triangle, and μ a mode in set $\{\ell, \mathbf{g}, \mathbf{w}\}$. Then, $X_{\mathcal{H},\triangle,\mu}$ is a random variable that takes integer values k with tail probability

$$\Pr(X_{\mathcal{H},\triangle,\mu} \geq k) = \sum_{H \sqsubseteq \mathcal{H}} \Pr[H \mid \mathcal{H}] \cdot \mathbb{1}_\mu(H, \triangle, k), \quad (1)$$

where indicator variable $\mathbb{1}_\mu(H, \triangle, k)$ is defined depending on mode μ as follows.

$\mathbb{1}_{\boldsymbol{\ell}}(H, \triangle, k) = 1$ if $\triangle$ is in H, and the support of $\triangle$ in H is at least k.

$\mathbb{1}_{\mathbf{g}}(H, \triangle, k) = 1$ if $\triangle$ is in H, and H is a deterministic k-nucleus.

$\mathbb{1}_{\mathbf{w}}(H, \triangle, k) = 1$ if $\triangle$ is in H, and there is a subgraph H' of H that contains $\triangle$ and is a deterministic k-nucleus.

It is clear that $(\mathbb{1}_{\mathbf{g}}(H, \triangle, k) = 1) \implies (\mathbb{1}_{\mathbf{w}}(H, \triangle, k) = 1) \implies (\mathbb{1}_{\boldsymbol{\ell}}(H, \triangle, k) = 1)$.

In the above definition, $\mathbb{1}_{\boldsymbol{\ell}}(H, \triangle, k)$ has a local quality because a possible world G satisfies its condition if it provides sufficient support to triangle $\triangle$ without considering other triangles in H. On the other hand, $\mathbb{1}_{\mathbf{g}}(H, \triangle, k)$ and $\mathbb{1}_{\mathbf{w}}(H, \triangle, k)$ have a global quality because a possible world H satisfies their conditions only when other triangles in H are considered as well (creating a nucleus together).

In the following, as *preconditions* for cohesiveness, we will assume *cliqueness* and *connectedness* for the nuclei subgraphs we define. Specifically, we will only consider subgraphs $\mathcal{H}$, which, ignoring edge probabilities, are unions of 4-cliques, and where each pair of triangles in $\mathcal{H}$ is connected in $\mathcal{H}$.

Definition 5: Let $\mathcal{G} = (V, E, p)$ be a probabilistic graph. Given threshold $\theta \in [0, 1]$, integer $k \geq 0$, and $\mu \in \{\boldsymbol{\ell}, \mathbf{g}, \mathbf{w}\}$, a μ-(k, θ)-nucleus $\mathcal{H}$ is a maximal subgraph of $\mathcal{G}$, such that $\Pr(X_{\mathcal{H},\triangle,\mu} \geq k) \geq \theta$ for each triangle $\triangle$ in $\mathcal{H}$.

Moreover, the μ-(k, θ)- *nucleusness* (or simply nucleusness when μ, k, and θ are clear from context) of a triangle $\triangle$ is the largest value of k such that $\triangle$ is contained in a μ-(k, θ)-nucleus.

Intuitively for $\mu = \boldsymbol{\ell}$, from a probabilistic perspective, a subgraph $\mathcal{H}$ of $\mathcal{G}$ can be regarded as a cohesive subgraph of $\mathcal{G}$ if the support of every triangle in $\mathcal{H}$ is no less than k with high probability (no less than a threshold θ). We call this version local nucleus.

Local nucleus is a nice concept for probabilistic subgraph cohesiveness, however, it has the following shortcoming. While it ensures that every triangle $\triangle$ in $\mathcal{H}$ has support at least k in a good number of instantiations of $\mathcal{H}$, it does not ensure those instantiations are deterministic nuclei themselves or they contain some nucleus which in turn contains $\triangle$. Obviously, nucleusness is a desirable property to ask for in order to achieve a higher degree of cohesiveness and this leads to the other two versions of probabilistic nucleus of a global nature, which we call global and weakly-global (obtained for $\mu = \mathbf{g}$ and $\mu = \mathbf{w}$).

In general, $\mathbf{g}$-(k, θ)-nuclei are smaller and more cohesive than $\mathbf{w}$-(k, θ)-nuclei. We remark that, every $\mathbf{g}$-(k, θ)-nucleus is contained in a $\mathbf{w}$-(k, θ)-nucleus which in turn is contained in an $\boldsymbol{\ell}$-(k, θ)-nucleus. In the full version [35], we give a detailed example illustrating the three nucleus notions we introduce.

Nucleus Decomposition. The *nucleus decomposition* finds the set of all the μ-(k, θ)-nuclei for different values of k. We will study the problem in the three different modes we consider. Specifically, we call nucleus-decomposition problems for the different modes $\boldsymbol{\ell}$-*NuDecomp*, $\mathbf{g}$-*NuDecomp*, and $\mathbf{w}$-*NuDecomp*, respectively.

We show that $\boldsymbol{\ell}$-*NuDecomp* can be computed in polynomial time and furthermore we give several algorithms to achieve efficiency for large graphs. In the full version [35], we show that $\mathbf{g}$-*NuDecomp* and $\mathbf{w}$-*NuDecomp* are $\#P$-hard and NP-hard, respectively. Nevertheless, as we show later in this paper, once we obtain the $\boldsymbol{\ell}$-*NuDecomp*, we can use it as basis, combined with sampling techniques, to effectively approximate $\mathbf{g}$-*NuDecomp* and $\mathbf{w}$-*NuDecomp*.

IV. Local Nucleus decomposition

Here we propose efficient algorithms for solving $\boldsymbol{\ell}$-*NuDecomp*. Peeling is a general strategy that has been used broadly in core and truss decompositions as well as in deterministic nucleus decomposition [28]. However, generalizing peeling to compute $\boldsymbol{\ell}$-*NuDecomp* creates significant computational challenges. For example, a challenge is finding the support score for each triangle. This is because of the combinatorial nature of finding the maximum value of k such that $\Pr(X_{\mathcal{G},\triangle,\boldsymbol{\ell}} \geq k) \geq \theta$ for a triangle $\triangle$. In particular, triangle $\triangle$ in a probabilistic graph can be part of different numbers of 4 cliques with different probabilities. As a result, considering all the subsets of 4-cliques which contain $\triangle$ results in exponential time complexity. In our algorithm, we identify two challenging tasks, namely computing and updating nucleus scores.

A. Computing initial nucleus scores

Our process starts by computing a nucleus score $\kappa_\triangle$ for each triangle $\triangle$, which initially is the maximum k for which $\Pr(X_{\mathcal{G},\triangle,\boldsymbol{\ell}} \geq k) \geq \theta$.

Given a probabilistic graph $\mathcal{G} = (V, E, p)$, let $\triangle = (u, v, w)$ be a triangle in $\mathcal{G}$. For $i = 1, \ldots, c_\triangle$, where $c_\triangle = |N(u) \cap N(v) \cap N(w)|$, let $z_i \in N(u) \cap N(v) \cap N(w)$ and $S_i = \{u, v, w, z_i\}$. In other words, for each i, S_i is the set of vertices of a 4-clique that contains $\triangle$. For notational simplicity, we will also denote by S_i the 4-clique on $\{u, v, w, z_i\}$.

Similarly, for each i, let $\mathcal{E}_i = \{(u, z_i), (v, z_i), (w, z_i)\}$ be the set of edges which connect vertex z_i to vertices of $\triangle$. Let $\Pr(\mathcal{E}_i) = p(u, z_i) \cdot p(v, z_i) \cdot p(w, z_i)$ be the existence probability of $\mathcal{E}_i$. We have:

$$\Pr(X_{\mathcal{G},\triangle,\boldsymbol{\ell}} \geq k) = \Pr(X_{\mathcal{G},\triangle,\boldsymbol{\ell}} \geq k-1) - \Pr(X_{\mathcal{G},\triangle,\boldsymbol{\ell}} = k-1) \quad (2)$$

Thus, we need to compute $\Pr(X_{\mathcal{G},\triangle,\boldsymbol{\ell}} = k)$ for any k, and find the maximum value of k for which the probability on the left-hand side of Equation 2 is greater than or equal to θ. In fact, $\Pr(X_{\mathcal{G},\triangle,\boldsymbol{\ell}} = k)$ gives the probability that $\triangle$ is contained in k number of 4-cliques in $\mathcal{G}$. Under the condition that $\triangle$ exists, we denote $\mathcal{X}(S_\triangle, k, j)$ to be the probability that $\triangle$ is contained in k of 4-cliques from $\{S_1, \cdots, S_j\} \subseteq \mathcal{S}_\triangle$, where $\mathcal{S}_\triangle$ the set of 4-cliques containing $\triangle$ in $\mathcal{G}$. In other words, $\mathcal{X}(S_\triangle, k, j)$ is a conditional probability (conditioning on the existence of $\triangle$).

We fix an arbitrary order on $\mathcal{S}_\triangle$. The event that $\triangle$ is contained in k of 4-cliques from $\{S_1, \cdots, S_j\}$, can be expressed as the union of the following two sub-events: **(1)** the event that

the 4-clique S_j exists and $\triangle$ is contained in $(k-1)$ of 4-cliques from $\{S_1, \cdots, S_{j-1}\}$, and **(2)** the event that the S_j does not exist and $\triangle$ is part of k of 4-cliques from $\{S_1, \cdots, S_{j-1}\}$. Thus, we have the following recursive formula:

$$\mathcal{X}(\mathcal{S}_\triangle, k, j) = \Pr(\mathcal{E}_j) \cdot \mathcal{X}(\mathcal{S}_\triangle, k-1, j-1) + (1 - \Pr(\mathcal{E}_j)) \cdot \mathcal{X}(\mathcal{S}_\triangle, k, j-1), \quad (3)$$

where $k \in [0, c_\triangle]$, and $j \in [0, c_\triangle]$. Initially, we set $\mathcal{X}(\mathcal{S}_\triangle, 0, 0) = 1$, $\mathcal{X}(\mathcal{S}_\triangle, -1, j) = 0$ for any j, and $\mathcal{X}(\mathcal{S}_\triangle, k, j) = 0$, if $k > j$. Setting $j = c_\triangle$ in Equation 3, and multiplying $\mathcal{X}(\mathcal{S}_\triangle, k, j)$ by $\Pr(\triangle)$ (existence probability of $\triangle$), gives the desired probability $\Pr(X_{\mathcal{G},\triangle,\boldsymbol{\ell}} = k)$. Thus, $\Pr(X_{\mathcal{G},\triangle,\boldsymbol{\ell}} = k) = \Pr(\triangle) \cdot \mathcal{X}(S_\triangle, k, c_\triangle)$.

Given a triangle $\triangle$, let the *neighbor triangles* of $\triangle$ be those triangles which form a 4-clique with $\triangle$. In the following we show how we can update $\Pr(X_{\mathcal{G},\triangle,\boldsymbol{\ell}} \geq k)$ when a neighbor triangle is processed in the decomposition.

B. Updating nucleus scores

Once the κ scores have been initialized as described above, a process of peeling "removes" the triangle $\triangle^*$ of the lowest κ-score, specifically marks it as *processed*, and updates the neighboring triangles $\triangle$ (those contained in the same 4-cliques as the removed triangle) in terms of $\Pr(X_{\mathcal{G},\triangle,\boldsymbol{\ell}} \geq k)$. Because of the removal of $\triangle^*$ the cliques containing it cease to exist, thus $\Pr(X_{\mathcal{G},\triangle,\boldsymbol{\ell}} \geq k)$ of the neighbors $\triangle$ will change. We recompute this probability using the formula in Equation 3, where the sets of cliques $\mathcal{S}_\triangle$ are updated to remove the cliques containing $\triangle^*$.

Algorithm 1 *$\boldsymbol{\ell}$-NuDecomp*

```
1:  function ℓ-NUCLEUSNESS(G, θ)
2:      for all triangles △ ∈ G do
3:          κ(△) ← arg max_k{X(S_△, k, c_△) ≥ θ}
4:          processed[△] ← false
5:      for all unprocessed △ ∈ G with minimum κ(△) do
6:          ν(△) ← κ(△)
7:          Find set S_△ of 4-cliques containing △
8:          for all S ∈ S_△ with non-processed triangles do
9:              for all △' ⊂ S, △' ≠ △, κ(△') > κ(△) do
10:                 κ(△') ← arg max_k{X(S_△'\S, k, c_△'−1) ≥ θ}
11:         processed[△] ← true
12:     return array ν(·)
```

Algorithm 1 computes the nucleusness of each triangle in $\mathcal{G}$. In line 3, for each triangle $\triangle$, $\kappa(\triangle)$ is initialized using Equation 3. Array *processed* records whether a triangle has been processed or not in the algorithm (line 4). At each iteration (line 5-11), an unprocessed triangle $\triangle$ with minimum $\kappa(\triangle)$ is considered, and its nucleus score is set and stored in array ν (line 6). Then, the $\kappa(\triangle')$ values of all the neighboring triangles $\triangle'$ are updated using Equation 3. The affected triangles are those unprocessed triangles which are part of the same 4-clique with triangle $\triangle$. The algorithm continues until all the triangles are processed. At the end, each triangle obtains its nucleus score and array ν with these scores is returned (line 12). Once all the nucleus scores are obtained, we build $\boldsymbol{\ell}$-(k,θ)-nuclei for each value of k.

Observe that the κ values for each triangle at each iteration decrease or stay the same. This implies that κ for each triangle $\triangle$ is a monotonic property function similar to properties described in [36] for vertices. Now, we can use a reasoning similar to the one in [36] to show that our algorithm, which repeatedly removes a triangle with the smallest κ value, gives the correct nucleusness for each triangle.

We show in the full version [35] that *$\boldsymbol{\ell}$-NuDecomp* can be computed in polynomial time and that its space complexity is $O(T_\mathcal{G})$. This is the same as the space complexity of deterministic nucleus decomposition.

While being able to compute *$\boldsymbol{\ell}$-NuDecomp* in polynomial time is good news, finding the maximum k such that $\Pr(X_{\mathcal{G},\triangle,l} \geq k) \geq \theta$ is quadratic in $c_\triangle$ which is not efficient for large probabilistic graphs. As an alternative approach, we will now propose efficient methods to approximate $\Pr(X_{\mathcal{G},\triangle,l} \geq k)$ in $O(c_\triangle)$ time such that the results are practically distinguishable from the exact values. The approximation is based on limit theorems, such as Le Cam's Poisson Limit Theorem [37] and Lyapunov's Central Limit Theorem [38].

C. Approximating κ scores

Framework. Given a triangle $\triangle = (u, v, w)$, let $S_i = \{u, v, w, z_i\}$ for $i = 1, \ldots, c_\triangle$, as before. Also, let $\mathcal{E}_i = \{(u, z_i), (v, z_i), (w, z_i)\}$ be the edges that connect z_i to the vertices of $\triangle$.

With slight abuse of notation, we also define each $\mathcal{E}_i$ as an indicator random variable which takes on 1, if all the edges in $\mathcal{E}_i$ exist, and takes on 0, if at least one of the edges in the set does not exist. We observe that the variables $\mathcal{E}_i$ are mutually independent since the sets $\mathcal{E}_i$ do not share any edge. Also, each Bernoulli variable $\mathcal{E}_i$ takes value 1 with probability $p(u, z_i) \cdot p(v, z_i) \cdot p(w, z_i)$ and 0 with $1 - (p(u, z_i) \cdot p(v, z_i)) \cdot p(w, z_i))$.

Let $\zeta = \sum_{i=1}^{c_\triangle} \mathcal{E}_i$. We can verify the following proposition.

Proposition 1: $\Pr(X_{\mathcal{G},\triangle,\boldsymbol{\ell}} \geq k) = \Pr(\triangle) \cdot \Pr[\zeta \geq k]$.

The expectation and variance of ζ are $\mu = \sum_{i=1}^{c_\triangle} \Pr(\mathcal{E}_i)$ and $\sigma^2 = \sum_{i=1}^{c_\triangle} \big(\Pr(\mathcal{E}_i) \cdot (1 - \Pr(\mathcal{E}_i)\big)$, respectively. Now we show that we can approximate the distribution of ζ using Le Cam's Theorem which makes use of Poisson Distribution [37].

Poisson Distribution [39]: A discrete random variable X is said to have Poisson distribution with positive parameter λ, if the probability mass function of X is given by:

$$\Pr[X = k] = \frac{\lambda^k e^{-k}}{k!}, \quad k = 0, 1, \cdots, \quad (4)$$

The expected value of a Poisson random variable is λ. Setting λ to μ, we can approximate the distribution of ζ by

the Poisson distribution. Using Le Cam's Theorem [37], the error bound on the approximation is as follows:

$$\sum_{k=0}^{c_\triangle} \left| \Pr(\zeta = k) - \frac{\lambda^k e^{-\lambda}}{k!} \right| < 2 \sum_{i=1}^{c_\triangle} \left(\Pr(\mathcal{E}_i)\right)^2 = 2(\mu - \sigma^2). \tag{5}$$

Equation 5 shows that the Poisson distribution is reliable if $\Pr(\mathcal{E}_i)$ and $c_\triangle$ are small.

In some applications, $\sum_{i=1}^{c_\triangle} \left(\Pr(\mathcal{E}_i)\right)^2$ in Equation 5 can be large, even if each $\Pr(\mathcal{E}_i)$ is small. As a result, the difference between the variance $\sigma^2 = \sum_{i=1}^{c_\triangle} \Pr(\mathcal{E}_i) - \sum_{i=1}^{c_\triangle} \left(\Pr(\mathcal{E}_i)\right)^2$ of ζ, and the variance $\lambda = \sum_{i=1}^{c_\triangle} \Pr(\mathcal{E}_i)$ of the Poisson approximation becomes large. To tackle the problem, we define a Translated Poisson [40] random variable $Y = \lfloor \lambda_2 \rfloor + \Pi_{\lambda - \lfloor \lambda_2 \rfloor}$, where $\lambda_2 = \lambda - \sigma^2$ and Π is Poisson distribution with parameter $\lambda - \lfloor \lambda_2 \rfloor$. In this formula $\lambda = \sum_{i=1}^{c_\triangle} \Pr(\mathcal{E}_i)$ is the expected value of distribution ζ. Thus, the difference between the variance of Y and ζ can be written as:

$$\begin{aligned} &\text{Var(Y)} - \text{Var}(\zeta) = \lambda - \lfloor \lambda_2 \rfloor - \sigma^2 = \lambda - \sigma^2 - \lfloor \lambda_2 \rfloor, \\ &= \lambda_2 - (\lambda_2 - \{\lambda_2\}) = \{\lambda_2\} < 1, \end{aligned} \tag{6}$$

where $\{\lambda_2\} = \lambda_2 - \lfloor \lambda_2 \rfloor$. As can be seen the difference between the variances becomes small in this case.

We will now consider the scenario when $c_\triangle$ is large. In this case, the variance of ζ will be large. In the following, we show the use of Central Limit Theorem for this case.

Central Limit Theorem. An important theorem in statistics, Lyapunov's Central Limit Theorem (CLT) [38] states that, given a set of random variables (not necessarily i.i.d.), their properly scaled sum converges to a normal distribution under certain conditions.

If $c_\triangle$ and hence σ^2 are large, then by [38], $Z = \frac{1}{\sigma} \sum_{i=1}^{c_\triangle} (\mathcal{E}_i - \mu_i)$ has standard normal distribution, where $\mu_i = \Pr(\mathcal{E}_i)$. To approximate $\Pr[\zeta \geq k] = \Pr[\sum_{i=1}^{c_\triangle} \mathcal{E}_i \geq k]$ using CLT we can subtract $\sum_{i=1}^{c_\triangle} \mu_i$ from the sum of $\mathcal{E}_i$'s and divide by σ. As a result, we have:

$$\Pr\left[\sum_{i=1}^{c_\triangle} \mathcal{E}_i \geq k\right] = \Pr\left[\frac{1}{\sigma} \sum_{i=1}^{c_\triangle} (\mathcal{E}_i - \mu_i) \geq \frac{1}{\sigma} \left(k - \sum_{i=1}^{c_\triangle} \mu_i\right)\right] \tag{7}$$

Since $Z = \frac{1}{\sigma} \sum_{i=1}^{c_\triangle} (\mathcal{E}_i - \mu_i)$ has standard normal distribution, we can find the maximum value of k such that the right-hand side of Equation 7 is at equal or greater than the threshold. Evaluation of each probability can be done in constant time. Thus, finding the maximum value of k can be done in $O(c_\triangle)$ time.

Binomial Distribution. In many networks, edge probabilities are close to each other and as a result, for each triangle $\triangle$, $\Pr(\mathcal{E}_i)$'s are also close to each other. In that case, the distribution of support of the triangle $\triangle$ can be well approximated by Binomial distribution. A random variable X is said to have Binomial distribution with parameters p and n, if the probability mass function of X is given by [41]:

$$\Pr[X = k] = \binom{n}{k} p^k (1-p)^{(n-k)}. \tag{8}$$

In the above equation, p is success probability, and n is the number of experiments. In statistics, the sum of non-identically distributed and independent Bernoulli random variables can be approximated by the Binomial distribution [42]. As discussed in [42], the Binomial distribution provides a good approximation, if its variance is close to the variance of ζ. For the approximation, we set $n = c_\triangle$ and $n \cdot p = \mu$.

Summary. We compute $\Pr(X_{\mathcal{G},\triangle,\boldsymbol{\ell}} \geq k)$ using the following set of conditions based on four thresholds A, B, C, D.

1) If $c_\triangle$ is large ($c_\triangle \geq A$), the *CLT* approximation is used.
2) If (1) does not hold, then if $c_\triangle$ and $\Pr(\mathcal{E}_i)$'s are small ($c_\triangle < B$ and $\Pr(\mathcal{E}_i)'s < C$), the *Poisson* approximation is used.
3) If (1) and (2) do not hold, then if $\sum_{i=1}^{c_\triangle} \left(\Pr(\mathcal{E}_i)\right)^2 > 1$, the *Translated Poisson* approximation is used.
4) If (1), (2), and (3) do not hold, then if the ratio of the variance of ζ to the variance of the Binomial distribution with $n = c_\triangle$ and $p = \mu/n$ is close to 1 (e.g. not less than a number D), the *Binomial* approximation is used.
5) Otherwise, *Dynamic Programming* is used.

For selecting the thresholds we refer to the literature in statistics. In particular, CLT gives a good approximation if the number (for our problem $c_\triangle$) of random variables in the sum is at least 30 ([43], p. 547). In fact, we set our threshold $A = 200$ to much higher than what is suggested by the literature. Also, regarding Poisson distribution, the existence probability (for our problem $\Pr(\mathcal{E}_i)$'s) of the indicator random variables in the sum should be less than 0.25 (see [37]). So, we set $C = 0.25$. We set B to be half of A so that it is considerably far from A (threshold on $c_\triangle$). We set $D = 0.9$ which is close enough to 1.

When using $A = 200$, $B = 100$, $C = 0.25$, $D = 0.9$, we observed that the results of computing $\Pr(X_{\mathcal{G},\triangle,\boldsymbol{\ell}} \geq k)$ using an approximation are practically indistinguishable from the solution of dynamic programming. Furthermore, as we observed in our experiments, falling back to dynamic programming in point (5) happens only in a small amount of cases, i.e. most triangles in real world networks satisfy one of the earlier conditions (1)-(4). This means we can avoid dynamic programming for most of the triangles.

V. Global and Weakly-Global Nucleus

In this section, we propose algorithms for computing global and weakly-global nucleus decomposition. Given a graph $\mathcal{H}$, computing $\Pr(X_{\mathcal{H},\triangle,\mathbf{g}} \geq k)$ and $\Pr(X_{\mathcal{H},\triangle,\mathbf{w}} \geq k)$ requires finding all the possible worlds of $\mathcal{H}$, which are in total $2^{|E(\mathcal{H})|}$, where $E(\mathcal{H})$ is the number of edges in $\mathcal{H}$. This is prohibitive. Thus, we use Monte Carlo sampling to estimate the probabilities, denoted by $\hat{\Pr}(X_{\mathcal{H},\triangle,\mathbf{g}} \geq k)$ and $\hat{\Pr}(X_{\mathcal{H},\triangle,\mathbf{w}} \geq k)$. The following lemma states a special version of the Hoeffding's inequality [44] that provides the minimum number of samples required to obtain an unbiased estimate.

Lemma 1: Let $Y_1, \cdots, Y_n$ be independent random variables bounded in the interval $[0, 1]$. Also, let $\bar{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i$. Then, we have that

$$\Pr\left[|\bar{Y} - \mathbb{E}[\bar{Y}]| \geq \epsilon\right] \leq 2e^{-2n\epsilon^2}. \tag{9}$$

In other words, for any $\epsilon, \delta \in (0,1]$, $\Pr\left[|\bar{Y} - \mathbb{E}[\bar{Y}]| \geq \epsilon\right] \leq \delta$, if $n \geq \left\lceil \frac{1}{2\epsilon^2} \ln\left(\frac{2}{\delta}\right)\right\rceil$.

Based on the above, using Monte Carlo sampling, we can obtain an estimate of $\Pr(X_{\mathcal{H},\triangle,\mathbf{g}} \geq k)$, and $\Pr(X_{\mathcal{H},\triangle,\mathbf{w}} \geq k)$ for any subgraph $\mathcal{H}$ by sampling n possible worlds of $\mathcal{H}$, $\{H_1, \cdots, H_n\}$, where $n = \left\lceil \frac{1}{2\epsilon^2} \ln\left(\frac{2}{\delta}\right)\right\rceil$, ϵ is an error bound, and δ is a probability guarantee. In particular, we have:

$$\hat{\Pr}(X_{\mathcal{H},\triangle,\mu} \geq k) = \sum_{i=1}^{n} \mathbb{1}_{\mu}(H_i, \triangle, k)/n, \tag{10}$$

where $\mu = \mathbf{g}$ or $\mathbf{w}$, and the indicator function $\mathbb{1}_{\mu}(H_i, \triangle, k)$ is given in Definition 4. Based on Lemma 1, what we obtain is an unbiased estimate. Thus, setting $\mu = \mathbf{g}, \mathbf{w}$, we have

$$\Pr\left[\left|\Pr\left(X_{\mathcal{H},\triangle,\mu} \geq k\right) - \hat{\Pr}\left(X_{\mathcal{H},\triangle,\mu} \geq k\right)\right| \geq \epsilon\right] \leq \delta. \tag{11}$$

g-(k,θ)-nucleus. In what follows, we propose an algorithm for finding all **g**-(k,θ)-nuclei for different values of $k = 1, \ldots, k_{\max}$, where $k_{\max}$ is the largest value for which the local nucleus is non-empty. This is because we extract global nuclei from local ones since every **g**-(k,θ)-nucleus is part of an ℓ-(k,θ)-nucleus. The main steps of our proposed algorithm are given in Algorithm 2.

Given a positive integer k, threshold θ, error-bound ϵ, and confidence level δ, the algorithm starts by creating subgraph $\mathcal{C}_k$ as the union of all ℓ-(k,θ)-nuclei (line 4). Then, the algorithm incrementally builds a candidate **g**-(k,θ)-nucleus $\mathcal{H}$ as follows. For each triangle $\triangle$ in $\mathcal{C}_k$, it adds to $\mathcal{H}$ all the 4-cliques in $\mathcal{C}_k$ containing $\triangle$ (line 6). By this process other triangles $\triangle'$ could potentially be added to $\mathcal{H}$ such that the number of 4-cliques containing $\triangle'$ is less than k. In order to remedy this, the algorithm adds all the 4-cliques of $\mathcal{C}_k$ containing $\triangle'$ to $\mathcal{H}$. This process continues until all the triangles in $\mathcal{H}$ are contained in at least k 4-cliques (lines 7-8). Once the candidate graph $\mathcal{H}$ is obtained, n samples of possible worlds of $\mathcal{H}$ are obtained (line 10). Then, the algorithm checks if the condition $\hat{\Pr}(X_{\mathcal{H},\triangle,\mathbf{g}} \geq k) \geq \theta$ is satisfied for each triangle $\triangle$ in $\mathcal{H}$ (lines 11-13). At the end, the algorithm returns all **g**-(k,θ)-nuclei $\mathcal{H}$ (line 15-17), for all the possible values of k.

w-(k,θ)-nucleus. In what follows, we propose an algorithm for finding all w-(k,θ)-nuclei, for different values of $k = 1, \ldots, k_{\max}$, where $k_{\max}$ is as before. We begin by noting that each **w**-(k,θ)-nucleus is an ℓ-(k,θ)-nucleus. The steps of our proposed algorithm are given in Algorithm 3.

We use array *global_score* to store the number of deterministic k-nuclei that each triangle belongs to. The array is initialized to zero for all the triangles in the candidate graph (line 5). For each candidate graph which is a ℓ-(k,θ)-nucleus, we obtain the required number n of possible worlds for the given ϵ and δ. Then, we perform deterministic nucleus decomposition on each world (lines 6-8). If triangle $\triangle$ is in a deterministic k-nucleus of that possible world, the corresponding index of $\triangle$ in array *global_score* is incremented by one (lines 9-10). In line 12, the approximate value $\hat{\Pr}(X_{\mathcal{H},\triangle,\mathbf{w}} \geq k)$ is obtained for each triangle. Then, we start creating the connected components $\mathcal{H}$ using triangles with $\hat{\Pr}(X_{\mathcal{H},\triangle,\mathbf{w}} \geq k) \geq \theta$ (line 13). At the end, the algorithm returns all **w**-(k,θ)-nuclei, for all the possible values of k.

Remark. Both of these algorithms run in polynomial time. They compute the correct answer provided the estimation of the threshold probabilities using Monte-Carlo sampling is close to the true value. If not, they give approximate solutions.

Space Complexity. For global and weakly global decompositions the space needed is $O(T_{\mathcal{G}} + m \cdot n)$, where m is the number of edges in $\mathcal{H}$ and n is the number of possible worlds for $\mathcal{H}$ we sample.

From a theoretical point of view, n, the number of samples, is constant for fixed values of ϵ and δ, and since m, number of edges, is absorbed by $T_{\mathcal{G}}$, we can say that the above

Algorithm 2 g-*NuDecomp*

```
1: function g_NUCLEUS(G, θ, ϵ, δ)
2:     solution ← {}
3:     for all k ← 1 to k_max do
4:         C_k ← the union of ℓ-(k,θ)-nuclei by Algorithm 1
5:         for all △ ∈ C_k do
6:             H ← all 4-cliques in C_k containing △
7:             while ∃△' ∈ H with less than k 4-cliques ∈ H
               containing it do
8:                 add all 4-cliques of C_k containing △' to H
9:             condition_hold ← true
10:            sample ← {H_1, ..., H_n}
11:            for all △ ∈ H do Pr^(X_{H,△,g} ≥ k) ← Eq.(10)
12:                if Pr^(X_{H,△,g} ≥ k) < θ then
13:                    condition_hold ← false
14:                    break
15:            if condition_hold == true then
16:                solution ← solution ∪ H
17:    return solution
```

Algorithm 3 w-*NuDecomp*

```
1: function w_NUCLEUS(G, θ, ϵ, δ)
2:     solution ← {}
3:     for all k ← 1 to k_max do
4:         for all ℓ-(k,θ) H do
5:             global_score[△] ← 0 for each △ ∈ H
6:             sample ← {H_1, ..., H_n}
7:             for all H ∈ sample do
8:                 H' ← k-nucleus of H
9:                 for all triangle △ ∈ H' do
10:                    global_score[△] ++
11:            for all △ ∈ H do
12:                Pr^(X_{H,△,w} ≥ k) ← global_score[△]/n
13:            solution ← solution ∪ connected union of △'s
               with Pr^(X_{H,△,w} ≥ k) ≥ θ
14:    return solution
```

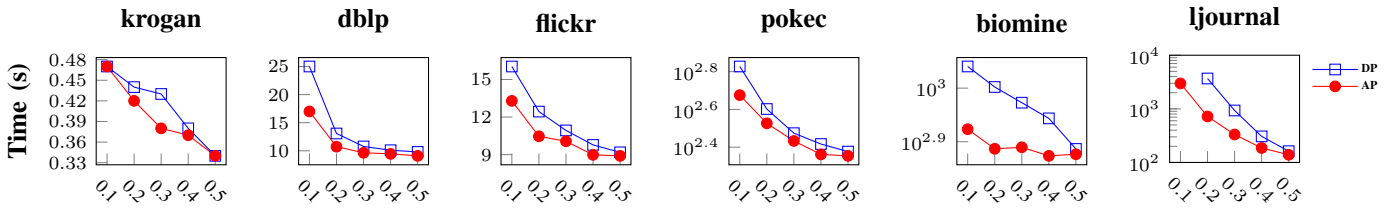


Fig. 1: Run times of DP and AP for varying θ (x axis). Both perform well on medium datasets. For bigger datasets, biomine and ljournal, the difference is more pronounced. For ljournal, for $\theta = 0.1$, it is only AP that can complete.

Graph	$\|V\|$	$\|E\|$	d_{max}	p_{avg}	$\|\triangle\|$
krogan	2,708	7,123	141	0.68	6,968
dblp	684,911	2,284,991	611	0.26	4,582,169
flickr	24,125	300,836	546	0.13	8,857,038
pokec	1,632,803	22,301,964	14,854	0.50	32,557,458
biomine	1,008,201	6,722,503	139,624	0.27	93,716,868
ljournal-2008	5,363,260	49,514,271	19,432	0.50	411,155,444

TABLE I: Dataset Statistics

complexity is again $O(T_{\mathcal{G}})$, i.e. same as the space complexity for deterministic nucleus.

From a practical point of view, for each sample graph (possible world) we use a bit array to record whether an edge exists in the sample or not. For practical values of ϵ and δ, $m \cdot n$ is about $200 \cdot m$ bits, which is $200/(32+32) \sim 3$ times more than the space needed to store the edges as adjacency lists (assuming an integer node id is 32 bits, and the graph is undirected, i.e. each edge is represented as two directed edges). In other words, to store the n possible worlds we only need about three times more space than what is needed to store $\mathcal{G}$.

VI. Experiments

We present our extensive experimental results to test the efficiency, effectiveness, and accuracy of our proposed algorithms. Our implementations are in Java and the experiments are conducted on a commodity machine with Intel i7, 2.2Ghz CPU, and 12Gb RAM, running Ubuntu 18.04.

Datasets and Experimental Framework. Statistics for our datasets are in Table I. We order the datasets based on the number of triangles they contain. Datasets with real probabilities are *flickr*, *dblp*, and *biomine* from [1], [45] and *krogan* from [46].

We also consider datasets *ljournal-2008* and *pokec*. *ljournal-2008* is obtained from Laboratory of Web Algorithmics (http://law.di.unimi.it/datasets.php) and *pokec* is from the Stanford Network Analysis Project (http://snap.stanford.edu). For these networks, we generated edge probabilities uniformly distributed in $(0, 1]$.

A. Efficiency Evaluation

In this section, we report the running times of our proposed algorithms for local nucleus decomposition: one that uses dynamic programming and the other that uses statistical approximations for computing and updating the support of triangles. We denote them by DP and AP, respectively. Next, we report the running times of our (fully) global and weakly-global nucleus decomposition algorithms, which we denote by FG and WG. We set error-bound $\epsilon = 0.1$ and confidence level $\delta = 0.1$. Based on these values and Lemma 1, we set the number of samples to $n = 200 > \lceil \frac{1}{2\epsilon^2} \ln\left(\frac{2}{\delta}\right) \rceil$ (i.e. greater than what is required by Hoeffding's inequality). As such, our results for global and weakly-global notions of nuclueus are approximate but come with strong quality guarantees. Running time results for DP and AP are shown in Figure 1 for different values of θ. Y-axis for the last 3 plots is in log-scale.

Both algorithms perform well on medium-size datasets, *dblp* and *flickr*; computing the nucleus decomposition of these two graphs takes less than 1 sec. For a larger-size dataset, *pokec*, both algorithms complete in less than 10 min. Note that AP clearly outperforms DP on large-size datasets such as *biomine* and *ljournal-2008* for small values of θ. For instance, for *ljournal-2008* with $\theta = 0.1$, it is only AP that can run to completion, whereas DP could not complete after one day. Nevertheless, both DP and AP are able to run in reasonable time for all the other cases, which is good considering that nucleus decomposition is a harder problem than core and truss decomposition.

In general, the running times of both DP and AP decrease significantly as θ increases. This is because the number of triangles which, (a) exist with probability greater than θ and (b) have a support at least k again with probability greater than θ, decreases. As can be seen, AP is faster than DP on all datasets for different values of θ. In addition to the *ljournal-2008* case for which only AP could complete, when $\theta = 0.1$, the gain of AP over DP is about 24% and 30% for *biomine* and *pokec*, respectively.

For speed-up evaluation of AP vs. DP we added two more datasets. The statistics of these datasets are given in Table II. The first dataset is *enwiki-2013*. What is special about this dataset is that its maximum initial nucleus score is $2,813$, which is much larger than in other graphs we consider. We set $\theta = 0.1$; when θ is small, more triangles can have enough probability to be part of a much larger number of 4-cliques. This can cause too much work for DP to compute nucleus scores and update these values when the triangles are being processed in the peeling step. For this dataset, DP was not able to complete the computation within one week. In contrast, AP completed in about 80K sec (less than a day).

The other additional dataset we considered is *itwiki-2013*.

Graph	$\|V\|$	$\|E\|$	$d_{\max}$	p_{avg}	$\|\triangle\|$
enwiki-2013	4,206,785	91,939,728	432,260	0.5	304,083,160
itwiki-2013	1,016,867	23,429,644	91,517	0.5	89,901,299

TABLE II: Additional datasets. $|V|$, $|E|$, $d_{\max}$, p_{avg}, $\triangle$, are number of vertices, edges, max degree, avg edge probability, and number of triangles in the graph, respectively.

The maximum initial nucleus score in this dataset is $1,866$. In this graph, using the same $\theta = 0.1$, DP needs about 40h, whereas AP $16h$, i.e. AP is 2.5 times faster than DP. Moreover, we ran DP and AP on *biomine* with $\theta = 0.01$. DP took about 37.5h, whereas AP 2.5h, thus being 15 times faster than DP.

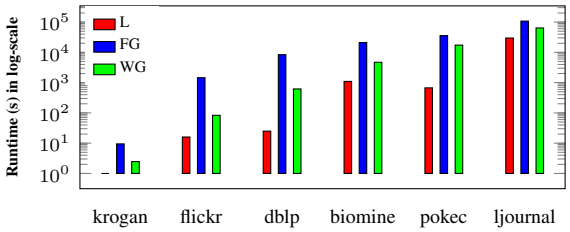


Fig. 2: Run time of L, FG, and WG. FG and WG include the time for L. WG is faster because it performs deterministic decomposition only on a fixed number of sample graphs while FG does so each time a candidate graph is discovered.

Model	$\theta = 0.1$			$\theta = 0.01$		
	Vertices	edges	Time	Vertices	Edges	Time
Local	75	3455	10^3	93	3785	10^5
Weakly-Global	15	157	$4 \cdot 10^3$	55	2332	$2 \cdot 10^5$
Global	4	6	$21 \cdot 10^3$	5	10	$6 \cdot 10^5$

TABLE III: Average number of vertices and edges, and running time for local, weakly-global, and global nucleus subgraphs with θ equal to 0.1 and 0.01.

We report the running time of FG and WG in Figure 2 along with the running time of local (denoted by L in the figure) nucleus decomposition for $\theta = 0.1$ (which as explained above is more difficult than $\theta = 0.2, \ldots, 0.5$). Note that the global and weakly-global nuclei are obtained from the local ones using Algorithms 2 and 3. Therefore, their running time includes the time required for obtaining local nuclei. For local decomposition, we use DP to obtain the probabilistic support of the triangles, except for *ljournal-2008* for which we use AP since DP does not scale for this threshold. Also, we report running times averaged across 5 runs, since the solutions of FG and WG depend on the random sampling steps.

In general WG is faster than FG. This is because WG performs deterministic nucleus decomposition only on a fixed number of sample graphs while FG does the decomposition every time that a candidate graph is detected. We also note that as the graph becomes larger, WG will have to perform nucleus decomposition on larger sample graphs leading to increased running time. For FG, usually candidate graphs are small even for large graphs. So, when the graph becomes lager, the runtime of WG increases more compared to FG.

Dataset	Avg Error	% of $\triangle$ with Error
	$\theta = 0.2/\theta = 0.4$	
krogan	0.0524/0.0209	5.24/2.08
dblp	0.0069/0.0041	0.69/ 0.41
flickr	0.0031/0.0	0.31/0.0
pokec	0.0014/4.15e-5	0.14/0.004
biomine	0.0/0.0	0.0/0.0
ljournal-2008	0.0179/0.0070	1.79/0.69

TABLE IV: Avg difference (error) of AP scores from true DP scores and pct's of triangles with error. Errors are very small.

Moreover, we compare the running time of nucleus decomposition algorithms, local, weakly-global, and global, on *biomine* with $\theta = 0.1$ and $\theta = 0.01$ in Table III (columns 4,7). For the local decomposition (L) we used DP because we are interested in the relative difference in running time for the different nucleus notions and L is the initial step for computing WG and FG.

When θ decreases, running times increase since more triangles can have enough probability to be contained in a local nucleus subgraph. In terms of the size of the results, Table III shows the average number of vertices and edges for the L, WG, and FG subgraphs aggregated over all $k \in [1, k_{\max}]$. In general, the average values increase as we decrease the threshold. This is due to the fact that by decreasing θ more triangles can have enough probability to be contained in 4-cliques.

B. Accuracy Evaluation

To evaluate the accuracy of the AP algorithm, we compare the final nucleus scores obtained by DP and AP algorithms. We report the results in Table IV. We show the results for θ equal to 0.2 and 0.4, since for the remaining values the error results do not differ significantly. The second column shows the average difference (error) from true value over the total number of triangles. The last column shows the percentage of triangles whose value is different from their exact value.

As can be seen, the average error is quite small for all the datasets we consider. Particularly, for *flickr* with $\theta = 0.4$ and *biomine* with $\theta = 0.2$ and $\theta = 0.4$ we have that AP computes nucleus decomposition with *zero error*. Also, the percentage of triangles with an error score is very small, namely less than 1% for all the datasets, except *krogan* and *ljournal-2008*. For these two, the percentages are still small, 5.24% and 1.79%, respectively. These results show that the output of AP is very close to that of exact computation by DP, and thus, AP is a reliable approximation methodology.

C. Quality Evaluation of Nucleus Subgraphs

Here we compare the cohesiveness of ℓ-(k, θ)-nucleus with the cohesiveness of local (k, γ)-truss [24] and (k, η)-core [1]. We use two metrics. The first metric is the probabilistic density (PD) of a graph $\mathcal{G}$, which we denote by $\text{PD}(\mathcal{G})$ and is defined as follows [24]: $\text{PD}(\mathcal{G}) = \frac{\sum_{e \in E} p(e)}{\frac{1}{2}|V| \cdot (|V|-1)}$. In words,

Graph	θ	$\lvert V_N\rvert/\lvert V_T\rvert/\lvert V_C\rvert$	$\lvert E_N\rvert/\lvert E_T\rvert/\lvert E_C\rvert$	$k_{Nmax}/k_{Tmax}/k_{Cmax}$	$\text{PD}_N/\text{PD}_T/\text{PD}_C$	$\text{PCC}_N/\text{PCC}_T/\text{PCC}_C$	$\text{Time}_N/\text{Time}_T/\text{Time}_C$
dblp	0.1	19/34/115	171/561/6555	9/14/26	0.800/0.611/0.264	0.790/0.620/0.317	25/100/15.86
dblp	0.3	14/26/138	108/366/6693	7/11/23	0.9917/0.785/0.277	0.9918/0.789/0.384	11/30/16.99
pokec	0.1	13/72/288	121/1335/10592	3/8/27	0.678/0.341/0.129	0.636/0.393/0.170	672/1162/4401
pokec	0.3	6/71/278	21/1031/10142	2/6/25	0.815/0.321/0.132	0.793/0.406/0.172	298/980/4349
biomine	0.1	103/102/430	5231/5127/92200	18/33/79	0.540/0.538/0.211	0.540/0.538/0.217	1098/7642/5792
biomine	0.3	7/102/431	23/5125/92625	2/28/73	0.714/0.538/0.212	0.701/0.539/0.218	939/1563/5685

TABLE V: Cohesiveness statistics of l-(k,θ)-nucleus N, (k,θ)-truss, T, and (k,θ)-core, C on *dblp*, *pokec*, and *biomine*. The number of vertices ($\lvert V_N\rvert/\lvert V_T\rvert/\lvert V_C\rvert$), the number of edges ($\lvert E_N\rvert/\lvert E_T\rvert/\lvert E_C\rvert$), maximum nucleus/truss/core score ($k_{Nmax}/k_{Tmax}/k_{Cmax}$), the probabilistic density ($\text{PD}_N/\text{PD}_T/\text{PD}_C$), and the probabilistic clustering coefficient ($\text{PCC}_N/\text{PCC}_T/\text{PCC}_C$), respectively.

n	AV(PD)		AV(PCC)		AV(Edge)		AV(Vertex)		ϵ	δ
150	.905	.726	.903	.770	12.744	55.631	5.336	11.971	0.1	0.1
300	.906	.733	.903	.773	12.725	52.960	5.334	11.543	0.07	0.05
500	.906	.729	.903	.767	13.005	53.883	5.383	11.703	0.05	0.06
1000	.905	.725	.902	.766	12.823	53.772	5.356	11.745	0.05	0.01
2000	.906	.727	.903	.768	12.782	54.264	5.350	11.792	0.03	0.05
AV	.906	.728	.903	.769	12.816	54.102	5.352	11.751		
SD	.0004	.003	.0003	.002	.112	.978	.020	.155		

TABLE VI: Effect of sample size (n), ϵ, and δ on different average metrics, average PD, average PCC, average number of edges, and average number of vertices for global and weakly-global nuclei. The first and second columns for each metric are for global and weakly-global nuclei, respectively. The results shown here are on krogan with $\theta = 0.1$. Observe that standard deviation (SD) is not more than 1.8% of the average for all columns. For some of the columns SD is much smaller, e.g. for average PD (first column) it is only 0.05%.

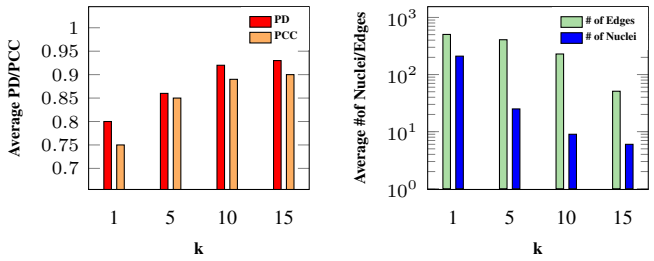


Fig. 3: Average PD and PPC, average number of edges, average number of ℓ-(k,θ)-nuclei for *flickr* with $\theta = 0.3$.

PD of a probabilistic graph is the ratio of the sum of edge probabilities to the possible number of edges in a graph.

The second metric is probabilistic clustering coefficient (PCC). It measures the level of tendency of the nodes to cluster together. Given a probabilistic graph $\mathcal{G}$, its PCC is defined as follows [24], [47]: $\text{PCC}(\mathcal{G}) = \frac{3\sum_{\triangle_{uvw}\in\mathcal{G}} p(u,v)\cdot p(v,w)\cdot p(u,w)}{\sum_{(u,v),(u,w),v\neq w} p(u,v)\cdot p(u,w)}$.

For probabilistic nucleus, probabilistic truss and probabilistic core subgraphs, we use the same threshold $\theta = \gamma = \eta$, set to 0.1 and 0.3. (γ is used as threshold in the truss case [24], and η is used as threshold in the core case [1]). Table V reports results on *dblp*, *pokec*, and *biomine*. Results for the other datasets are similar. For a given threshold, we report the statistics of local (k_{Nmax},θ)-nucleus, (k_{Tmax},γ)-truss, and (k_{Cmax},η)-core, where k_{Nmax}, k_{Tmax}, and k_{Cmax} are maximum nucleus, truss and core scores, respectively. Also, for k_{Nmax}, k_{Tmax}, and k_{Cmax}, we might obtain more than one connected component; we report the average statistics over such components. We denote by V_N, V_T, V_C, the sets of nodes, by E_N, E_T, E_C, the sets of edges, by PD_N, PD_T, PD_C, the PD's and by PCC_N, PCC_T, PCC_C, the PCC's of nucleus, truss, and core components, respectively. The last column shows the running time for computing each decomposition. We observe that sometimes nucleus decomposition is faster than truss decomposition. This is because in nucleus decomposition there could be fewer triangles that survive the specified threshold in terms of support than edges in truss decomposition.

As can be seen in the table, (k_{Nmax},θ)-nucleus produces high quality results in terms of PD and PCC. We achieve a significantly higher PD and PCC for nucleus compared to truss and core. For instance, for *dblp* the PD for nucleus is 0.8 versus 0.611 and 0.264 for truss and core, which translates for nucleus being about 30% and 200% more dense than truss and core. Similar conclusions can be drawn for PCC as well.

Moreover, Figure 3 reports the average PD, average PCC, average edges in each ℓ-(k,θ)-nucleus, and number of connected components (ℓ-(k,θ)-nuclei) for an example dataset *flickr* with fixed $\theta = 0.3$ and varying k. We see that even for small values of k, PD and PCC are considerably high (above 70-80%). In general, PD and PCC become larger as k increases, since denser nuclei will be detected by removing triangles having low support probability to be part of a 4-clique. This causes the final subgraphs to have edges with high probability only. Furthermore, since ℓ-(k,θ)-nucleus implies connectivity, the number of connected components increases as k decreases. It results in an increase in the average number of edges in each ℓ-(k,θ)-nucleus.

Finally, we compare the PD and PCC values of **g**-(k,θ)-nucleus, **w**-(k,θ)-nucleus over 5 runs of these algorithms, and ℓ-(k,θ)-nucleus, for *krogan*, *flickr*, and *dblp* datasets using $\theta = 0.001$, and averaging over all the possible values of k. The results are shown in Figure 4. We see that **g**-(k,θ)-nucleus achieves higher cohesiveness as expected. In addition, **w**-(k,θ)-nucleus exhibits quite good PD and PCC values higher than those for ℓ-(k,θ)-nucleus.

Effect of ϵ and δ. We consider *krogan* dataset with $\theta = 0.1$. The choice of ϵ and δ influence the number n of possible worlds we sample. For $\epsilon = 0.1$ and $\delta = 0.1$ we obtain $n = 150$. In order to see the fidelity of our results, we experiment by increasing n to higher values, namely $300, 500, 1000, 2000$.

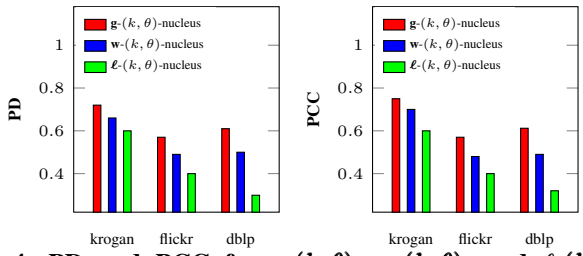


Fig. 4: PD and PCC for g-(k,θ), w-(k,θ), and ℓ-(k,θ) nuclei on *krogan*, *flickr*, and *dblp*.

As the results in Table VI show, the following metrics about global and weakly-global nuclei: average PD, average PCC, average number of vertices, and average number of edges change very little. Specifically, the first two metrics are dispersed by not more than 0.4% around their mean over the different values of n, and the last two metrics are dispersed by not more than 1.8%. There can be many ϵ and δ values corresponding to a given sample size; for illustration, for $n = 150$, we can have $\epsilon = 0.1$, $\delta = 0.1$, whereas for $n = 2000$, we can have $\epsilon = 0.03$, $\delta = 0.05$, i.e. we see that even though in the latter case the ϵ and δ decrease by a factor of 3 and 2, respectively, still the nuclei results in terms of the aforementioned metrics are almost the same. This validates the choice of ϵ and δ to 0.1 since lower values do not offer significant improvement in the quality of results.

D. Case Studies

DBLP. To show the usefulness of nucleus decomposition in probabilistic graphs, we apply our decomposition algorithms to solve the *task-driven team formation* problem for a DBLP network. In task-driven team formation [1], we are given a probabilistic graph $\mathcal{G}^T = (V, E, p^T)$, which is particularly obtained for task T. Vertices in $\mathcal{G}^T$ are individuals and edge probabilities are obtained with respect to task T as described in [1]. Given a query $\langle Q, T\rangle$, where $Q \subset V$, and T is a set of keywords describing a task, the goal is to find a set of vertices that contain Q and make a good team to perform the task described by the keywords in T. By a good team we mean a good affinity among the team members in terms of collaboration for the given task. To solve task-driven team formation using nucleus decomposition, we extend the definition of [1] to employ probabilistic nucleus: Given a probabilistic graph $\mathcal{G}^T = (V, E, p^T)$ with respect to a task T, a query set Q of vertices, and a threshold θ, apply nucleus decomposition on $\mathcal{G}^T$ and find a (k,θ)-nucleus (local/weakly-global/global) which **(1)** contains the vertices in Q, and **(2)** has the highest value of k for the given θ, and return the obtained subgraph as a solution.

In our experiment, we use a DBLP collaboration network from [1], where vertices are authors, and edges represent collaboration on at least one paper. The dataset has $1,089,442$ vertices and $4,144,697$ edges. For each edge, we take the bag of words of the title of all papers coauthored by the two authors connected by the edge and apply Latent Dirichlet Allocation (LDA) [1], [48] to infer its topics and calculate the edge probability. Given a task T with keywords, and the input collaboration network, we obtain a probabilistic graph $\mathcal{G}^T$, in which $p(u, v)$ represents the collaboration level in the papers co-authored by u and v related to task T ([1], [24]).

The first sample query we consider is $\langle$\{"algorithm"\}, \{"Erik_D._Demaine", "J._Ian_Munro", "John_Iacono"\}$\rangle$. Figure 5a shows the subgraph obtained by ℓ-(k,θ)-nucleus and **w**-(k,θ)-nucleus decompositions, where $k = 2$ and $\theta = 10^{-11}$. The threshold is the same as in the case studies of previous works (on truss and core). As discussed in [1], the edge probabilities in the dataset are very small, which requires setting threshold θ to a small value. More systematically, picking an appropriate value for the threshold can be done using binary search over $(0, b]$, where $b \leq 1$. The subgraph contains all the three authors in the query. It has 10 vertices and 33 edges. As can be seen, the obtained subgraph is quite good for task-driven team formation. All the authors in the subgraph are well-known and have strong collaboration affinity to work on a research paper related to algorithms. A **g**-(k,θ)-nucleus (same k and θ) that contains the query vertices is shown with thick blue edges in the same figure. As expected, this subgraph is more cohesive and it happens to be a clique of size 6. Its density and clustering coefficient (PCC) is 0.138 and 0.140 as opposed to 0.099 and 0.110 for the local and weakly-global subgraphs. From a research perspective the collaborations of the academicians in the blue subgraph are more focused on designing efficient data-structures.

We run the global truss algorithm on the dataset. As expected the global truss subgraph which contains the query authors is bigger than global nucelus (9 vertices and 18 edges) and its PD and PCC are lower (0.067 and 0.086).

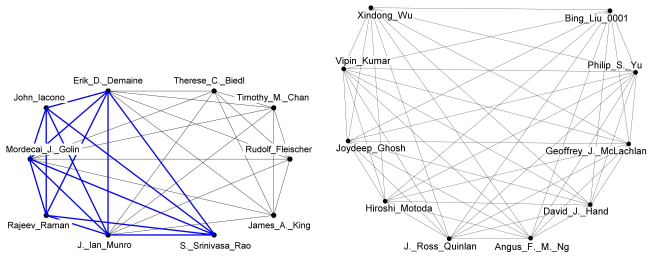
We also run global core decomposition as in [26] for the same value k and θ. It should be noted that the global core definition is different from global truss and global nucleus. Also, it does not assume connectivity between nodes. However, for fairness of comparison, we considered a connected component of this subgraph which contains the query authors. The obtained subgraph contains 569 vertices and 5294 edges, with PD 0.003 and PCC 0.061.

Regarding local truss, we obtained a subgraph of 170 vertices and 1033 edges with PD 0.008 and PCC 0.0872. On the other-hand, local core results in PD and PCC being equal to 0.0084 and 0.0659 with 226 vertices and 2631 edges. As can be seen, our nucleus decomposition results in much better subgraphs in terms of subgraph size and cohesiveness.

The second query we use shows the usefulness of the weakly-global notion. It has keyword \{"algorithm"\} and vertices \{"Xindong_Wu", "Bing_Liu_0001", "Vipin_Kumar"\}. Figure 5b shows the **w**-(k,θ) nucleus for this query, where the threshold is the same as before, and $k = 1$. The local nucleus containing the query authors had more than 100 nodes while the global nucleus containing these three query authors was empty. This example shows that the weakly global notion can discover interesting teams when the other two notions produce teams that are too big or too small (or empty). In particular, all the authors in the resulting subgraph are very

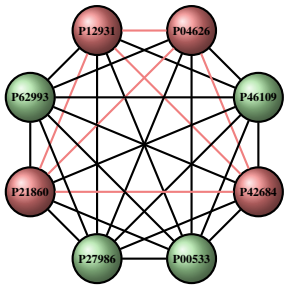
Notion	Max k	Nodes	Density
l-core	88	2408	0.04
g-core	31	10026	0.01
l-truss	4	5787	0.01
l-nucleus	1	95	0.06
g-truss	2	10	**0.44**
w-nucleus	1	8	**0.51**
g-nucleus	1	4	**0.56**

TABLE VII: Comparison of different dense subgraph notions with respect to (1) max k for which the subgraph contains the proteins of interest, (2) number of nodes in the subgraph, and (3) density of the subgraph. $\theta = 0.001$.



(a) **(b)**

Fig. 5: a) A case study of task-driven team formation with keyword {"algorithm"}, and query vertices {"Erik_D._Demaine", "J._Ian_Munro", "John_Iacono"}, $k = 2$, and $\theta = 10^{-11}$. The depicted graph with thick blue edges corresponds to a g-(k, θ) nucleus. The whole graph (of 10 vertices) is a ℓ-(k, θ) nucleus which coincides with a w-(k, θ) nucleus in this example. b) A weakly-global w-(k, θ) nucleus for task-driven team formation with query nodes {"Xindong_Wu", "Bing_Liu_0001", "Vipin_Kumar"}, and keyword { "algorithm"}. $k = 1$, and $\theta = 10^{-11}$.



(a)

Fig. 6: w-(k, θ) nucleus (green and pink nodes) and a g-(k, θ)-nucleus (pink nodes) that contain the proteins of interest P04626, P12931, P42684. $k = 1$ and $\theta = 0.001$.

well-know and have similar research area which can form a good team related to keyword algorithm (query keyword). On the other-hand, global truss gave empty result. In addition, both local truss and core decomposition, did not lead to a desired team as the number of vertices in such graphs was very large, $16,663$ and $31,300$, respectively. In fact, it is not realistic for this amount of authors to collaborate on a paper or project related to algorithms. The PD and PCC for weakly-global subgraph is 0.036 and 0.0388, as opposed to density 0.00005 and PCC 0.0280 in local truss and PD 0.000001 and PCC 0.0236 in local core. A similar argument holds for global core with $2,997$ vertices, $35,354$ edges, PD 0.0004, and PCC 0.0294. For local nucleus decomposition, cohesiveness results show PD 0.03 and PCC 0.0331 with a number of vertices of 100 which is much smaller than local core and local truss.

Human Biomine. We use the human biomine dataset [49], which has 861,812 nodes and 8,666,287 edges. This dataset is different from the biomine dataset we used for our efficiency evaluation. We consider how our notions perform in detecting proteins/genes that interact with the **SARS-CoV-2** coronavirus. Bouhaddou et al. [50] found that during the **SARS-CoV-2** virus infection, changes in activities can happen for human kinases. We select three proteins, P04626, P12931 and P42684; they are tyrosine kinase-related proteins and come from UniProt, which is a freely accessible database of protein sequences and functional information. The gene names associated with these proteins are SRC, ERBB2, and ABL2. These proteins have received literature support for interaction with **SARS-CoV-2** coronavirus [50]–[56]. We refer to them as proteins of interest. We find the subgraphs obtained by local, weakly-global, and global nucleus decomposition which contain these three nodes. Moreover, at the same time we compare these graphs with their counterparts, truss and core in terms of density and size of the subgraph. For all the notions we set threshold $\theta = 0.001$.

Table VII shows the comparison of different dense subgraph notions with respect to (1) largest k for which the subgraph contains the proteins of interest, (2) number of nodes in the subgraph, and (3) density of the subgraph. We see that l-nucleus is denser than l-truss and both l-core and g-core. Also, w-nucleus and g-nucleus are denser than g-truss. In terms of nodes, l-nucleus gives a subgraph which is much smaller than the subgraphs of l-core, g-core, and l-truss. More precisely, with respect to l-nucleus, the three proteins of interest appear in a nucleus of 95 vertices and 509 edges. To see which kind of biology function/process our detected community represent, we use *Metascape* (https://metascape.org/gp/index.html#/main/step1). Metascape [57] is a web-based portal that provides comprehensive gene list annotation and analysis resources. Using Metascape, we find that the proteins in the local nucleus are associated with several diseases, most of them being forms of cancer (16 out 20). The p-values of the association are less than 10^{-18}, which is statistically very significant. Please see our full version [35] for a figure with more details.

Figure 6 shows the weakly global and global nuclei which contain the proteins of interest. All the nodes (green and pink) comprise the weakly global subgraph. The pink nodes comprise the global nucleus subgraph. Using Metascape, we find that the proteins in our weakly-global and global subgraphs are associated with some more specific forms of cancer such as *Uterine Carcinosarcoma* and *Hormone Refractory Prostate Cancer*, respectively, with p-values less than 10^{-6}, which are statistically quite significant, especially given the fact that these subgraphs are much smaller than the local nucleus (in general, the more observations we have, the smaller the p-values become). These findings are useful to biologists in order to perform targeted tests for checking whether drugs for the

treatment of these diseases can also be repurposed for treating COVID-19 [54]. There are over 250 anticancer drugs approved by the FDA, but far fewer for specific kinds of cancer. Thus, showing connections to specific forms helps narrow the choice of drugs to repurpose.

In summary, it is running all the three versions of nucleus decomposition on the Biomine dataset that gives surprising subgraphs pointing to potentially useful further investigation by biologists. Running only local nucleus decomposition will miss such interesting groups, no matter how we set k and θ.

BrightKite. *BrightKite* was a location-based social networking service provider where users shared their locations by checking-in. The friendship network was collected using their public API, and consists of 58,228 nodes and 214,078 edges, and 4,491,143 checkins between April 2008 and October 2010. We generated probabilities for each edge based on the Jaccard similarity between the neighborhoods of two endpoints. Running weakly-global and global nucleus decompositions on this dataset with $\theta = 0.1$, we retrieve 300 and 20 **g**-(k, θ) and **w**-(k, θ) nuclei, respectively. For weakly-global subgraphs, k ranges in $[1, 5]$, and for global subgraphs, k can take on values of 1 and 2.

As expected, global nuclei obtain better cohesiveness in terms of density and clustering coefficient. In particular, the average density and clustering coefficient in global nuclei over all values of k, is 0.6951 and 0.6947 as opposed to 0.4844 and 0.5052 in weakly-global nuclei. We also report another interesting observation on this dataset. We obtain the average number of checkins by users in the detected subgraphs. The average number of user checkins in global nuclei is about 6% more than those in weakly-global nuclei. Moreover, there exist periods, for instance, the period between August 2008 and April 2009, in which the average number of checkins of the users in the global nuclei is 57% more than the average number of checkins in the weakly-global subgraphs. These results show that global nuclei can capture better user engagement (as measured by the checkins) than weakly-global nuclei.

Remark. We explain that all our three models are useful and they should be used in tandem. Local nucleus helps to identify dense subgraphs of interest. We can adjust k and θ to obtain smaller and denser subgraphs. However, global and weakly global nuclei can identify pockets that are impossible to obtain with local nucleus no matter how we adjust k and θ.

In our DBLP use case, the local and weakly-global notions helped us identify a dense subgraph of researchers working on Algorithms, however, the global nucleus gave a particular pocket of researchers, who, after close examination, turned out to be especially focused on designing efficient data-structures. Then in the same case study, we were able to identify a useful weakly-global nucleus of researchers, who are well known to work on algorithms for data mining. The local nucleus was too big, whereas the global nucleus was empty. In the Biomine dataset, we observe that the group of proteins in a local nucleus containing three proteins of interest were related to many forms of cancer even though the proteins of interest have received literature support related to COVID-19. Regarding weakly-global and global notions, they were able to find subgraphs of the local nucleus that were comprised of proteins related to more specific cancer diseases. All these examples show that an analyst should run all the three versions of nucleus decomposition in tandem on a dataset and then closely examine the results. More than density, what is important is the detection of small pockets of nodes with nice properties that escape getting identified by other notions.

VII. Related Work

In deterministic graphs, core and truss decompositions have been studied extensviely [58]–[69]. Core decomposition in probabilistic graphs has been studied in [1], [23], [26], [70]. Bonchi et al. [1] were the first to introduce core decomposition for such graphs. They focus on finding a subgraph in which each vertex is connected to k neighbors within that subgraph with high probability. In [23] more efficient algorithms were proposed which can also handle graphs that do not fit in main memory. In [26], the authors focus on finding a subgraph which contains nodes with high probability to be k-core member in the probabilistic graph. In [70], an index-based structure is defined for processing core decomposition in probabilistic graphs.

In the probabilistic context, the notion of local (k, η)-truss is introduced by Huang, Lu, and Lakshmanan in [24]. Their proposed algorithm for computing local (k, η)-truss is based on iterative peeling of edges with support less than k and updating the support of affected edges. Also, [24] proposed the notion of global (k, η)-truss based on the probability of each edge belonging to a k-truss in a possible world. In [71] an approximate algorithm for the local truss decomposition is proposed to efficiently compute the tail probability of edge supports in the peeling process of [24]. In [72] truss decomposition is computed using an index-based approach.

Building on the well-studied notions of core and truss decomposition, Sarıyüce et al. [28] introduce nucleus decomposition in *deterministic* graphs. They propose an algorithm for computing $(3, 4)$-nuclei. In a more recent work, Sarıyüce et al. [22] propose efficient distributed algorithms for nucleus decomposition. Our work is the first to study nucleus decomposition in probabilistic graphs.

VIII. Conclusions

In this work, we made several key contributions. We introduced the notion of local, weakly-global and global nuclei for probabilistic graphs. We showed that computing weakly-global and global nuclei is intractable. We complemented these hardness results with effective algorithms to approximate them using techniques from Monte-Carlo sampling.

We designed a polynomial time, peeling based algorithm for computing local nuclei based on dynamic programming and showed that its efficiency can be much improved using novel approximations based on Poisson, Binomial and Normal distributions. Finally, using an in-depth experimental study, we demonstrated the efficiency, scalability and accuracy of our algorithms for nucleus decomposition on real world datasets.

REFERENCES

[1] F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich, "Core decomposition of uncertain graphs," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2014, pp. 1316–1325.

[2] A. P. Mukherjee, P. Xu, and S. Tirthapura, "Mining maximal cliques from an uncertain graph," in *2015 IEEE 31st Int. Conf. on Data Engineering*. IEEE, 2015, pp. 243–254.

[3] R. Jin, L. Liu, and C. Aggarwal, "Discovering highly reliable subgraphs in uncertain graphs," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2011, pp. 992–1000.

[4] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 137–146.

[5] C. Budak, D. Agrawal, and A. El Abbadi, "Limiting the spread of misinformation in social networks," in *Proceedings of the 20th International Conference on World Wide Web*, 2011, pp. 665–674.

[6] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: Near-optimal time complexity meets practical efficiency," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 2014, pp. 75–86.

[7] R. Jin, L. Liu, B. Ding, and H. Wang, "Distance-constraint reachability computation in uncertain graphs," *Proceedings of the VLDB Endowment*, vol. 4, no. 9, pp. 551–562, 2011.

[8] Z. Zou, H. Gao, and J. Li, "Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 633–642.

[9] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Learning influence probabilities in social networks," in *Proceedings of the third ACM International Conference on Web search and Data Mining*, 2010, pp. 241–250.

[10] N. Korovaiko and A. Thomo, "Trust prediction from user-item ratings," *Social Network Analysis and Mining*, vol. 3, no. 3, pp. 749–759, 2013.

[11] U. Kuter and J. Golbeck, "Using probabilistic confidence models for trust inference in web-based social networks," *ACM Transactions on Internet Technology (TOIT)*, vol. 10, no. 2, p. 8, 2010.

[12] G. Cavallaro, "Genome-wide analysis of eukaryotic twin cx 9 c proteins," *Molecular BioSystems*, vol. 6, no. 12, pp. 2459–2470, 2010.

[13] M. T. Dittrich, G. W. Klau, A. Rosenwald, T. Dandekar, and T. Müller, "Identifying functional modules in protein–protein interaction networks: an integrated exact approach," *Bioinformatics*, vol. 24, no. 13, pp. i223–i231, 2008.

[14] J. Dong and S. Horvath, "Understanding network concepts in modules," *BMC systems biology*, vol. 1, no. 1, p. 24, 2007.

[15] R. Sharan, I. Ulitsky, and R. Shamir, "Network-based prediction of protein function," *Molecular systems biology*, vol. 3, no. 1, p. 88, 2007.

[16] F. Zhao and A. Tung, "Large scale cohesive subgraphs discovery for social network visual analysis," *Proceedings of the VLDB Endowment*, vol. 6, pp. 85–96, 2012.

[17] B. Zhang and S. Horvath, "A general framework for weighted gene co-expression network analysis," *Statistical applications in genetics and molecular biology*, vol. 4, no. 1, 2005.

[18] E. Fratkin, B. T. Naughton, D. L. Brutlag, and S. Batzoglou, "Motifcut: regulatory motifs finding with maximum density subgraphs," *Bioinformatics*, vol. 22, no. 14, pp. e150–e157, 2006.

[19] Y. Fang, X. Huang, L. Qin, Y. Zhang, W. Zhang, R. Cheng, and X. Lin, "A survey of community search over big graphs," *The VLDB Journal*, vol. 29, no. 1, pp. 353–392, 2020.

[20] R. Li, L. Qin, F. Ye, G. Wang, J. X. Yu, X. Xiao, N. Xiao, and Z. Zheng, "Finding skyline communities in multi-valued networks," *The VLDB Journal*, pp. 1–26, 2020.

[21] L. Antiqueira, O. N. Oliveira Jr, L. da Fontoura Costa, and M. d. G. V. Nunes, "A complex network approach to text summarization," *Information Sciences*, vol. 179, no. 5, pp. 584–599, 2009.

[22] A. E. Sariyüce, C. Seshadhri, and A. Pinar, "Local algorithms for hierarchical dense subgraph discovery," *Proc. of the VLDB Endowment*, vol. 12, no. 1, pp. 43–56, 2018.

[23] F. Esfahani, V. Srinivasan, A. Thomo, and K. Wu, "Efficient computation of probabilistic core decomposition at web-scale." in *Proceedings of the 22nd International Conference on Extending Database Technology (EDBT)*, 2019, pp. 325–336.

[24] X. Huang, W. Lu, and L. V. Lakshmanan, "Truss decomposition of probabilistic graphs: Semantics and algorithms," in *Proceedings of the 2016 International Conference on Management of Data*. ACM, 2016, pp. 77–90.

[25] W. Khaouid, M. Barsky, V. Srinivasan, and A. Thomo, "K-core decomposition of large networks on a single pc," *Proceedings of the VLDB Endowment*, vol. 9, no. 1, pp. 13–23, 2015.

[26] Y. Peng, Y. Zhang, W. Zhang, X. Lin, and L. Qin, "Efficient probabilistic k-core computation on uncertain graphs," in *Proceedings of the IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1192–1203.

[27] J. Wang and J. Cheng, "Truss decomposition in massive networks," *Proceedings of the VLDB Endowment*, vol. 5, no. 9, 2012.

[28] A. E. Sariyüce, C. Seshadhri, A. Pinar, and U. V. Catalyurek, "Finding the hierarchy of dense subgraphs using nucleus decompositions," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 927–937.

[29] A. E. Sariyüce, C. Seshadhri, A. Pinar, and Ü. V. Çatalyürek, "Nucleus decompositions for identifying hierarchy of dense subgraphs," *ACM Transactions on the Web (TWEB)*, vol. 11, no. 3, pp. 1–27, 2017.

[30] R. Saxena, S. Kaur, and V. Bhatnagar, "Social centrality using network hierarchy and community structure," *Data Mining and Knowledge Discovery*, vol. 32, no. 5, pp. 1421–1443, 2018.

[31] Y. Zhao, X. Dong, and Y. Yin, "Effective and efficient dense subgraph query in large-scale social internet of things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2726–2736, 2019.

[32] S. Zhang, D. Zhou, M. Y. Yildirim, S. Alcorn, J. He, H. Davulcu, and H. Tong, "Hidden: hierarchical dense subgraph detection with application to financial fraud detection," in *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 2017, pp. 570–578.

[33] Q. Wu, X. Huang, A. Culbreth, J. Waltz, L. E. Hong, and S. Chen, "Extracting brain disease-related connectome subgraphs by adaptive dense subgraph discovery," *bioRxiv*, 2020.

[34] X. Ma, G. Zhou, J. Shang, J. Wang, J. Peng, and J. Han, "Detection of complexes in biological networks through diversified dense subgraph mining," *Journal of Computational Biology*, vol. 24, no. 9, pp. 923–941, 2017.

[35] F. Esfahani, V. Srinivasan, A. Thomo, and K. Wu, "Nucleus decomposition in probabilistic graphs: Hardness and algorithms," *arXiv preprint arXiv:2006.01958*, 2021, also available at: https://tinyurl.com/294fmk7b.

[36] V. Batagelj and M. Zaveršnik, "Fast algorithms for determining (generalized) core groups in social networks," *Advances in Data Analysis and Classification*, vol. 5, no. 2, pp. 129–145, 2011.

[37] L. Le Cam, "An approximation theorem for the poisson binomial distribution." *Pacific Journal of Mathematics*, vol. 10, no. 4, pp. 1181–1197, 1960.

[38] A. Lyapunov, "Nouvelle forme de la théoreme dur la limite de probabilité," *Mémoires de l'Academie Impériale des Sci. de St. Petersbourg*, vol. 12, pp. 1--24, 1901.

[39] F. A. Haight, "Handbook of the poisson distribution," 1967.

[40] A. Röllin, "Translated poisson approximation using exchangeable pair couplings," *The Annals of Applied Probability*, vol. 17, no. 5/6, pp. 1596–1614, 2007.

[41] A. Papoulis and S. U. Pillai, *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education, 2002.

[42] W. Ehm, "Binomial approximation to the poisson binomial distribution," *Statistics & Probability Letters*, vol. 11, no. 1, pp. 7–16, 1991.

[43] N. Mukhopadhyay, *Probability and statistical inference*. CRC Press, 2000.

[44] W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The Collected Works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.

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

[46] N. J. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, N. Datta, A. P. Tikuisis *et al.*, "Global landscape of protein complexes in the yeast saccharomyces cerevisiae," *Nature*, vol. 440, no. 7084, p. 637, 2006.

[47] J. J. Pfeiffer and J. Neville, "Methods to determine node centrality and clustering in graphs with uncertain structure," in *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[48] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[49] V. Podpečan, c. Ramšak, K. Gruden, H. Toivonen, and N. Lavrač, "Interactive exploration of heterogeneous biological networks with biomine explorer," *Bioinformatics*, 06 2019. [Online]. Available: https://doi.org/10.1093/bioinformatics/btz509

[50] M. Bouhaddou, D. Memon, B. Meyer, K. M. White, V. V. Rezelj, M. C. Marrero, B. J. Polacco, J. E. Melnyk, S. Ulferts, R. M. Kaake *et al.*, "The global phosphorylation landscape of sars-cov-2 infection," *Cell*, vol. 182, no. 3, pp. 685–712, 2020.

[51] M. Marchetti, "Covid-19-driven endothelial damage: complement, hif-1, and abl2 are potential pathways of damage and targets for cure," *Annals of hematology*, pp. 1–7, 2020.

[52] W.-j. Zheng, Q. Yan, Y.-s. Ni, S.-f. Zhan, L.-l. Yang, H.-f. Zhuang, X.-h. Liu, and Y. Jiang, "Examining the effector mechanisms of Xuebijing injection on COVID-19 based on network pharmacology," *BioData mining*, vol. 13, p. 17, 2020.

[53] K. Taniguchi-Ponciano, E. Vadillo, H. Mayani, C. R. Gonzalez-Bonilla, J. Torres, A. Majluf, G. Flores-Padilla, N. Wacher-Rodarte, J. C. Galan, E. Ferat-Osorio *et al.*, "Increased expression of hypoxia-induced factor 1α mRNA and its related genes in myeloid blood cells from critically ill COVID-19 patients," *Annals of Medicine*, vol. 53, no. 1, pp. 197–207, 2021.

[54] Y. Guo, F. Esfahani, X. Shao, V. Srinivasan, A. Thomo, L. Xing, and X. Zhang, "Integrative COVID-19 Biological Network Inference with Probabilistic Core Decomposition," *Briefings in Bioinformatics*, 2021 (in press), doi: 10.1093/bib/bbab455, https://www.biorxiv.org/content/10.1101/2021.06.23.449535v1.full.pdf.

[55] H. Zhao, M. Mendenhall, and M. W. Deininger, "Imatinib is not a potent anti-sars-cov-2 drug," *Leukemia*, vol. 34, no. 11, pp. 3085–3087, 2020.

[56] K. H. Ebrahimi, J. Gilbert-Jaramillo, W. S. James, and J. S. McCullagh, "Interferon-stimulated gene products as regulators of central carbon metabolism," *The FEBS journal*, vol. 288, no. 12, p. 3715, 2021.

[57] Y. Zhou, B. Zhou, L. Pache, M. Chang, A. H. Khodabakhshi, O. Tanaseichuk, C. Benner, and S. K. Chanda, "Metascape provides a biologist-oriented resource for the analysis of systems-level datasets," *Nature communications*, vol. 10, no. 1, pp. 1–10, 2019.

[58] S. B. Seidman, "Network structure and minimum degree," *Social networks*, vol. 5, no. 3, pp. 269–287, 1983.

[59] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu, "Querying k-truss community in large and dynamic graphs," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 2014, pp. 1311–1322.

[60] J. Cohen, "Trusses: Cohesive subgraphs for social network analysis," *National security agency technical report*, vol. 16, pp. 3–1, 2008.

[61] Y. Zhang and S. Parthasarathy, "Extracting analyzing and visualizing triangle k-core motifs within networks," in *Proceedings of IEEE 28th International Conference on Data Engineering*. IEEE, 2012, pp. 1049–1060.

[62] P. Chen, C. K. Chou, and M. Chen, "Distributed algorithms for k-truss decomposition," in *Proceedings of 2014 IEEE International Conference on Big Data (Big Data)*. IEEE, 2014, pp. 471–480.

[63] S. Chen, R. Wei, D. Popova, and A. Thomo, "Efficient computation of importance based communities in web-scale networks using a single machine," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016, pp. 1553–1562.

[64] A. Montresor, F. De Pellegrini, and D. Miorandi, "Distributed k-core decomposition," *IEEE Transactions on parallel and distributed systems*, vol. 24, no. 2, pp. 288–300, 2012.

[65] F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, "When engagement meets similarity: Efficient (k,r)-core computation on social networks," *Proc. VLDB Endow.*, vol. 10, no. 10, p. 998–1009, 2017.

[66] J. Cheng, Y. Ke, S. Chu, and M. T. Özsu, "Efficient core decomposition in massive networks," in *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*. IEEE, 2011, pp. 51–62.

[67] A. E. Saríyüce, B. Gedik, G. Jacques-Silva, K. L. Wu, and Ü. V. Çatalyürek, "Streaming algorithms for k-core decomposition," *Proceedings of the VLDB Endowment*, vol. 6, no. 6, pp. 433–444, 2013.

[68] K. Wang, X. Lin, L. Qin, W. Zhang, and Y. Zhang, "Efficient bitruss decomposition for large-scale bipartite graphs," in *Proceedings of 2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 661–672.

[69] G. Preti, G. De Francisci Morales, and F. Bonchi, "Strud: Truss decomposition of simplicial complexes," in *Proceedings of the Web Conference 2021*, 2021, pp. 3408–3418.

[70] B. Yang, D. Wen, L. Qin, Y. Zhang, L. Chang, and R. Li, "Index-based optimal algorithm for computing k-cores in large uncertain graphs," in *Proceedings of the IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 64–75.

[71] F. Esfahani, J. Wu, V. Srinivasan, A. Thomo, and K. Wu, "Fast truss decomposition in large-scale probabilistic graphs." in *Proceedings of the 22nd International Conference on Extending Database Technology (EDBT)*, 2019, pp. 722–725.

[72] Z. Sun, X. Huang, J. Xu, and F. Bonchi, "Efficient probabilistic truss indexing on uncertain graphs," in *Proceedings of the Web Conference 2021*, 2021, pp. 354–366.