

# SaPHyRa: A Learning Theory Approach to Ranking Nodes in Large Networks

Phuc Thai

Virginia Commonwealth University  
thaipd@vcu.edu

My T. Thai

University of Florida  
mythai@cise.ufl.edu

Tam Vu

Oxford University  
tam.vu@cs.ox.ac.uk

Thang Dinh

Virginia Commonwealth University  
tndinh@vcu.edu

**Abstract**—Ranking nodes based on their centrality stands a fundamental, yet, challenging problem in large-scale networks. Approximate methods can quickly estimate nodes’ centrality and identify the most central nodes, but the ranking for the majority of remaining nodes may be meaningless. For example, ranking for less-known websites in search queries is known to be noisy and unstable.

To this end, we investigate a new node ranking problem with two important distinctions: a) *ranking quality*, rather than the centrality estimation quality, as the primary objective; and b) *ranking only nodes of interest*, e.g., websites that matched search criteria. We propose Sample space Partitioning Hypothesis Ranking, or SaPHyRa, that transforms node ranking into a hypothesis ranking in machine learning. This transformation maps nodes’ centrality to the expected risks of hypotheses, opening doors for theoretical machine learning (ML) tools. The key of SaPHyRa is to partition the sample space into exact and approximate subspaces. The exact subspace contains samples related to the nodes of interest, increasing both estimation and ranking qualities. The approximate space can be efficiently sampled with ML-based techniques to provide theoretical guarantees on the estimation error. Lastly, we present SaPHyRa<sub>bc</sub>, an illustration of SaPHyRa on ranking nodes’ *betweenness centrality* (BC). By combining a novel bi-component sampling, a 2-hop sample partitioning, and improved bounds on the Vapnik–Chervonenkis dimension, SaPHyRa<sub>bc</sub> can effectively rank any node subset in BC. Its performance is up to 200x faster than state-of-the-art methods in approximating BC, while its rank correlation to the ground truth is improved by multifold.

**Index Terms**—ranking subset, centrality, betweenness centrality, sampling, VC dimensions

## I. INTRODUCTION

Ranking nodes in a network poses a fundamental problem in network analysis, resulting in various centrality measures from degree centrality [1], closeness centrality [2], betweenness centrality [3], to Google’s PageRank [4]. It finds applications in identifying influential users in social networks, analyzing location in urban networks, characterizing brain networks [1, 5], and so on.

In large networks the exact computation of centrality is intractable, thus, approximate methods [2, 3, 6, 7] have been developed to quickly estimate nodes’ centrality. Those methods are effective in identifying the most central nodes, however, the induced ranking for the majority of remaining nodes is often inaccurate. For nodes with small centrality values, a small error in estimation can result in a large perturbation in ranking, as seen in the ranking for less-known websites in search results [8]. The same challenge in ranking also arises from analyzing locations of lesser centrality in urban networks in which a

majority of nodes have small betweenness centrality [9]. Thus, there is a *lack of approximate methods that provide an accurate ranking of nodes in large-scale networks*.

Moreover, most approximate methods to estimate centrality often produce the estimation for all nodes in the network [10, 11, 3, 12], even when only ranking for a small subset of nodes is required. This often leads to the analysis of separate subnetworks, cut-off from a large network [9], risking inaccurate assessment of nodes centrality in the complete network. *Is it possible to design methods to rank node subsets substantially faster than ranking all nodes in the network?*

To this end, we investigate a new node ranking problem, called subset ranking, with two important distinctions: a) *ranking quality*, rather than the estimation quality, as the primary objective; and b) *ranking only target nodes*, e.g., websites matched search criteria.

Our proposed solution is a framework, called Sample space Partitioning Hypotheses Ranking, or SaPHyRa, that transforms ranking nodes into ranking hypotheses by mapping nodes’ centrality to the expected risks of hypotheses, following the empirical risk minimization framework in machine learning [13]. In SaPHyRa, we partition the sample space into exact and approximate subspaces and combine the evaluations of the hypothesis in both subspaces. The exact space contains samples that are directly linked to the target nodes, providing close “heuristic” estimation of the nodes’ centrality. Further, the risks of the hypothesis in the approximate spaces can be effectively estimated using *statistical learning theory* tools, including Vapnik–Chervonenkis (VC) dimension [14] and empirical Bernstein [13].

Lastly, we demonstrate the proposed framework on the task of ranking nodes’ using betweenness centrality. Given a graph  $G = (V, E)$ , betweenness centrality (BC) [15, 16] defines the importance of each node  $u \in V$  through the fraction of all-pairs shortest paths passing through  $u$ . Methods to approximate BC can be divided into two large groups: heuristics to either relax the shortest paths [17, 18] or estimate BC via a subset of (non-uniformly) selected shortest paths [10, 19, 20] and sampling-based methods with additive error guarantees [10, 11, 3, 12]. Unfortunately, the recent benchmark [21] using 96,000 CPUs and roughly 400 TB RAM indicates that existing methods either do not scale to large networks, e.g., Orkut network with 125 million edges, or produce *poor ranking quality*.

In contrast, our proposed solution, called SaPHyRa<sub>bc</sub>, offers

both substantially better ranking quality and scalability. First, SaPHyRa<sub>bc</sub> has a new sampling method, called bi-component sampling. Inspired by the approach to compute exact BC in [22], our new sampling limits the attention to only the shortest paths with both ends belonging to the same bi-component, thus, reduces the complexity of the sample space. Second, SaPHyRa<sub>bc</sub> deploys a 2-hop-based sample partitioning to guarantee non-zero estimation for nodes with small centrality. Third, SaPHyRa<sub>bc</sub> has a smaller sample complexity by reducing the VC dimension from  $O(\log VD(G))$  in [6] to  $O(1)$  in many scenarios. Our experiments on large networks show that SaPHyRa<sub>bc</sub> is up to 200x faster than state-of-the-art methods in approximating BC, while its rank quality is improved by multifold.

Our contributions are summarized as follows:

- We propose a novel formulation of ranking a node subset in large networks with the focus on the ranking quality and time-saving in ranking subset (but not all nodes). We also propose SaPHyRa, a general framework to effectively rank nodes, especially, when nodes have small centrality values. SaPHyRa provides an  $(\epsilon, \delta)$ -estimation for the nodes of interest, using fewer samples, yet, with higher ranking quality, thanks to its sample space partitioning strategy.
- We propose SaPHyRa<sub>bc</sub>, an illustration of SaPHyRa for ranking nodes using betweenness centrality. SaPHyRa<sub>bc</sub> significantly improved ranking quality in comparisons to the state-of-the-art BC approximate methods. It also provides new VC-dimension bounds, i.e., tighter sample complexity.
- We perform comprehensive experiments on both real-world and synthesis networks with sizes up to 100 million nodes and 2 billion edges. Our experiments indicate the superior of SaPHyRa<sub>bc</sub> algorithm in terms of both accuracy and running time in comparison to the state-of-the-art algorithms.

**Organization.** The rest of the paper is organized as follows: In section 2, we introduce the Ranking Subset and the Hypotheses Ranking problems. We propose SaPHyRa framework to solve Hypotheses Ranking problem in section 3. To rank nodes using betweenness centrality, we develop SaPHyRa<sub>bc</sub> algorithm in section 4. In section 5, we present empirical evidence on the efficacy of SaPHyRa<sub>bc</sub> algorithm (and the proposed SaPHyRa framework).

**Related work.** A few methods focus on estimating the rank without first computing the exact values of the centrality. In [23, 24], Saxena et al. estimate the rank of nodes based on their closeness centrality. In [25, 26, 27], heuristics are proposed to compute centrality based only on localized information restricted to a limited neighborhood around each node. Several recent works [28, 29, 30, 31] aim to approximate node centrality for large networks using neural networks and graph embedding techniques. Notably, [32] demonstrate the multitask learning capability of the model to allow the ability to learn multiple centralities in the same model. While these heuristics can quickly estimate the nodes' ranking, there are no guarantees

on the estimation errors. In contrast, we aim for fast ranking estimation with theoretical bounds on the estimation error.

Exact computation of betweenness centrality takes  $O(MN)$  times in unweighted networks [33]. Bader et al. [3] introduce an adaptive sampling algorithm to reduce the number of single-source shortest paths. Riondato et al. introduce a different sampling method that samples node pairs, resulting in faster computation with the same probabilistic guarantees on the estimation quality. In an algorithm called ABRA, The sampling method is further fine-tuned using Rademacher averages in [6]. KADABRA is proposed by Borassi et al. [12] to speed up the sample generation via a new Bread-first-search approach. These approaches provide approximate betweenness centrality for all nodes in the network with rigorous theoretical guarantees on the additive estimation errors. Unfortunately, the estimated centrality values result in poor ranking as shown in our experiments. Further, there is little saving in computational effort to estimate the centrality values for just a few nodes, compared to that for all the nodes in the network.

Many advances in developing parallel and distributed algorithms to compute and estimate centrality in networks [3, 34, 35, 36, 37]. We note that our effort in ranking nodes here is orthogonal to these efforts. Our sampling framework can be potentially combined with parallel and distributed methods to boost scalability.

## II. PRELIMINARIES

Consider a network, abstracted as a graph  $G = (V, E)$  with  $n = |V|$  nodes and  $m = |E|$  edges. In our *ranking subset problem*, we wish to rank the nodes in a subset  $A \subseteq V$  according to a centrality measure  $c(\cdot)$ . We focus on the case that it is computationally intractable to compute the exact centrality values  $c(v)$  for  $v \in A$ , e.g., when the network has billions of edges or nodes. However, we should be able to estimate the centrality measure with some guaranteed error through sampling.

### A. Ranking subset problem (RSP)

Let  $A \subseteq V$  be a subset of nodes, called *target nodes*, that we wish to rank using some centrality measure  $c(\cdot)$ . Our goal is to produce a ranking of the nodes in  $A$  that is close to the ground truth ranking.

To produce the ranking of the nodes, we estimate the centrality values based on a set of  $N$  samples  $x_1, \dots, x_N$  and a function  $g(\cdot, \cdot)$ . For each node  $v \in A$ , we compute the approximation

$$\tilde{c}(v) = \frac{1}{N} \sum_{i=1}^n g(v, x_i).$$

For example, consider the betweenness centrality that measures the importance of nodes in a network based on the fraction of shortest paths that pass through them. To estimate the betweenness centrality [6], we generate each sample  $x_i$  ( $i \in [N]$ ) as a random shortest path. More precisely, we first randomly select a pair of nodes  $(u, v)$  in  $V$ . Then, we randomly select a shortest path  $p$  between  $u, v$  and set  $x_i = p$ . The function  $g(v, x_i)$  is a binary function that outputs 1 if  $v$  is an inner node of  $x_i$ .

Another example is  $k$ -path centrality [38]. The  $k$ -path centrality measures the importance of nodes based on the fraction of  $k$ -hop paths that pass through them. In  $k$ -path centrality estimation [38], each sample  $x_i$  ( $i \in [N]$ ) is a random path that consists of at most  $k$  edges. Here, we first randomly select a node  $u$ . Then, we perform an  $l$ -hop (where  $l \leq k$ ) random walk from  $u$  and set  $x_i$  as the  $l$ -hop random walk. The function  $g(v, x_i)$  is a binary function that outputs 1 if  $v$  belongs to  $x_i$ .

We measure the quality of an estimation based on the *ranking quality* and the *estimation quality*.

a) *Ranking quality*: To measure the ranking quality, we adopt Spearman's rank correlation [39]. Other rank correlation measures such as Kendall's  $\tau$  [40] can also be used.

Let  $A = \{v_1, v_2, \dots, v_k\}$  where  $k = |A|$  is the size of  $A$ . Denote by  $\mathbf{c} = \langle c(v_1), c(v_2), \dots, c(v_k) \rangle$  and  $\tilde{\mathbf{c}} = \langle \tilde{c}(v_1), \tilde{c}(v_2), \dots, \tilde{c}(v_k) \rangle$  the centrality vector and the approximate centrality vector, respectively.

As the ranks of the nodes are distinct integers between 1 and  $k = |A|$ , the rank correlation can be computed using the following simple formula [39]

$$r_s = 1 - \frac{6 \sum_{i=1}^k dr_i^2}{k(k^2 - 1)}, \quad (1)$$

where  $dr_i$  is the difference between the actual rank of  $c(v_i)$  in  $\mathbf{c}$  and the rank of its estimation  $\tilde{c}(v_i)$  in  $\tilde{\mathbf{c}}$ .

b) *Estimation quality*: We say  $\tilde{\mathbf{c}}$  is an  $(\epsilon, \delta)$ -estimation of  $\mathbf{c}$  if and only if

$$\Pr[\forall v_i \in A, |c(v_i) - \tilde{c}(v_i)| < \epsilon] \geq 1 - \delta. \quad (2)$$

To obtain an  $(\epsilon, \delta)$ -estimation of  $\mathbf{c}$ , it requires  $O(\frac{1}{\epsilon^2} (\log n + \log \frac{1}{\delta}))$  number of sample on the full network and  $O(\frac{1}{\epsilon^2} (\log |A| + \log \frac{1}{\delta}))$  on the subset  $A$ .

## B. RSP as a hypothesis ranking problem

We provide the mapping from RSP to the hypotheses ranking problem, a fundamental problem in machine learning.

a) *Hypothesis ranking (HR) problem*: Consider a (discrete) sample space  $\mathcal{X}$  and a distribution  $\mathcal{D}$  over  $\mathcal{X}$ , where the probability of each sample  $x \in \mathcal{X}$  is denoted as  $\Pr_{x_0 \sim \mathcal{D}}[x_0 = x]$ . Each sample  $x \in \mathcal{X}$  is mapped with a label  $y = f(x)$ . Here,  $f: \mathcal{X} \rightarrow \mathcal{Y}$  labeling function. We will restrict the label space  $\mathcal{Y}$  to be a two-element set, usually  $\{0, 1\}$  or  $\{-1, +1\}$ . In a machine learning problem, the algorithm needs to learn a function  $h: \mathcal{X} \rightarrow \mathcal{Y}$ , called hypothesis, which outputs a label  $y$  for a sample  $x$ .

We use a non-negative real-valued *loss function*  $L(y', y)$  to measure the difference between the output  $y'$  of a hypothesis and the true label  $y$ . The *expected risk* of a hypothesis  $h$  is defined as the expectation of the loss function, i.e.,

$$\mathcal{R}(h) \stackrel{\text{def}}{=} \sum_{x \in \mathcal{X}} \Pr_{x_0 \sim \mathcal{D}}[x_0 = x] L(h(x), f(x))$$

Given a set of  $k$  hypotheses  $\mathcal{H} = \{h_1, \dots, h_k\}$ . Our goal now is to rank the hypotheses based on the expected risk. That is to compute an  $(\epsilon, \delta)$  approximation of the expected risks that can provide a high rank correlation to the expected risks.

*Mapping from RSP to Hypothesis ranking*. The ranking subset problems in which there exist sampling-based method to estimate the centrality. In that case, we can design the

hypotheses so that the centrality value  $c(v_i)$  equals the expected risk of  $h_i$ . Ranking the nodes is now equivalent to ranking the hypotheses based on their expected risks.

## C. Ranking subset based on betweenness centrality ( $RSP_{bc}$ )

Betweenness centrality (BC) of a node  $v \in V$ , denoted by  $bc(v)$ , measures the transitivity of  $v$ , i.e., how frequently  $v$  lies on shortest paths among other nodes. Mathematically, define

$$bc(v) = \frac{1}{n(n-1)} \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}, \quad (3)$$

where  $\sigma_{st}$  denotes the number of shortest paths from  $s$  to  $t$  and  $\sigma_{st}(v)$  denotes the number of shortest paths from  $s$  to  $t$  that  $v$  lies on, respectively.

Computing exact BC takes  $O(mn)$  [10], where  $n$  and  $m$  are the number of nodes and edges in the graph, respectively. Thus, it is intractable for large networks.

To approximate the BC, several sampling methods have been proposed, mapping the BC value of a node  $v$  to the expectation of whether  $v$  will lie on a random shortest path. By treating the probability that  $v$  lies on a random shortest path as an expected risk, we can turn the ranking node subset based on BC into a hypothesis ranking problem.

*Mapping from  $RSP_{bc}$  to Hypothesis ranking*. The SP sample space consists of all shortest paths between two nodes in the graphs. To be precise, let  $P_{st}$  be the set of all shortest paths from  $s$  to  $t$  in the graph. The sample space is defined as follows.

$$\mathcal{X}_b = \{p | (s, t) \in V, p \in P_{st}\}. \quad (4)$$

The SP distribution  $\mathcal{D}_b$  is a distribution over the SP sample space, where the probability of a shortest path  $p$  from  $s$  to  $t$  is

$$\Pr_{x \sim \mathcal{D}_b} [x = p] = \frac{1}{n(n-1)} \frac{1}{\sigma_{st}}. \quad (5)$$

For a shortest path  $p$  from  $s$  to  $t$ , we refer all nodes  $v \in p$ , except  $s, t$ , as *inner nodes*. We define a function  $g_v: \mathcal{X}_{bc} \rightarrow \{0, 1\}$  as follows

$$g(v, p) = \begin{cases} 1 & \text{If } v \text{ is an inner node in } p \\ 0 & \text{Otherwise} \end{cases} \quad (6)$$

Given the target nodes  $A \subseteq V$ , we define for each node  $v \in A$  a hypothesis  $h_v$  that maps each random path  $p$  to a value  $h_v(p) = g(v, p)$ . We use 0-1 loss function and choose the labeling function  $f$  that always output 0 for any path  $p$ , i.e.,  $L(h_v(p), f(p)) = \mathbb{1}_{h_v(p)=f(p)} = g(v, p)$ . Following [6], betweenness centrality of a node  $v$  equals the expected risk  $\mathcal{R}(h_v)$ .

**Lemma 1.** For any node  $v \in V$ , we have

$$\mathbb{E}_{p \sim \mathcal{D}_b} L(h_v(p), f(p)) = \mathbb{E}_{p \sim \mathcal{D}_b} h_v(p) = bc(v)$$

### III. SAPHYRA: SAMPLE SPACE PARTITIONING HYPOTHESES RANKING

In this section, we present the sample space partitioning (SaPHyRa) framework to solve the *hypothesis ranking* (HR) problem. An application of the SaPHyRa framework in ranking nodes in a subset using betweenness centrality will be presented later in Section IV. Note that, due to the space limitation, here, we omit the proofs of the lemmas. The detailed proofs are presented in the Appendix of the full version [41].

#### A. Direct estimation.

An efficient solution is to estimate the expected risks through sampling and rank the hypotheses based on the estimation. To be precise, consider a sequence of  $N$  samples  $\mathbf{x} = (x_1, \dots, x_N)$  where,  $x_i \sim \mathcal{D}, \forall i \in [N]$ . Here, we write  $x \sim \mathcal{D}$  to mean that  $x$  is drawn from the distribution  $\mathcal{D}$ .

The estimation of a hypothesis  $h \in \mathcal{H}$  over  $X$  is computed as

$$\mathcal{R}_e(h) \stackrel{\text{def}}{=} \frac{1}{N} \left( \sum_{i=1}^N L(h(x_i), f(x_i)) \right)$$

*Sampling complexity.* The goal here is to find the number of samples  $N$  to ensure that with a probability of at least  $1 - \delta$  (where  $\delta \in (0, 1)$ ), the difference between the estimation and the expected risk of any hypothesis is smaller than  $\epsilon \in (0, 1)$ , i.e.,

$$\Pr[\forall h \in \mathcal{H}, |\mathcal{R}(h) - \mathcal{R}_e(h)| < \epsilon] \geq 1 - \delta \quad (7)$$

In this case, we refer to this as an  $(\epsilon, \delta)$ -estimation of the expected risks. With  $N = O(\frac{1}{\epsilon^2} (\log |A| + \log \frac{1}{\delta}))$ , we can guarantee an  $(\epsilon, \delta)$ -estimation of the expected risks.

**Challenge in ranking hypotheses with low expected risks.** Ranking the hypotheses with low expected risk usually requires many more samples. Assume that we can only afford to generate enough samples to guarantee an  $(\epsilon, \delta)$ -estimation of the expected risks due to a time limit. Consider the two hypotheses that have the expected risks of  $\mu_1, \mu_2$ , respectively. If  $\mu_1, \mu_2 < \epsilon$ , i.e.,  $\epsilon > |\mu_1 - \mu_2|$ , there will be a high chance that the relative ranking of the two hypotheses is incorrect.

#### B. Sample space partitioning framework

Our sample space partitioning (SaPHyRa) framework partitions the sample space  $\mathcal{X}$  into two disjoint subspaces  $\mathcal{X} = \hat{\mathcal{X}} \cup \tilde{\mathcal{X}}$  where  $\hat{\mathcal{X}}$  and  $\tilde{\mathcal{X}}$  are called *exact subspace* and *approximate subspace*, respectively.

The partition is done so that the exact subspace  $\hat{\mathcal{X}}$  will contain samples that are directly linked to the hypotheses in  $\mathcal{H}$ , aiding the estimation of the expected risks. Especially, this will resolve the above challenge in estimating the hypotheses with low expected risks. The approximate subspace contains the majority of the samples, providing  $(\epsilon, \delta)$ -estimation guarantees for the expected risks. Combining them together, we have an estimation that provide both high ranking quality and theoretical guarantee on the estimation error.

Specifically, for each hypothesis  $h_i$ , we combine the expected risk the expected risks of  $h_i$  on the exact subspace  $\hat{\ell}_i$  and the estimation  $\tilde{\ell}_i$  of the expected risks on the approximate subspace as follows

$$\ell_i = \hat{\ell}_i + \lambda \tilde{\ell}_i, \quad (8)$$

---

#### Algorithm 1: Sample Space Partitioning (SaPHyRa)

---

**Input** : A sample space  $\mathcal{X}$ , a probability distribution  $\mathcal{D}$ , a labeling function  $f$ , and a hypothesis class  $\mathcal{H} = \{h_1, h_2, \dots, h_k\}$ . Parameter  $\epsilon, \delta \in (0, 1)$

**Output** : The rank of the hypotheses based on the expected risks

- 1 Partition  $\mathcal{X} = \hat{\mathcal{X}} \cup \tilde{\mathcal{X}}$ ;
- 2  $\tilde{\mathcal{D}}$  be the distribution over  $\tilde{\mathcal{X}}$  as in Eq. 10
- 3  $(\hat{\lambda}, \hat{\ell}_1, \hat{\ell}_2, \dots, \hat{\ell}_k) \leftarrow \text{Exact}(\cdot)$  ▷ Compute the exact value in exact subspace  $\hat{\mathcal{X}}$
- 4  $\lambda \leftarrow 1 - \hat{\lambda}$
- 5  $\epsilon' = \epsilon/\lambda$
- 6  $N_0 \leftarrow \frac{c}{\epsilon'^2} \ln 1/\delta$  ▷ the initial #samples
- 7  $N_{max} \leftarrow \frac{c}{\epsilon'^2} (VC(\mathcal{H}) + \ln 1/\delta)$  ▷ the maximum #samples
- 8  $N \leftarrow N_0$  and  $\mathbf{x} \leftarrow \emptyset$
- 9 Allocate the probability error  $\delta_i$  such that Eq. 13 is satisfied
- 10 **for**  $rd := 1$  to  $\lceil \log(\frac{N_{max}}{N_0}) \rceil$  **do**
- 11     **for**  $j := |\mathbf{x}| + 1$  to  $N$  **do**
- 12         Generate samples  $x_j \sim \tilde{\mathcal{D}}$  using  $\text{Gen}(\cdot)$  and add  $x_j$  to  $\mathbf{x}$
- 13     **for**  $i := 1$  to  $k$  **do**
- 14          $\mathbf{z}^{(i)} \leftarrow [z_j^{(i)} = L(h_i(x_j), f(x_j))]_{j=1..N}$
- 15          $\epsilon_i \leftarrow \varepsilon(N, \delta_i, \text{Var}(\mathbf{Z}^{(i)}))$
- 16         **if**  $\max_{i \in [k]} \epsilon_i \leq \epsilon'$  **then**
- 17             **Break**
- 18          $N = \min(2N, N_{max})$
- 19     **for**  $i := 1$  to  $k$  **do**
- 20          $\tilde{\ell}_i \leftarrow \frac{1}{N} \left( \sum_{j=1}^N L(h_i(x_j), f(x_j)) \right)$
- 21          $\ell_i \leftarrow \hat{\ell}_i + \lambda \tilde{\ell}_i$
- 22 **Return**  $(\ell_1, \ell_2, \dots, \ell_k)$  and the rank of the hypotheses

---

where  $\lambda = \Pr_{x \sim \mathcal{D}}[x \in \tilde{\mathcal{X}}]$  be the probability that a random sample  $x \sim \mathcal{D}$  belongs to the approximate subspace.

**Exact subspace.** We select the exact subspace  $\hat{\mathcal{X}}$  such that the expected risks on the exact subspace should provide good estimations for the hypotheses with small expected risks. At the same time, there should be an algorithm that can efficiently compute the expected risks on the exact subspace. Let  $\hat{\mathcal{L}} = (\hat{\ell}_1, \dots, \hat{\ell}_k)$  be the expected risks of  $\mathcal{H} = \{h_1, \dots, h_k\}$  on exact subspace  $\hat{\mathcal{X}}$ . For each hypothesis  $h_i$  ( $i \in [k]$ ),  $\hat{\ell}_i$  is defined as follow,

$$\hat{\ell}_i \stackrel{\text{def}}{=} \sum_{x \in \hat{\mathcal{X}}} \Pr_{x_0 \sim \mathcal{D}}[x_0 = x] L(h_i(x), f(x)) \quad (9)$$

We assume the expected risks on the exact subspace can be computed via an algorithm **Exact** that returns  $(\hat{\lambda}, \hat{\ell}_1, \hat{\ell}_2, \dots, \hat{\ell}_k)$ , in which, for all  $i \in [k]$ ,  $\hat{\ell}_i$  is computed as in Eq. 9. An efficient partitioning of the sample space will need to provide close estimations for all the hypotheses in  $\mathcal{H}$ .

**Approximate subspace.** The approximate subspace  $\tilde{\mathcal{X}} = \mathcal{X} \setminus \hat{\mathcal{X}}$  contains the samples outside the exact subspace. We assume

that there exists an effective algorithm to draw samples from the approximate subspace  $\tilde{\mathcal{X}}$ . Let  $\tilde{\mathcal{D}}$  be a distribution over the approximate subspace  $\tilde{\mathcal{X}}$ , where the probability of a sample  $x \in \tilde{\mathcal{X}}$  is

$$\Pr_{x_0 \sim \tilde{\mathcal{D}}} [x_0 = x] = \frac{1}{\lambda} \Pr_{x_0 \sim \mathcal{D}} [x_0 = x]. \quad (10)$$

For each hypothesis  $h_i \in \mathcal{H}$  ( $i \in [k]$ ), we denote  $\tilde{\mathcal{R}}(h_i)$  as the expected risk of  $h$  on the distribution  $\tilde{\mathcal{D}}$ , i.e.,

$$\tilde{\mathcal{R}}(h_i) = \sum_{x \in \tilde{\mathcal{X}}} \Pr_{x_0 \sim \tilde{\mathcal{D}}} [x_0 = x] L(h(x), f(x)). \quad (11)$$

Let  $\epsilon' = \epsilon/\lambda$  and  $(\tilde{\ell}_1, \dots, \tilde{\ell}_n)$  be an  $(\epsilon', \delta)$ -estimation of the expected risk on the distribution  $\tilde{\mathcal{D}}$ , i.e.,

$$\Pr \left[ \forall i \in [k], |\tilde{\mathcal{R}}(h_i) - \tilde{\ell}_i| < \epsilon' \right] \geq 1 - \delta. \quad (12)$$

**Lemma 2.** For any partition of  $\mathcal{X} = \hat{\mathcal{X}} \cup \tilde{\mathcal{X}}$ . Let  $(\hat{\ell}_1, \dots, \hat{\ell}_n)$  be the expected risks on exact subspace  $\hat{\mathcal{X}}$  (see Eq.9). Let  $\tilde{\mathcal{R}}(h_i)$  be the expected risk of a hypothesis  $h_i$  on the approximate subspace  $\tilde{\mathcal{X}}$  (see Eq.11). Then, we have

$$\forall i \in [k], \hat{\ell}_i + \tilde{\mathcal{R}}(h_i) = \mathcal{R}(h_i).$$

### C. Risk Estimation in the Approximate Subspace

To estimate the risk within the approximate subspace we adopt two theoretical machine learning tools, namely, *adaptive sampling* and *Vapnik–Chervonenkis (VC) dimension*.

**Adaptive sampling.** We apply empirical Bernstein' inequality [13] to construct an *adaptive sampling* method to estimate expected risk on the approximate subspace.

We allocate the error probability  $\delta_i$  for each hypothesis  $h_i$  such that

$$\sum_{i=1}^k 2\delta_i = \frac{\delta}{\lceil \log(\frac{N_{max}}{N_0}) \rceil}. \quad (13)$$

To minimize the number of iterations, we optimize the allocation of  $\delta_i$  as follows. We first compute a sample variance  $v_i$  for each hypothesis  $h_i$  by taking  $N_0 = \frac{c}{\epsilon^2} \ln 1/\delta$  number of samples. Note that, the samples here are independent with the samples in  $X$ . For each hypothesis  $h_i$ , we use the sample variance  $v_i$  and set  $\epsilon_i = \epsilon$  for all  $\epsilon_i$ , to compute  $\delta_i$  that satisfies Eq.15 (this can be done by binary search). After that, we rescale  $\delta_i$  such that Eq. 13 is satisfied.

Next, we compute the empirical risk. Initially, we have a list of  $N_0$  samples. After each iteration, we double the number of samples and measure the current error probability using empirical Bernstein' inequality [13].

**Lemma 3** (Empirical Bernstein' inequality (Theorem 4 [13])). Let  $\mathbf{z} = (z_1, z_2, \dots, z_N)$  be a vector of independent identically random variables. Let  $\mu$  be the expected value of a random variable  $z_j$  ( $j \in [N]$ ) and  $\delta_0 \in (0, 1)$ . Then, we have,

$$\Pr \left[ \mu - \frac{1}{N} \sum_{j=1}^N z_j \leq \epsilon(N, \delta_0, Var(\mathbf{z})) \right] \geq 1 - \delta_0, \quad (14)$$

where

$$\epsilon(N, \delta_0, Var(\mathbf{z})) = \sqrt{1/N \left( 2Var(\mathbf{z}) \ln \frac{2}{\delta_0} \right)} + 7 \ln \frac{2}{\delta_0} / 3N$$

and  $Var(\mathbf{z})$  is the sample variance

$$Var(\mathbf{z}) = \frac{1}{N(N-1)} \sum_{1 \leq j_1 < j_2 \leq N} (z_{j_1} - z_{j_2})^2.$$

In Lemma 3, we only show one-sided error. We can show the errors on both sides by considering the random variables  $z'_j = 1 - z_j$ . Then, using union bound, we have

$$\Pr \left[ \left| \mu - 1/N \sum_{j=1}^N z_j \right| \leq \epsilon(N, \delta_0, Var(\mathbf{z})) \right] \geq 1 - 2\delta_0$$

Let  $\mathbf{x} = (x_1, \dots, x_N)$  be the current list of samples. For each hypothesis  $h_i$ , let  $\mathbf{z}^{(i)} = (z_1^{(i)}, \dots, z_N^{(i)})$  be the list of random variables for  $h_i$ , where

$$z_j^{(i)} = L(h_i(x_j), f(x_j))$$

We compute the error  $\epsilon_i$  for each hypothesis  $h_i$  base on  $\delta_i$  and sample variance  $V(\mathbf{z}^{(i)})$  as in Lemma 3, i.e.,

$$\epsilon_i = \epsilon(N, \delta_i, Var(\mathbf{z}^{(i)})) \quad (15)$$

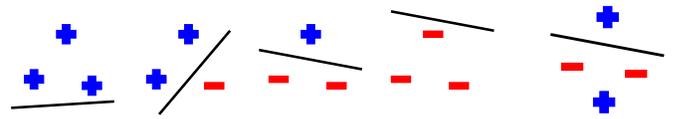
If the maximum value of  $\epsilon_i$  is smaller than or equal to  $\epsilon'$ , we stop the algorithm and take the current estimation of the expected risks.

**Reducing sample complexity using VC-dimension.** We can reduce the number of samples using VC-dimension, a standard complexity measure for concept classes in probably approximately correct (PAC) learning [14].

Intuitively, VC-dimension measures the capacity of a hypothesis class. It is defined as the cardinality of the largest set of points that the algorithm can shatter.

**Definition 1** (VC-dimension). For any subset  $S = \{x_1, \dots, x_w\} \subseteq \mathcal{X}$ , let  $\mathcal{H}_S = \{(h(x_1), \dots, h(x_w)) : h \in \mathcal{H}\}$  be the restriction of  $\mathcal{H}$  to  $S$ . We say  $\mathcal{H}$  shatters  $S$  if the restriction of  $\mathcal{H}$  to  $S$  is the set of all functions from  $S$  to  $\{0, 1\}$ , i.e.,  $|\mathcal{H}_S| = 2^w$ . The VC-dimension of a hypothesis class  $\mathcal{H}$ , denoted  $VC(\mathcal{H})$  is the maximum size of a set  $S \subseteq \mathcal{X}$  that can be shattered by  $\mathcal{H}$ .

For example, consider a two-dimensional binary classification problem where we need to label each point in a two-dimensional plane as positive (or 1) or negative (or 0). Here, each hypothesis  $h \in \mathcal{H}$  is a straight line in which all data points above the line are labeled as positive and all data points below the line are labeled as negative. We say the hypothesis class  $\mathcal{H}$  shatters a set of points (samples)  $S$  if for any labeling on  $S$ , we can always find a hypothesis to separate positive data points from negative data points. We now show that the VC-dimension  $VC(\mathcal{H}) = 3$ . Indeed, there exist a set of 3 points that can be shattered by  $\mathcal{H}$  (see Fig. 1a)). Plus, by Radon's theorem [42], we can divide the four points into two subsets with intersecting convex hulls. Thus, it is not possible to separate the two subsets with a straight line (see Fig. 1b)).



a) A set of 3 points that can be shattered. We can always find a hypothesis to separate positive data points from negative data points.

b) Any set of 4 points cannot be shattered.

Fig. 1: The VC-dimension of the class  $\mathcal{H}$  of threshold functions.

We can bound the number of samples using VC-dimension as follows.

**Lemma 4** (THEOREM 6.8 in [14]). *Given a hypothesis space  $H$ , defined over sample space  $\mathcal{X}$ , such that  $h_i : \mathcal{X} \rightarrow \mathcal{Y}$  for  $\forall h_i \in H$ .  $VC(\mathcal{H})$  is the VC dimension of  $H$ . Let  $\mathbf{x} = (x_1, \dots, x_N)$  be a collection of independent identically distributed random events, taken from sample space  $\mathcal{X}$ , selected by probability  $p$ . We can obtain an  $(\epsilon, \delta)$ -estimation of the expected risk with*

$$N = \frac{c}{\epsilon^2} (VC(\mathcal{H}) + \ln \frac{1}{\delta}), \quad (16)$$

where constant  $c$  is approximately 0.5.

Thus, we can obtain an  $(\epsilon, \delta)$ -estimation of the expected risk with  $N = \frac{c}{\epsilon^2} (SC(\mathcal{H}) + \ln \frac{1}{\delta})$  samples.

Now, we present a bound for the VC-dimension of a hypothesis class  $\mathcal{H}$ .

**Lemma 5.** *For a sample  $x \in \mathcal{X}$ , let  $\pi(x)$  be the number of hypotheses  $h \in \mathcal{H}$  such that  $h(x) = 1$ . Let  $\pi_{\max} = \max_{x \in \mathcal{X}} \pi(x)$ .*

$$VC(\mathcal{H}) \leq \lfloor \log(\pi_{\max}) \rfloor + 1.$$

#### D. Correctness and Complexity.

We first state the correctness of our framework in providing theoretical guarantee on the estimation error.

**Theorem 6.** *Consider a distribution  $\mathcal{D}$  over the sample space  $\mathcal{X}$ , where the probability of a sample  $x \in \mathcal{X}$  is  $p(x)$ , a label space  $\mathcal{Y}$ , a label function  $f$ , and a hypothesis class  $\mathcal{H} = \{h_1, \dots, h_k\}$ . If Algorithm **Exact** returns the expected risks on the exact subspace as in Eq. 9, and Algorithm **Gen** return a random sample  $x \sim (\mathcal{X}, p)$ . Then,  $(\ell_1, \ell_2, \dots, \ell_k)$ , returned by Algorithm 1, is an  $(\epsilon, \delta)$ -estimation of the expected risks i.e.,*

$$\Pr [\forall i \in [k], |\mathcal{R}(h_i) - \ell_i| < \epsilon] \geq 1 - \delta.$$

The sample complexity is a function of the weight of the approximate space, i.e.,  $\lambda$  and the VC-dimension of the hypotheses  $VC(\mathcal{H})$ .

**Lemma 7.** *The worst-case number of samples in Algorithm 1 is*

$$\frac{c\lambda^2}{\epsilon^2} (VC(\mathcal{H}) + \ln 1/\delta), \quad (17)$$

which is reduced by a factor of  $1/\lambda^2$ , in comparison with the direct estimation approach.

*Variance reduction analysis.* We show that our SaPhyRa framework results in random variables with generally smaller variances. Thus, the expected risks can be estimated with fewer samples. From Eq. 15, if the variance is reduced by a factor of  $\alpha$ , we can also reduce the number of samples by a factor of approximately  $\alpha$  to achieve the same error. Indeed, in Eq. 15, the first part is  $\Theta(\sqrt{N})$  times bigger than the second part. Thus, if  $N$  is big enough, we can approximate

$$N \approx \epsilon_i^2 \left( 2\text{Var} \left( \mathbf{z}^{(i)} \right) \ln \frac{2}{\delta_i} \right).$$

Let  $Z^{(i)}$  be the random variable that represents the output of the loss function of  $h_i$  on a sample that is drawn from the distribution  $\hat{\mathcal{D}}$ . Let  $Z'^{(i)}$  be the random variable that represents the output of the loss function of  $h_i$  on a sample that is drawn from the distribution  $\mathcal{D}$ . Let  $\hat{\ell}_i = \hat{\mu}_i$  be expected risk on

the exact subspace of  $h_i$ . Let  $\text{Var}(Z^{(i)})$ ,  $\text{Var}(Z'^{(i)})$  be the variances of  $Z^{(i)}$ ,  $Z'^{(i)}$ . The expected value of  $Z'^{(i)}$  and  $Z^{(i)}$  are  $\mu_i$  and  $\mu_i - \hat{\mu}_i$ , respectively. Recall that, in this work, we consider 0-1 loss function. Thus,  $Z'^{(i)}$  and  $Z^{(i)}$  Bernoulli random variable. Hence, we have,  $\text{Var}(Z'^{(i)}) = \mu_i(1 - \mu_i)$ , and  $\text{Var}(Z^{(i)}) = (\mu_i - \hat{\mu}_i)(1 - \mu_i + \hat{\mu}_i)$ . Therefore, the reduction in the variance is

$$\frac{\text{Var}(Z^{(i)})}{\text{Var}(Z'^{(i)})} = \frac{\mu_i(1 - \mu_i)}{(\mu_i - \hat{\mu}_i)(1 - \mu_i + \hat{\mu}_i)}$$

**Claim 8.** *Consider a hypothesis  $h_i \in \mathcal{H}$  in which the expected risk  $\mu_i < 1/2$ . We have,  $\text{Var}(Z^{(i)}) < \text{Var}(Z'^{(i)})$ . Specifically, if  $\mu_i \ll 1$ , we have*

$$\frac{\text{Var}(Z^{(i)})}{\text{Var}(Z'^{(i)})} \approx \frac{\mu_i}{\mu_i - \hat{\mu}_i}$$

In this work, we assume the expected risk  $\mu_i$  of any hypothesis  $h_i \in \mathcal{H}$  is smaller than  $1/2$ . Here,  $1/2$  is the expected risk of a baseline hypothesis that randomly output 0 and 1 with the same probability. In this case,  $\frac{\mu_i}{\mu_i - \hat{\mu}_i} < 1$ . From claim 8, the random variables in SaPhyRa framework have smaller random variables. In other words, SaPhyRa framework uses a smaller number of samples to achieve the same error guarantee.

#### IV. SAPHYRA<sub>bc</sub>: RANKING NODE SUBSET WITH BETWEENNESS CENTRALITY

In this section, we describe SaPhyRa<sub>bc</sub> algorithm, an application of SaPhyRa framework for RSP<sub>bc</sub>. We start by introducing an auxiliary sample space, call *intra-component shortest path (ISP)*. The ISP sample space is constructed by breaking shortest paths to intra-component shortest paths in which all nodes must belong to the same bi-component. In SaPhyRa<sub>bc</sub> algorithm, we select the exact subspace as the set of all 2-hop shortest paths that go through a node in the subset of target nodes and propose an algorithm that efficiently computes the expected risks on the exact subspace. For the approximate subspace, use the sampling method that is described in Subsection III-C to estimate the expected risk. We also present a tight bound on the VC dimension that is based on the characteristics of the subset.

##### A. Sample Space for RSP<sub>bc</sub>

We start by introducing an auxiliary sample space, termed *intra-component shortest paths (ISP)*, that contains the shortest paths with all nodes belong to the the same bicomponent.

Given the target nodes, our sample space will be a “personalized” version of the ISP, obtained by removing shortest paths that have no connections to the target nodes.

**ISP sample space.** The ISP sample space consists of *intra-component shortest paths*, i.e., the shortest path in which source node and the destination node belong to the same *bi-component* [43].

$$\mathcal{X}_c = \bigcup_{s', t' \text{ in same bi-component}} P(s', t')$$

*Bi-component and block-cut tree.* A bi-component is a maximal “nonseparable” subgraph, i.e., if any one node is removed, the subgraph still remain connected. We denote  $\mathbf{C} = \{C_1, C_2, \dots, C_\ell\}$  ( $\ell \geq 1$ ) as the set of bi-components

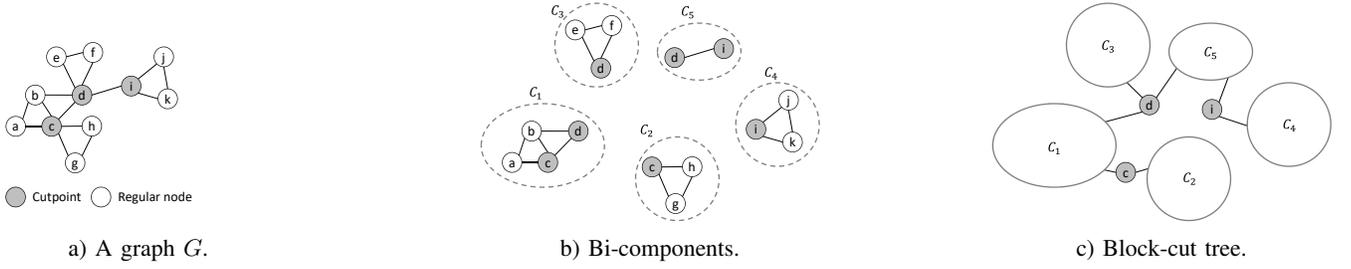


Fig. 2: Bi-components in a graph  $G$  a) Cutpoints whose removal increase the number of bi-components b) Five bi-components in  $G$  c) Block-cut tree of  $G$ , created by adding a vertex for each bi-component and each cutpoint in  $G$ , and adding an edge for each pair of a bi-component and a cutpoint that belongs to that bi-component.

of  $G$ . Usually, each node in the graph belongs to exactly one bi-component. The nodes that belong to more than one bi-component are referred as *cutpoints*. The removal of a cutpoint makes the graph disconnected. The structure of the bi-components and cutpoints can be described by a tree  $G_T = (V_T, E_T)$ , called the *block-cut tree* [44]. The set of nodes  $V_T$  consists of all bi-components and cutpoints. Each edge in  $E_T$  is a pair of bi-component and a cutpoint that belongs to that bi-component. For example, in Fig. 2,  $V_T = \{C_1, C_2, C_3, C_4, C_5, c, d, i\}$  and  $E_T = \{(c, C_1), (c, C_2), (d, C_1), (d, C_3), (d, C_5), (i, C_4), (i, C_5)\}$ . This tree has a node for each bi-component and for each cutpoint of the given graph. There is an edge in the block-cut tree for each pair of a bi-component and a cutpoint that belongs to that component.

*Breaking a shortest path into intra-component shortest paths.* We construct the ISP sample space by dividing each shortest path into multiple intra-component shortest paths such that two consecutive intra-component shortest paths belong to different bi-component. In other words, an intra-component shortest path  $p' \in \mathcal{X}_c$  in a bi-component  $C_i$  is a part of a shortest path  $p \in \mathcal{X}_b$  if  $p'$  is the maximal intra-component shortest path in  $p$  that belongs to the bi-component  $C_i$ . We denote  $I(p)$  as the set of intra-component shortest paths when we break the shortest path  $p$ .

The ISP distribution  $\mathcal{D}_c$  over the ISP sample space  $\mathcal{X}_c$  is defined as

$$\Pr_{x \sim \mathcal{D}_c} [x = p'] = \frac{1}{\gamma} \times \Pr_{p \sim \mathcal{D}_b} [p' \in I(p)],$$

where  $\gamma$  is the normalizing factor that will be given in Eq.19 to ensure the sum of all probabilities equals 1.

**ISP distribution.** To compute ISP distribution, we introduce the definition of out reach set as follows.

*Out reach set.* The out reach set  $R_i(v)$  of a node  $v$ , regarding to a bi-component  $C_i$  is the set of all nodes  $u$  that can be reach by  $v$  without going through any nodes in  $C_i$ . If  $v$  is not a cutpoint, the out reach set  $R_i(v)$  only consists of the node  $v$  itself. If  $v$  is a cutpoint, the out reach set  $R_i(v)$  consists of the node  $v$  itself and all nodes  $u$  that belong to a bi-component  $C_j$  that can be reach by  $v$  in the block-cut tree  $G_T$  without going through  $C_i$ . We can compute the out reach of all cutpoints with the time complexity of  $O(|V'|)$  using dynamic programming.

**Claim 9.** Consider a bi-component  $C_i \in \mathbf{C}$ , each  $v \in V$  belongs to exactly one out reach set of a node  $u$ , regarding to  $C_i$

For Claim 9, for any bi-component  $C_i \in \mathbf{C}$ , we have

$$\sum_{v \in C_i} r_i(v) = n \quad (18)$$

Given a bi-component  $C_i$ , for any pair of node  $s' \neq t'$  and  $s', t' \in C_i$ , let

**Claim 10.** For any intra-component shortest paths  $p', p'' \in P_{s't'}$ , we have,

$$\Pr_{p \sim \mathcal{D}_b} [p' \in I(p)] = \Pr_{p \sim \mathcal{D}_b} [p'' \in I(p)]$$

Let

$$q_{s't'} = \Pr_{p \sim \mathcal{D}_b} [p' \in P_{s't'} \text{ and } p' \in I(p)]$$

From Claim 10,  $\forall p' \in P_{s't'}$ , we have

$$\Pr_{p \sim \mathcal{D}_b} [p' \in I(p)] = \frac{1}{\sigma_{s't'}} \sum_{p'' \in P_{s't'}} \Pr_{p \sim \mathcal{D}_b} [p'' \in I(p)] = \frac{1}{\sigma_{s't'}} q_{s't'}$$

**Lemma 11.** Consider an intra-component shortest path  $p'$  from  $s'$  to  $t'$ , where  $s', t' \in C_i$ . If  $p'$  is a part of a shortest path from  $s$  to  $t$ , then the  $s \in R_i(s')$  and  $t \in R_i(t')$

**Lemma 12.** Consider two node  $s', t' \in C_i$ . Consider a shortest path  $p$  from  $s$  to  $t$ , where  $s \in R_i(s')$  and  $t \in R_i(t')$ . Then, intra-component shortest path  $p'$  from  $s'$  to  $t'$  such that  $p'$  is a part of  $p$ . Consider an intra-component shortest path  $p'$  from  $s'$  to  $t'$ . If  $p'$  is a part of a shortest  $p$  from  $s$  to  $t$ , then the  $s \in R_i(s')$  and  $t \in R_i(t')$

From Lemma 11 and Lemma 12, we have

$$q_{s't'} = \frac{1}{n(n-1)} |R_i(s')| \times |R_i(t')| = \frac{1}{n(n-1)} r_i(s') r_i(t').$$

Let  $\gamma$  be the sum of  $q_{st}$  for all pair  $s, t$ , i.e.,

$$\gamma = \sum_{i=1}^{\ell} \sum_{s \neq t \in C_i} q_{st} = \frac{1}{n(n-1)} \sum_{i=1}^{\ell} \sum_{s \in C_i} r_i(s) (n - r_i(s)) \quad (19)$$

To compute  $\gamma$ , we iterate through all bi-components and for each bi-component  $C_i$ , we iterate through all nodes in  $C_i$ . This can be done in  $O(n)$ .

For an intra-component shortest path  $p'$  from  $s'$  to  $t'$ , we have

$$\Pr_{x \sim \mathcal{D}_c} [x = p'] = \frac{1}{\gamma} \frac{1}{\sigma_{s't'}} q_{s't'} \quad (20)$$

*Betweenness centrality for cutpoints.* When we break a shortest path into multiple intra-component shortest paths, the target node of the previous intra-component shortest path is the source node of the next intra-component shortest path. We refer to those nodes as *break points*. Since the break points belong to more than one bi-component, they must be cutpoints.

The break points were inner nodes in the original shortest path but they are not accounted when we compute the betweenness centrality on the ISP sample space.

For a cutpoint  $v$ , let  $bc(v)$  be the probability that  $v$  is a break point of a shortest path  $p \in \mathcal{X}_b$ .

$$bc_a(v) = \Pr_{p \sim \mathcal{D}_b} [v \text{ is a break point of } p].$$

**Lemma 13.** For any node  $v \in V$ ,

$$bc(v) = \mathbb{E}_{p \sim \mathcal{D}_b} g(v, p) = \gamma \mathbb{E}_{p \sim \mathcal{D}_c} g(v, p) + bc_a(v),$$

where  $g(v, p) = 1$  if  $v$  is an inner node of  $p$  (see Eq. 6)

Next, we show how to compute the value of  $bc_a(v)$  for a cutpoint  $v$ . Consider the block-cut tree  $G_T$  and take  $v$  as the root node of  $G_T$ . By removing the root node  $v$ , we can divide  $G_T$  into multiple subtrees where the root nodes of those subtrees are the bi-components that  $v$  belongs to. The node  $v$  is a break point of a shortest path  $p$  from  $s$  to  $t$  if the source node  $s$  and the target node  $t$  belong to two different subtrees.

Formally, let  $T_i(v)$  be the set of nodes (except  $v$ ) that belong a bi-component in the subtree in which  $C_i$  is the root node. We have,

$$T_i(v) = V \setminus R_i(v) \text{ and } \bigcup_{i \in [\ell]: v \in C_i} T_i(v) = V \setminus \{v\}.$$

**Lemma 14.** A node  $v \in V$  is a break point of a shortest path  $p$  from  $s \in T_i(v)$  to  $t \in T_j(v)$  if  $i \neq j$ .

From Lemma 14, we have,

$$\begin{aligned} bc_a(v) &= \frac{1}{n(n-1)} \sum_{C_i \in \mathcal{C} \setminus v \in C_i} |T_i(v)| \sum_{C_j \in \mathcal{C} \setminus \{C_i\} \mid v \in C_j} |T_j(v)| \\ &= \frac{1}{n(n-1)} (r_i(v) - 1)(n - r_i(v)) \end{aligned} \quad (21)$$

**Personalized ISP (PISP) Sample Space.** Given a subset  $A \subseteq V$ , we construct a personalized sample space by taking the necessary samples in the ISP sample space. Specifically, the personalized ISP sample space  $\mathcal{X}_c^{(A)}$  consists of all shortest paths from  $s$  to  $t$  such that both  $s, t$  belong to some bi-component  $C_j$  that contains at least one node in  $A$ .

Formally, let  $I(A) = \{i \in [\ell] : C_i \cap A \neq \emptyset\}$  be the set of the index of bi-components that contains at least one node in the subset  $A$ . We have

$$\mathcal{X}_c^{(A)} = \bigcup_{i \in I(A)} \bigcup_{s' \neq t' \in C_i} P(s', t') \quad (22)$$

Let  $\eta$  be the probability that a shortest path in the ISP sample space  $\mathcal{X}_c$  belongs to the PISP sample space  $\mathcal{X}_c^{(A)}$ , i.e.,

$$\begin{aligned} \eta &= \Pr_{x \sim \mathcal{D}_c} [x \in \mathcal{X}_c^{(A)}] = \frac{\sum_{i \in I(A)} \sum_{s \neq t \in C_i} q_{st}}{\sum_{i=1}^{\ell} \sum_{s \neq t \in C_i} q_{st}} \\ &= \frac{\sum_{i \in I(A)} \sum_{s \in C_i} r_i(s)(n - r_i(s))}{\sum_{i=1}^{\ell} \sum_{s \in C_i} r_i(s)(n - r_i(s))} \end{aligned} \quad (23)$$

We can compute  $\eta$  in  $O(n)$  (similar with computing  $\gamma$ ).

We define The PISP distribution  $\mathcal{D}_c^{(A)}$  over the PISP sample space  $\mathcal{X}_c^{(A)}$  as follows. For any  $i \in I(A), \forall (s, t) \in C_i, \forall p \in P_{st}$ ,

$$\Pr_{x \sim \mathcal{D}_c^{(A)}} [x = p'] = \frac{1}{\eta} \times \Pr_{x \sim \mathcal{D}_c} [x = p'] = \frac{1}{\sigma_{st} \gamma \eta} q_{st}, \quad (24)$$

**Lemma 15.** For any node  $v \in A$ , we have,

$$bc(v) = \gamma \eta \mathbb{E}_{p \sim \mathcal{D}_c^{(A)}} g(v, p) + bc_a(v) \quad (25)$$

*B. Sample space partitioning for RSP<sub>bc</sub>*

Now, we show how to model the betweenness centrality ranking problem as a hypothesis ranking problem and how to apply the SaPHyRa framework.

Here, we consider a binary label space

$$\mathcal{Y}_c = \{0, 1\}, \quad (26)$$

the labeling function  $f_c$  always returns 0, i.e.,

$$f_c(x) = 0, \forall x \in \mathcal{X}_c^{(A)}, \quad (27)$$

and the hypothesis class

$$\mathcal{H}_c^{(A)} = \{h_v \stackrel{\text{def}}{=} g(v, \cdot)\}_{v \in A} \quad (28)$$

where  $g$  is given in Eq. 6, i.e.,  $h_v(p) = 1$  if  $v$  is an inner node of  $p$ .

As  $f_c = 0, \forall x \in \mathcal{X}_c^{(A)}$ , we have,  $L(h_v(x), f_c(x)) = h_v(x)$ . We denote  $\mathcal{R}_c^{(A)}(h_v)$  as the expected risk of the hypothesis  $h_v$ , i.e.,

$$\mathcal{R}_c^{(A)}(h_v) = \sum_{p \in \mathcal{X}_c^{(A)}} \Pr_{x \sim \mathcal{D}_c^{(A)}} [x = p] L(h_v(p), f_c(p))$$

**Lemma 16.** For any node  $v \in A$ , we have,

$$bc(v) = \gamma \eta \mathcal{R}_c^{(A)}(h_v) + bc_a(v)$$

Following SaPHyRa framework, we divide  $\mathcal{X}_c^{(A)}$  into exact subspace and approximate subspace.

**Exact subspace.** We choose the exact subspace is the set of all shortest paths  $p$  that have the length equals 2 ( $len(p) = 2$ ) and there exists a node  $v \in A$  such that  $v$  is an inner node of  $p$  ( $g(v, p) = 1$ ).  
 $\hat{\mathcal{X}}_c^{(A)} = \{p \in \mathcal{X}_c^{(A)} \mid len(p) = 2 \text{ and } \exists v \in A \text{ s.t. } g(v, p) = 1\}$  (29)

For a node  $v \in A$ , the expected risk of  $h_v$  on the exact subspace is computed as follows

$$\hat{\ell}_v = \sum_{p \in \hat{\mathcal{X}}_c^{(A)}} \Pr_{x \sim \mathcal{D}_c^{(A)}} [x = p] L(h_v(p), f_c(p))$$

Here, we present the Exact<sub>bc</sub> algorithm to efficiently compute the expected risks on the exact subspace. Let  $B$  be the set of all neighbors of nodes in  $A$ . For each bi-component  $C_i$ , for each source node  $s \in B \cap C_i$ , we execute two phases as follows. In the first phase, we find all the shortest paths of length 2 from  $s$  to  $t \in B \cap A$ . Let  $\Delta_s$  be the set of nodes  $t$  such that the distance from  $s$  to  $t$  is 2 (i.e.,  $d_{st} = 2$ ). For a node  $t \in \Delta_s$ , we denote  $w_t$  as the number of shortest paths from  $s$  to  $t$ . Initially, all we set  $w_t = 0$ , for all  $t \in B$ . To find the value of  $w_t$ , we iterate through all neighbors  $v$  of  $s$ , then iterate through all neighbors  $t$  of  $v$ . If  $t$  is not a neighbor of  $s$ , i.e.,  $d_{st} = 2$ , we add  $t$  to  $\Delta_s$  and increase the value of  $w_t$  by 1. In the second phase, we calculate the two-hop expected risks on the exact subspace of all nodes  $v \in A$  based on the number

of shortest paths that we found in the first phase. Due to the space limitation, we present the pseudocode of SaPHYRa<sub>bc</sub> algorithm in the Appendix of the full version [41].

**Lemma 17.** Let  $\{\hat{\ell}_v\}_{v \in A}$  and  $\hat{Q}/Q$  be the output of Algorithm Exact<sub>bc</sub>. For all node  $v \in A$ , we have

$$\hat{\ell}_v = \sum_{x \in \hat{\mathcal{X}}_c^{(A)}} \Pr_{x \sim \mathcal{D}_c^{(A)}} [x = p] L(h_v(p), f_c(p))$$

$$\Pr_{x \sim \mathcal{D}_c^{(A)}} [x \in \hat{\mathcal{X}}_c^{(A)}] = \hat{\lambda}$$

**Lemma 18.** Algorithm Exact<sub>bc</sub> has the time complexity of  $O(K)$ , where

$$K = \sum_{v \in B} \deg(v)^2.$$

Here,  $\deg(v) = |\text{Adj}(v)|$  is the degree of  $v$ .

Note that, the expected risks on the exact subspace provide “non-empty” estimations for the expected risks.

**Lemma 19.** For all node  $v \in V$ , we have

$$\text{If } \mathcal{R}_c^{(A)}(h_v) > 0, \text{ then } \hat{\ell}_v > 0.$$

**Approximate subspace.** The approximate subspace is the set of the remaining shortest paths after removing the shortest path in the exact subspace, i.e.,

$$\tilde{\mathcal{X}}_c^{(A)} = \mathcal{X}_c^{(A)} \setminus \hat{\mathcal{X}}_c^{(A)} \quad (30)$$

We define the distribution  $\tilde{\mathcal{D}}_c^{(A)}$  over the approximate subspace  $\tilde{\mathcal{X}}_c^{(A)}$ , where the probability to select a path  $p'$  from  $s'$  to  $t'$  is

$$\Pr_{x \sim \tilde{\mathcal{D}}_c^{(A)}} [x = p'] = \frac{1}{1 - \hat{\lambda}} \Pr_{x \sim \mathcal{D}_c^{(A)}} [x = p'] = \frac{1}{1 - \hat{\lambda}} \frac{1}{\gamma} \frac{1}{\sigma_{s't'}} q_{s't'}$$

$$(31)$$

The expected risk on the approximate subspace  $\tilde{\mathcal{X}}_c^{(A)}$  is computed as follows

$$\tilde{\mathcal{R}}_c^{(A)}(h_v) = \sum_{x \in \tilde{\mathcal{X}}_c^{(A)}} \Pr_{x \sim \tilde{\mathcal{D}}_c^{(A)}} [x = p] L(h_v(x), f_c(x)) \quad (32)$$

### C. Risk Estimation in the Approximate Space

We use the same techniques in Subsection III-C to estimate the expected risk of the hypotheses in the approximate space. Here, we present an algorithm to generate samples in the approximate space and show a bound VC dimension, thus, obtaining a stronger bound on sample complexity.

**Generating samples.** We use rejection sampling and multistage sampling techniques to generate samples in the approximate space.

*Rejection sampling.* We apply a rejection sampling method to sample a shortest path  $p$  from  $\tilde{\mathcal{X}}_c^{(A)}$  by repeating sampling a shortest path  $p$  from  $\mathcal{X}_c^{(A)}$  until  $p \notin \hat{\mathcal{X}}_c^{(A)}$ .

*Multistage sampling.* We use a multistage sampling method to reduce the space complexity of  $O(n)$ . Our multistage sampling method consists of 4 steps (please see Algorithm 2). First, we pick a bi-component  $C_i$  with probability  $\Pr_{x \sim \mathcal{D}_c^{(A)}} [x \in C_i]$ . Secondly, we pick a source node  $s \in C_i$  with probability  $\Pr[s|x \in C_i]$ . Thirdly, we pick a target node  $t \in C_i \setminus \{s\}$  with probability  $\Pr[t|s]$ . Finally, we pick a shortest path  $p$  from  $s$  to  $t$  with probability  $\Pr[p|st]$ . By using the multistage sampling method as above, the probability that we pick a shortest path  $p$  from  $s$  to  $t$  in a bi-component  $C_i$  is

---

### Algorithm 2: Algorithm Gen<sub>bc</sub>

---

**Input :** A graph  $G$ , its set of bi-components  $\mathbf{C}$ , and a subset  $A$

**Output :** A sample  $p$  as a random shortest path in the estimation subspace  $\tilde{\mathcal{X}}_c^{(A)}$  with the probability proportional to the probability distribution  $p_c^{(A)}$

1 **repeat**

2     Pick a  $i \in I(A)$  randomly with probability

$$\Pr_{x \sim \mathcal{D}_c^{(A)}} [x \in C_i] = \frac{n^2 - \sum_{s \in C_i} r_i(s)^2}{\gamma \eta}$$

3     Pick a source node  $s \in C_i$  randomly with probability  $\Pr[s|x \in C_i] = \frac{r_i(s)(n-r_i(s))}{n^2 - \sum_{s \in C_i} r_i(s)^2}$

4     Pick a target node  $t \in C_i \setminus \{s\}$  randomly with probability  $\Pr[t|s] = \frac{r_i(t)}{n-r_i(s)}$

5     Uniformly pick a shortest path  $p$  from  $s$  to  $t$ , i.e.,  $\Pr[p|st] = \frac{1}{\sigma_{st}}$ ;

6 **until**  $p \notin \hat{\mathcal{X}}_c^{(A)}$ ;

7 **Return**  $p$

---

$$\Pr_{x \sim \mathcal{D}_c^{(A)}} [x \in C_i] \times \Pr[s|x \in C_i] \times \Pr[t|s] \times \Pr[p|st] = \frac{1}{\sigma_{st}} \frac{q_{st}}{\gamma \eta}$$

To uniformly sample a shortest path  $p$  from  $s$  to  $t$ , we perform a balanced bidirectional BFS (breadth-first search) [12] to find all shortest paths from  $s$ . We execute two BFSs from both the source node  $s$  and the target node  $t$ , in such a way that the two BFSs are likely to explore about the same number of edges. When the two BFSs “touch each other”, we can obtain the distance and all the shortest paths from  $s$  to  $t$ .

**Lemma 20.** The probability that algorithm Gen<sub>bc</sub> returns a shortest path  $p'$  from  $s'$  to  $t'$  with probability

$$\Pr_{x \sim \tilde{\mathcal{D}}_c^{(A)}} [x = p'] = \frac{1}{1 - \hat{\lambda}} \frac{1}{\gamma} \frac{1}{\sigma_{s't'}} q_{s't'}$$

Borassi et al. [12] analyze the time complexity of balanced bidirectional BFS in a random graph as follows.

**Lemma 21** (Theorem 4 [12]). Let  $G$  be a graph generated through the aforementioned models[12]. For each pair of nodes  $s, t$ , w.h.p., the time needed to compute an  $st$ -shortest path through a bidirectional BFS is  $(n^{\frac{1}{2}+o(1)})$  if the degree distribution has finite second moment.

**Personalized VC dimension and Sample Complexity.** Here, we show the analysis for the VC dimension on the personalized ISP sample space. By using the bi-component-based sampling method, we can reduce the VC-dimension from log of the diameter of the graph [45] to log of maximum diameter of a bi-component in the graph. Further, for a specific subset of nodes, we can further reduce the VC-dimension based on the properties of the subset.

For a shortest path  $p \in \mathcal{X}_c^{(A)}$ , let  $\pi(p)$  be the number of hypotheses  $h_v \in \mathcal{H}_c^{(A)}$  such that  $h_v(p) = 1$ . From Eq. 6,  $h_v(p) = 1$  iff  $v$  is an inner node in  $p$ . Thus,  $\pi(p)$  is the number of nodes in  $A$  that are inner nodes of  $p$ . Recall that, in Lemma 5, we have shown that  $VC(\mathcal{H}_c^{(A)}) \leq \lfloor \log(\pi_{\max}) \rfloor + 1$ ,

Subset	Full network	Any subset $A$	$l$ -hop neighbors
Riondato et al.[45]	$\lfloor \log(VD(V) - 1) \rfloor + 1$	$\lfloor \log(VD(V) - 1) \rfloor + 1$	$\lfloor \log(VD(V) - 1) \rfloor + 1$
SaPHYRa <sub>bc</sub>	$\lfloor \log(BD(V) - 1) \rfloor + 1$	$\lfloor \log(BS(A)) \rfloor + 1$	$\lfloor \log(2l + 1) \rfloor + 1$

TABLE I: The comparison on the bound of the VC-dimension. Here  $VD(V)$  is the diameter of the graph,  $BD(V)$  is the maximum diameter of a bicomponent of the graph, and  $BS(A)$  is the maximum number of nodes  $A$  that appear in the same shortest path.

where  $\pi_{\max} = \max_{x \in \mathcal{X}} \pi(x)$ . Thus, we have the following corollary.

**Corollary 22.** *Let  $BS(A)$  be the maximum number of nodes in  $A$  that are inner nodes of a shortest path in  $\tilde{\mathcal{X}}_c^{(A)}$ . Let  $\mathcal{H}_c^{(A)}$  is defined as in Eq. 28 We have,*

$$VC(\mathcal{H}_c^{(A)}) \leq \lfloor \log(BS(A)) \rfloor + 1 \quad (33)$$

Note that, it is expensive to compute the exact value of  $BS(A)$ . Thus, we bound the value of  $BS(A)$  as follows.

**Lemma 23.** *For a subset of nodes  $A' \subseteq V$ , let  $VD(A') = \max_{s, t \in A'} d_{st}$  be the diameter of  $A'$ . We have,*

$$BS(A) \leq \max_{i=1}^{\ell} (\min(VD(C_i) - 1, VD(A \cap C_i), +1, |A \cap C_i|)) \quad (34)$$

We can simplify the bound for  $BS(A)$  based on the maximum diameter of a bi-component

$$BD(V) = \max_{i=1}^{\ell} VD(C_i) \quad (35)$$

and the maximum distance between any two nodes in the same bi-component in  $A$

$$SD(A) = \max_{i=1}^{\ell} VD(A \cap C_i). \quad (36)$$

Indeed, from Eq. 34, we have,

$$\begin{aligned} BS(A) &\leq \min(\max_{i=1}^{\ell} VD(C_i) - 1, \max_{i=1}^{\ell} VD(A \cap C_i) + 1) \\ &= \min(BD(V) - 1, SD(A) + 1) \end{aligned}$$

We bound the diameter of a subset of nodes  $A'$  as follows. We pick a random source nodes  $s$  and perform a breadth-first search [46] to find the distance from  $s$  to all other node in  $A'$ . The diameter of  $A'$  cannot be bigger than double of the maximum distance from  $s$  to a node  $t \in A'$ , i.e.,

$$\forall s \in A', VD(A') \leq 2 \max_{t \in A'} d_{st}$$

*Comparison on the bound of the VC-dimension.* In Table I, we compare the VC-dimension of SaPHYRa<sub>bc</sub> and the work in [45]. In Riondato et al.[45], the VC-dimension equals on log of the diameter  $VD(V)$  of the network. In SaPHYRa<sub>bc</sub>, by using the bi-component-based sampling method, on the full network, the VC-dimension reduces to log of the maximum diameter  $BD(V)$  of a bi-component in the network. On a subset  $A$ , the VC-dimension further reduces to log the maximum distance between two nodes in  $A$  that belong to the same bi-component. Specifically, if  $A$  is a subset of  $l$ -hop neighbors of a node  $v$ , the VC-dimension equals log of  $2l + 1$ .

#### D. SaPHYRa<sub>bc</sub> algorithm

We now describe SaPHYRa<sub>bc</sub> algorithm. At the beginning, we decompose graph  $G$  into bi-components  $\{C_1, \dots, C_\ell\}$  and compute that out reach for each node. This can be done in  $O(m + n)$ . We define  $\mathcal{X}_c^{(A)}$ ,  $f_c$ ,  $\mathcal{D}_c^{(A)}$ ,  $\mathcal{H}_c^{(A)}$  as in Eq. 22, Eq.

27, Eq. 24, Eq. 28, respectively. The sample space  $\mathcal{X}_c^{(A)}$  is partitioned into  $\hat{\mathcal{X}}_c^{(A)} \cup \tilde{\mathcal{X}}_c^{(A)}$  where

$$\begin{aligned} \hat{\mathcal{X}}_c^{(A)} &= \{p \in \mathcal{X}_c^{(A)} \mid \text{len}(p) = 2 \text{ and } \exists v \in A \text{ s.t. } g(v, p) = 1\} \\ \tilde{\mathcal{X}}_c^{(A)} &= \mathcal{X}_c^{(A)} \setminus \hat{\mathcal{X}}_c^{(A)} \end{aligned}$$

Then, we compute  $\gamma, \eta$  as in Eq. 19, Eq. 23, respectively. The computation of  $\gamma, \eta$  can be done in  $O(n)$ . Let  $\epsilon^* = \epsilon\gamma\eta$ . We obtain the estimation  $\{\ell_v\}_{v \in A}$  by running SaPHYRa with input  $(\mathcal{X}_c^{(A)}, p_c^{(A)}, f_c, \mathcal{H}_c^{(A)}, \epsilon^*, \delta)$ , a partition  $\mathcal{X}_c^{(A)} = \hat{\mathcal{X}}_c^{(A)} \cup \tilde{\mathcal{X}}_c^{(A)}$ . In SaPHYRa<sub>bc</sub> algorithm, we use algorithm Exact<sub>bc</sub> to compute the compute the expected risks on the exact subspace, and algorithm Gen<sub>bc</sub> to generate a sample.

For each node  $v \in V$ , we compute  $bc_a(v)$  as in Eq.21 as output an estimation for the betweenness centrality

$$\tilde{bc}(v) = bc_a(v) + \gamma\eta\ell_v.$$

Due to space limitations, we present the pseudocode of SaPHYRa<sub>bc</sub> algorithm in the Appendix of the full version [41].

#### E. Correctness and Complexity.

**Theorem 24.** *Let  $\{\tilde{bc}(v)\}_{v \in A}$  be the estimation that is returned by SaPHYRa<sub>bc</sub> algorithm. We have*

$$\Pr \left[ \forall v \in A, \tilde{bc}(v) - bc(v) < \epsilon \right] \geq 1 - \delta.$$

**Lemma 25.** *Let  $G$  be a graph generated through the aforementioned models[12]. SaPHYRa<sub>bc</sub> algorithm has a time complexity of  $O(m + n + K + \frac{1}{\epsilon^2} (\lfloor \log(BS(A)) \rfloor + 1 + \ln \frac{1}{\delta}) n^{1/2+o(1)})$ ,*

Note that, due to the space limitation, here, we omit the proofs of the lemmas. The detailed proofs are presented in the Appendix of the full version [41].

## V. EXPERIMENTS

### A. Experiments settings

**Algorithms.** We compare SaPHYRa<sub>bc</sub> algorithm, that is described in Subsection IV-D, with ABRA [47] (that uses node-pair sampling) and KADABRA [12] (that uses path sampling with bi-directed BFS). Note that, both ABRA and KADABRA can only estimate the betweenness centrality for the whole network. We also show the experiment result on SaPHYRa<sub>bc</sub>-full, i.e., the SaPHYRa<sub>bc</sub> algorithm with the subset of nodes is the whole network.

TABLE II: Networks' summary.

Networks	#Nodes	#Edges	Diam.
Flickr	1.6 M	15.5 M	24
LiveJournal	5.2 M	49.2 M	23
USA-road	23.9 M	58.3 M	1524
Orkut	3.1 M	117.2 M	10

**Networks and subsets.** We use 4 real-world networks from [48, 49] as shown in Table II. We ignore the information on the weight and direction of the edges, treating the networks as undirected and unweighted. Unless otherwise mentioned,

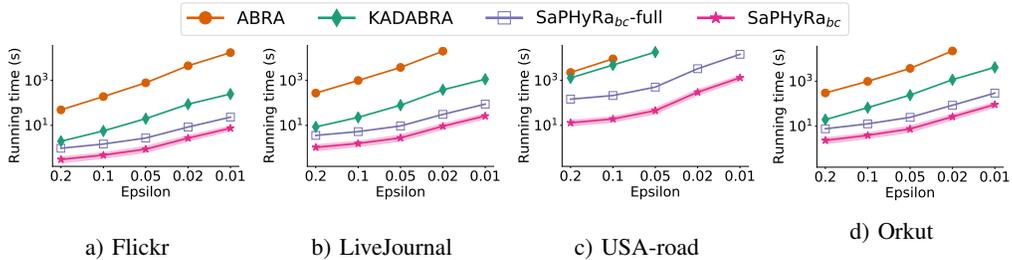


Fig. 3: Running time (log-scale). The shaded areas show the 95% confident intervals of SaPHyRa over different target subsets.

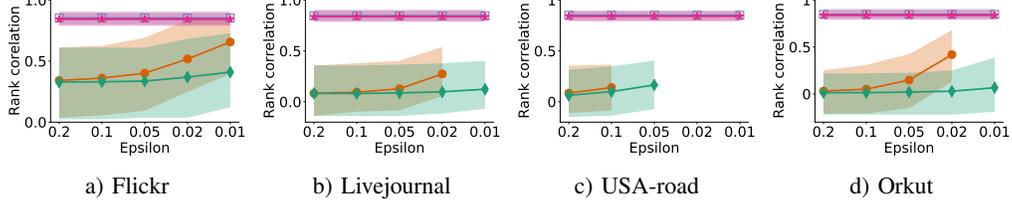


Fig. 4: Rank correlation at different error guarantees  $\epsilon$ . Shading areas show the 95% confident intervals.

in our experiments, we select 1000 different subsets in which each subset consists of 100 random nodes. We set  $\epsilon$  to 0.05 and  $\delta$  to 0.01.

**Ground truth.** We use the ground truth for Flickr, LiveJournal, and Orkut provided in [21]. The ground truth was found in [21] by running a parallel version of the Brandes algorithm on a Cray XC40 supercomputer with 96,000 CPU cores and roughly 400TB of RAM. It took 2 million core hours (or roughly 10 years of calculations) to complete the calculation for 20 networks [21]. We obtain the ground truth for USA-road network using a parallel version of the Brandes’s algorithm on our server with 96 Xeon E7-8894 CPUs (and 6TB memory) in about 2 weeks.

**Metrics.** We compare the performance of the algorithms based on the following metrics.

- *Running time.* Here, we exclude the time to load the network when we measure the running time.
- *Rank quality.* For rank quality, we compute the *Spearman’s rank correlation* (see Eq.1) between the estimation and the ground truth. Note that, when we compute the rank of nodes, if there are two nodes with the same betweenness centrality, we break the tie by the nodes’ IDs.
- *(Signed) relative error.* For a node  $v$ , let  $bc(v)$  be the betweenness centrality of  $v$  and  $\tilde{bc}(v)$  be the estimation, the relative error is given as  $\left(\frac{\tilde{bc}(v)}{bc(v)} - 1\right) \times 100\%$ . In the case where  $bc(v) = 0$ , if  $\tilde{bc}(v) = 0$ , the relative error is 0. Otherwise, the relative error is  $\infty$ .

**Environment.** We implemented our algorithms in C++ and obtained the implementations of others from the corresponding authors. We conducted all experiments on a CentOS machine Intel(R) Xeon(R) CPU E7-8894 v4 2.40GHz. We set the time limit to 10h (36,000s).

## B. Experiment results

First, we run an experiment with varying  $\epsilon \in \{0.2, 0.1, 0.05, 0.02, 0.01\}$  and  $\delta = 0.01$ . We select 1,000 different subset in which each subset consists of 100 random nodes.

**Running time.** From Fig. 3, the running time of SaPHyRa<sub>bc</sub> is 7 – 235 times smaller than KADABRA and 90 – 425 times faster than ABRA. In 10 hours, ABRA can not finish in 5 cases and KADABRA can not finish in 2 cases.

Furthermore, the running time of SaPHyRa<sub>bc</sub> on a set of target nodes is also better than the running time of SaPHyRa<sub>bc</sub>-full. Indeed, on average SaPHyRa<sub>bc</sub> runs 4 – 11 times faster than SaPHyRa<sub>bc</sub>-full.

**Rank correlation.** SaPHyRa<sub>bc</sub> and SaPHyRa<sub>bc</sub>-full always provides a better ranking correlation, in comparison with ABRA and KADABRA (see Fig. 4). For example, in LiveJournal graph, for  $\epsilon = 0.05$ , the Spearman’s rank correlation of the estimation of SaPHyRa<sub>bc</sub> and the ground truth is 0.84. The rank correlation of the estimation of ABRA and KADABRA are 0.13, 0.09, respectively. Furthermore, the rank quality of ABRA and KADABRA are widely varying. For example, for  $\epsilon = 0.05$ , the rank correlation of ABRA varies from 0.12 to 0.63 and the rank correlation of KADABRA varies from 0.02 to 0.58.

We also run an experiment with fixed  $\epsilon = 0.05$  and varying subset size from 10 to 100. As shown in Fig. 5, the varying range of the rank quality of ABRA and KADABRA increases as the subset size decreases.

**Relative error.** We measure the relative error in an experiment where  $\epsilon = 0.05$  and the subset size is 100. In Fig. 6, we show the histogram of the relative error of the estimations with the ground truth. Here, we group all nodes that have relative error bigger than 150% to a single bucket.

From Fig. 6, we observe that for ABRA and KADABRA, more 95% of nodes that have the relative error equal either 0 or  $-100\%$ . A close investigation reveal that those are the nodes with an estimated centrality zero. Those can be further divided into *true zeros*: nodes with betweenness centrality that are correctly estimated as zeros and *false zeros*: nodes with positive betweenness centrality that are incorrectly estimated as zeros.

Combine with the rank quality in Fig. 4, we have the following observations.

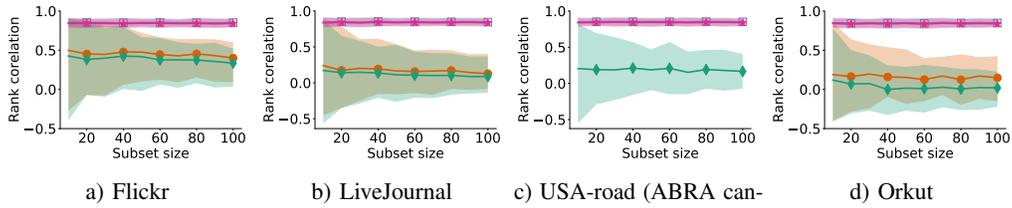


Fig. 5: Rank correlation with different subset sizes. The shaded areas show the 95% confident intervals.

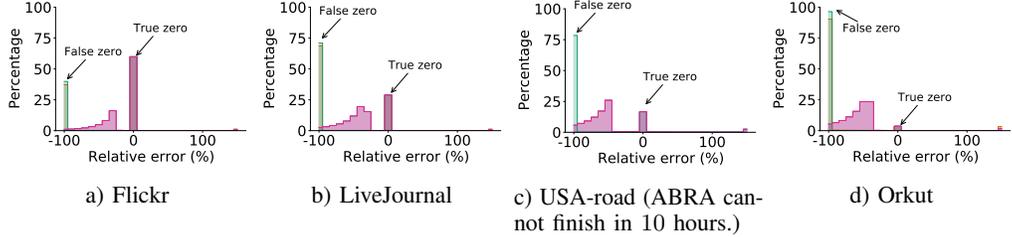


Fig. 6: (Signed) relative error

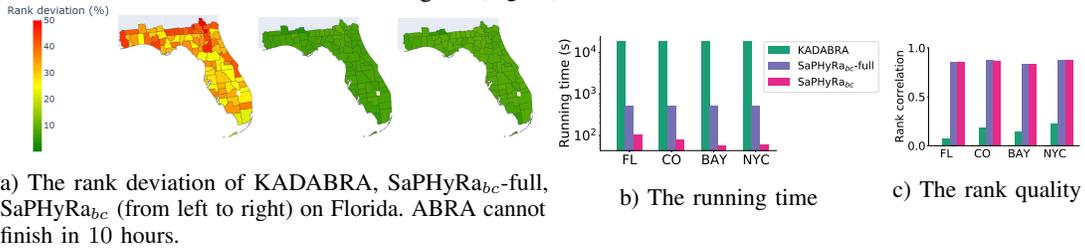


Fig. 7: USA-road

- *The more true zeros, the higher rank quality.* From Fig. 6, the fractions of true zeros are 59, 29, 16, 4% on Flickr, LiveJournal, USA-road, Orkut, respectively. For a node  $v$  with  $bc(v) = 0$ , all the studied algorithms will return 0 as the estimation, i.e., true zeros are the easy cases that cannot be incorrectly estimated. Since the true zero in Flickr network is higher, ABRA and KADABRA provide the estimation with better rank quality on Flickr (see Fig. 5).
- *The fewer false zeros, the higher rank quality.* From Fig. 6, the fractions of false zeros for ABRA are 37, 68, 90% on Flickr, LiveJournal, Orkut, respectively. For KADABRA, the percentages are 39, 71, 78, 96% on Flickr, LiveJournal, USA-road, Orkut, respectively. For SaPHyRa<sub>bc</sub>-full and SaPHyRa<sub>bc</sub>, as we have shown in Lemma 19, there will be no false zeros. As a result, SaPHyRa<sub>bc</sub>-full and SaPHyRa<sub>bc</sub> can provide the estimation with better rank quality than ABRA and KADABRA.

**Case study on USA-road.** Here, we select 4 subsets as 4 areas in [49] (see Table III for summary). More concretely, we extract the longitude, latitude of nodes in 4 areas, and map them with the node in USA-road network.

TABLE III: Subset' summary.

Networks	#Nodes	#Edges
New York City (NYC)	264 K	734 K
San Francisco Bay Area (BAY)	321 K	800 K
Colorado (CO)	435 K	1,057 K
Florida(FL)	1,070 K	2,713 K

Similar to the previous experiments, SaPHyRa<sub>bc</sub>-full and SaPHyRa<sub>bc</sub> outperform KADABRA on both running time

and rank quality (Fig. 7). Furthermore, the running time of SaPHyRa<sub>bc</sub> is better with the subset size smaller size. For example, as the subset size reduces from 1,070K (FL) to 264K (NYC), the running time of SaPHyRa<sub>bc</sub> reduces from 105s to 59.4s.

In Fig. 7a), we show the average rank deviation of nodes in the areas of Colorado. SaPHyRa<sub>bc</sub>-full and SaPHyRa<sub>bc</sub> outperforms KADABRA in term of rank deviation (ABRA cannot finish in 10 hours). For KADABRA, the highest average rank deviation in an area is 39%. Meanwhile, the highest average rank deviation in an area of SaPHyRa<sub>bc</sub>-full and SaPHyRa<sub>bc</sub> are 11%, 12%, respectively.

## VI. CONCLUSION.

We propose and investigate the ranking subset problem when it is computationally prohibitive to obtain the exact centrality values. Our proposed SaPHyRa framework indicates the possibility to reduce the running time significantly when ranking a subset in contrast to ranking all nodes in the network. It also demonstrates an effective way to hybrid good estimation heuristics with sampling-based estimation methods to obtain both high ranking quality and theoretical guarantees on the error. Future directions include extending the framework to other centrality measures such as closeness centrality, nodes' influence, and Shapley value. Further, designing ranking methods with *provable guarantees on the ranking* (not just the estimation errors) is of particular interest.

**Acknowledgment.** The work of My T. Thai is partially supported by NSF under award number CNS-1814614.

## REFERENCES

- [1] M. Newman, "Networks - an introduction," in *Oxford University Press*, 2010.
- [2] K. Okamoto, W. Chen, and X.-Y. Li, "Ranking of closeness centrality for large-scale social networks," in *International workshop on frontiers in algorithmics*. Springer, 2008, pp. 186–195.
- [3] D. Bader, S. Kintali, K. Madduri, and M. Mihai, "Approximating betweenness centrality," in *International Workshop on Algorithms and Models for the Web-Graph*. Springer, 2007, pp. 124–137.
- [4] M. Bianchini, M. Gori, and F. Scarselli, "Inside pagerank," *ACM Transactions on Internet Technology (TOIT)*, vol. 5, no. 1, pp. 92–128, 2005.
- [5] P. Zhao, S. Nackman, and C. Law, "On the application of betweenness centrality in chemical network analysis: Computational diagnostics and model reduction," in *Combustion and Flame*, vol. 162, 2015.
- [6] M. Riondato and E. Upfal, "Abra: Approximating betweenness centrality in static and dynamic graphs with rademacher averages," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 12, no. 5, pp. 1–38, 2018.
- [7] E. Nathan, G. Sanders, J. Fairbanks, D. A. Bader *et al.*, "Graph ranking guarantees for numerical approximations to katz centrality," *Procedia Computer Science*, vol. 108, pp. 68–78, 2017.
- [8] G. Ghoshal and A.-L. Barabási, "Ranking stability and super-stable nodes in complex networks," *Nature communications*, vol. 2, no. 1, pp. 1–7, 2011.
- [9] A. Kirkley, H. Barbosa, M. Barthelemy, and G. Ghoshal, "From the betweenness centrality in street networks to structural invariants in random planar graphs," *Nature communications*, vol. 9, no. 1, pp. 1–12, 2018.
- [10] U. Brandes and C. Pich, "Centrality estimation in large networks," in *International Journal of Bifurcation and Chaos*, vol. 17, no. 7, 2007, pp. 2303–2318.
- [11] U. Brandes, "On variants of shortest-path betweenness centrality and their generic computation," in *Social Networks*, vol. 30, no. 2, 2008, pp. 136–145.
- [12] M. Borassi and E. Natale, "Kadabra is an adaptive algorithm for betweenness via random approximation," in *Proceedings of the 24th European Symposium on Algorithms*, 2016.
- [13] A. Maurer and M. Pontil, "Empirical bernstein bounds and sample-variance penalization," in *COLT*, 2009.
- [14] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
- [15] L. Freeman, "A set of measures of centrality based on betweenness," in *Sociometry*, vol. 40, 1977.
- [16] J. M. Anthonisse, "The rush in a directed graph," in *Stichting Mathematisch Centrum. Mathematische Besliskunde, No. BN 9/71.*, 1971.
- [17] M. Everett and S. Borgatti, "Ego network betweenness," in *Social Networks*, vol. 27, no. 1, 2005, pp. 31–38.
- [18] J. Pfeffer and K. Carley, "k-centralities: local approximations of global measures based on shortest paths," in *Proceedings of the 21st International Conference on World Wide Web*, 2012, pp. 1043–1050.
- [19] S. S. Khopkar, R. Nagi, and G. Tauer, "A penalty box approach for approximation betweenness and closeness centrality algorithms," *Social Network Analysis and Mining*, vol. 6, no. 1, p. 4, 2016.
- [20] M. H. Chehreghani, "An efficient algorithm for approximate betweenness centrality computation," *The Computer Journal*, vol. 57, no. 9, pp. 1371–1382, 2014.
- [21] Z. AlGhamdi, F. Jamour, S. Skiadopoulos, and P. Kalnis, "A benchmark for betweenness centrality approximation algorithms on large graphs," in *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, 2017.
- [22] A. E. Sariyüce, E. Saule, K. Kaya, and Ü. V. Çatalyürek, "Shattering and compressing networks for betweenness centrality," in *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 686–694.
- [23] A. Saxena, R. Gera, and S. Iyengar, "A faster method to estimate closeness centrality ranking," *arXiv preprint arXiv:1706.02083*, 2017.
- [24] A. Saxena, V. Malik, and S. Iyengar, "Estimating the degree centrality ranking," in *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*. IEEE, 2016, pp. 1–2.
- [25] K. Wehmuth and A. Ziviani, "Daccer: Distributed assessment of the closeness centrality ranking in complex networks," *Computer Networks*, vol. 57, no. 13, pp. 2536–2548, 2013.
- [26] F. L. Cabral, C. Osthoff, D. Ramos, and R. Nardes, "Mdaccer: Modified distributed assessment of the closeness centrality ranking in complex networks for massively parallel environments," in *2015 International Symposium on Computer Architecture and High Performance Computing Workshop (SBAC-PADW)*. IEEE, 2015, pp. 43–48.
- [27] Z. C. Steinert-Threlkeld, "Longitudinal network centrality using incomplete data," *Political Analysis*, vol. 25, no. 3, pp. 308–328, 2017.
- [28] M. R. F. de Mendonca, A. M. S. Barreto, and A. Ziviani, "Approximating network centrality measures using node embedding and machine learning," *IEEE Transactions on Network Science and Engineering*, 2020.
- [29] F. Grando, L. Z. Granville, and L. C. Lamb, "Machine learning in network centrality measures: Tutorial and outlook," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–32, 2018.
- [30] F. Grando and L. C. Lamb, "Computing vertex centrality measures in massive real networks with a neural learning model," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.
- [31] A. Kumar, K. G. Mehrotra, and C. K. Mohan, "Neural networks for fast estimation of social network centrality measures," in *Proceedings of the Fifth International*

- Conference on Fuzzy and Neuro Computing (FANCCO-2015)*. Springer, 2015, pp. 175–184.
- [32] P. Avelar, H. Lemos, M. Prates, and L. Lamb, “Multitask learning on graph neural networks: Learning multiple graph centrality measures with a unified network,” in *International Conference on Artificial Neural Networks*. Springer, 2019, pp. 701–715.
- [33] U. Brandes, “A faster algorithm for betweenness centrality,” in *The Journal of Mathematical Sociology*, vol. 25, 2001.
- [34] A. E. Sariyüce, K. Kaya, E. Saule, and Ü. V. Çatalyürek, “Betweenness centrality on gpus and heterogeneous architectures,” in *Proceedings of the 6th Workshop on General Purpose Processor Using Graphics Processing Units*, 2013, pp. 76–85.
- [35] A. McLaughlin and D. A. Bader, “Scalable and high performance betweenness centrality on the gpu,” in *SC’14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2014, pp. 572–583.
- [36] M. Bernaschi, M. Bisson, E. Mastrostefano, and F. Vella, “Multilevel parallelism for the exploration of large-scale graphs,” *IEEE transactions on multi-scale computing systems*, vol. 4, no. 3, pp. 204–216, 2018.
- [37] A. van der Grinten and H. Meyerhenke, “Scaling betweenness approximation to billions of edges by mpi-based adaptive sampling,” in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2020, pp. 527–535.
- [38] T. Alahakoon, R. Tripathi, N. Kourtellis, R. Simha, and A. Iamnitchi, “K-path centrality: A new centrality measure in social networks,” in *Proceedings of the 4th workshop on social network systems*, 2011, pp. 1–6.
- [39] C. Spearman, “The proof and measurement of association between two things,” *The American journal of psychology*, vol. 15, no. 1, pp. 72–101, 1904.
- [40] M. G. Kendall, “Rank correlation methods.” 1948.
- [41] P. Thai, M. Thai, T. Vu, and T. Dinh, “Saphyra: A learning theory approach to ranking nodes in large networks.” [Online]. Available: <https://www.dropbox.com/s/3gnf0fmps4qeagn/icde394.pdf?dl=0>
- [42] E. G. Bajmóczy and I. Bárány, “On a common generalization of borsuk’s and radon’s theorem,” *Acta Mathematica Academiae Scientiarum Hungarica*, vol. 34, no. 3-4, pp. 347–350, 1979.
- [43] J. Hopcroft and R. Tarjan, “Algorithm 447: efficient algorithms for graph manipulation,” *Communications of the ACM*, vol. 16, no. 6, pp. 372–378, 1973.
- [44] F. Harary and D. Welsh, “Matroids versus graphs,” in *The many facets of graph theory*. Springer, 1969, pp. 155–170.
- [45] M. Riondato and E. Kornaropoulos, “Fast approximation of betweenness centrality through sampling,” in *Data Mining and Knowledge Discovery*, vol. 30, no. 2, 2016, pp. 438–475.
- [46] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.
- [47] M. Riondato and E. Upfal, “Abra: Approximating betweenness centrality in static and dynamic graphs with rademacher averages,” in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’16. New York, NY, USA: ACM, 2016, pp. 1145–1154. [Online]. Available: <http://doi.acm.org/10.1145/2939672.2939770>
- [48] “Stanford large network dataset collection.” [Online]. Available: <http://snap.stanford.edu/data/index.html>
- [49] *The shortest path problem: Ninth DIMACS implementation challenge*. [Online]. Available: <http://www.diag.uniroma1.it/challenge9/download.shtml>