Scalable Nearest Neighbors with Guarantees in Large and Composite Networks

Technical Report, September 2010

Petko Bogdanov, Ambuj K. Singh

Deptartment of Computer Science, UC Santa Barbara
Santa Barbara, CA 93106-5110
petko@cs.ucsb.edu
ambuj@cs.ucsb.edu

***Abstract*— We address the problem of k Nearest Neighbor (kNN) search in networks, according to a random walk proximity measure called *Effective Importance*. Our approach retrieves the exact top neighbors *at query time* without relying on off-line indexing or summaries of the entire network. This makes it suitable for very large dynamic networks, as well as for composite network overlays mixed at query time. We provide scalability and flexibility without compromising the quality of results due to theoretical bound guarantees that we develop and incorporate in our search procedure. We incrementally construct a subgraph of the underlying network, sufficient to obtain the exact top k neighbors. We guide the construction of the relevant subgraph in order to achieve fast refinement of the lower and upper proximity bounds, which in turn enables effective pruning of infeasible candidates.**

We apply our kNN search algorithm on social, information and biological networks and demonstrate the effectiveness and scalability of our approach. For networks in the order of a *million nodes*, our method retrieves the exact top 20 using less than 0.2% of the network edges in a fraction of a second on a conventional desktop machine without prior indexing. When employed for nearest neighbors search in composite network overlays, it scales linearly with the number of networks mixed in the overlay.

I. Introduction

The recent growth of online social and information networks gave rise to the emerging field of *Network Science* [5]. It aims at studying and modeling the behavior of agents, interacting within communication, socio-cognitive and information networks treated as a single composite ecosystem of inter-dependent layers. The individual network layers within this system are typically large-scale and dynamic. Pairwise relations among entities and agents in networks arise from different sources and can be based on multiple features. For example, people may have accounts in several online social networks, targeted towards different interaction types. A single individual can be a user of *Facebook* to keep up with her friends, *LinkedIn* to organize and maintain her business contacts, and *Last.fm* to discuss musical trends with fellow listeners. This diverse social ambiance is complemented by an information layer in the form of email communication, generation and discussion of blog entries or shared photographs. Supporting exploratory and analysis tasks in such composite systems has to be flexible and scalable in order to reflect frequent network changes and user-centric prioritization of the different components.

Networks have also become popular in Bioinformatics for modeling interactions of genes within the cell that perform high level cellular processes. Edges in such networks represent interactions between genes. Evidence for interactions among the same set of genes in a genome are often available from multiple data sources and detected by a variety of methods. Gene interaction networks are leveraged for *in silico* drug design [11], functional inference [7] and phenotype prediction [24]. The drug design process, for example, involves studying the effect of regulating a single or a set of genes on the cellular apparatus. An essential computational challenge in drug design is the ability to identify the set of highly impacted genes as a result of inhibiting a target gene of interest. The affected genes map to the closest interaction partners of the target in the network. For a specific drug design task, some interaction data sources may be more relevant than others, and hence the need to find the closest network neighbors according to a query-driven weighting of network layers in the composite environment.

The common ground for networks of different genres and from diverse domains is the need for flexible and scalable analysis methods. We propose a scalable approach for k *Nearest Neighbor (kNN)* search in networks. Given a query node in a network, the problem is to identify its k closest nodes. We generalize kNN to multiple networks over the same set of nodes, called network overlays.

A scalable kNN search provides an important tool for exploration and analysis of the abundance of networked data. It allows for characterization of the neighborhood of a node and enables diverse applications, such as community identification, anomaly node detection, classification, link prediction and collaborative filtering. For example, in the case of community identification, one is interested in the underlying network clusters, formed due to network connections. One can approach this problem by first obtaining the nearest neighbors of nodes in the network and then merging nodes of significantly overlapping neighbor sets. Such application scenarios demand a scalable and flexible kNN search solution suitable for large and dynamic networks.

Existing work on proximity in networks adopts either indexing of the network to facilitate online queries [18], [14], or proximity approximations that work well in practice but lack theoretical accuracy guarantees [33], [6]. Time-expensive indexing approaches are not feasible when the underlying network changes or when search is performed in a composite network, constructed according to query-specific prioritization. Approximations may be undesirable when accuracy is a priority. In addition, previous approaches are not flexible enough to handle prioritization of layers at query time in composite networks. We propose a scalable solution for nearest neighbor search that explores a small subnetwork around the query node, sufficient to retrieve the exact k nearest neighbors.

Instead of relying on off-line indexing, we adopt online pruning of infeasible candidate neighbors based on deterministic bounds on their proximity to the query, computed *on the fly* at search time. The unique advantage of our online pruning, enabled by local bounds computed from scratch, lies in its direct applicability to nearest neighbors search in composite, query-centric mixtures of networks and large dynamic networks undergoing frequent updates. For both of these applications, the answer to the question *"Who are the top neighbors of a given node?"* should reflect a user prioritization of the network layers, and should be provided based on local computation that does not involve the whole network. In line with the above goals, our design of a kNN search method revolves around two key points – *flexibility* in query expression and *scalability*.

Flexibility in layer prioritization is essential, since giving more weight to some tiers in a composite network, as opposed to others, may result in qualitatively different query answers. For example, the closest neighbors of a person for the purpose of music recommendation are different than her closest neighbors in terms of research collaboration opportunities. Our second key goal of scalability of kNN search with the size of the network is dictated by the tremendous rates at which contemporary networks grow. Services like Facebook and Twitter enjoy multi-million user base, generating large amounts of information content. A practical neighbor search method for such scales and rates of changes should obtain the k Nearest Neighbors of a node locally, without computation involving the whole network.

The network proximity measure is the key ingredient for robust nearest neighbors search. The real value of networked data lies in captured transitive connectivity information, encoded in the network structure. A good proximity measure should incorporate a network's structure, beyond the immediate neighbor connections. Network nodes, in the same network community that are connected by multiple good paths are intuitively closer than weakly connected nodes that do not cluster together. Consider, for example, a network of two clusters that are connected by a single link. Although the nodes, adjacent to this link are direct neighbors, they belong to different communities and are expected to be less similar than direct neighbors within the community.

A robust proximity measure employed for social, information and biological networks should also incorporate weights on the edges as opposed to binary links. This is essential for kNN search in networks, as relations have inherently different strengths that affect how close a given neighbor is to the query. For example, interactions in gene networks have levels of certainty and magnitude captured by their method of detection. Similarly, co-authorship links in a scientific paper collaboration network are "stronger" for collaborators with multiple joint manuscripts or when they have collaborated on a manuscript of high impact. A proximity measure and search method should not be agnostic to this quantitative nature of the links in networks.

We define a network proximity measure, called *Effective Importance (EI)*. EI captures community structure in an online fashion and at the same time exhibits advantageous theoretical properties that allow its efficient bounding. In addition, our measure and pruning criteria are specifically tailored to weighted links, which allow us to model strength of connections and determine the top closest neighbors according to it.

Our contributions in this paper are as follows:

1) We introduce the *Effective Importance (EI)* as a probabilistic proximity measure in networks; this measure captures community structure and can be computed efficiently with theoretical guarantees.
2) We propose a scalable algorithm for kNN search in networks that does not rely on off-line indexing.
3) We develop effective pruning criteria, based on locality properties of our proximity measure.
4) We propose the problem of nearest neighbors search in a query-specific composition of networks and design a scalable solution for it.
5) Our experimental evaluation demonstrates the scalability of our approach. It is capable of retrieving the top neighbors 100 times faster than an exhaustive technique while using less than 0.2% of all network edges.

II. Related Work

The problem of nearest neighbors search in networks has been previously addressed in the context of different network types and in terms of different proximity measures. The shortest path measure has been proposed for kNN search on road networks [16], [26]. Although appropriate for navigation, it disregards the multiplicity of sub-optimal paths as well as the degrees of connecting neighbors along the shortest path. Conversely, processes in information, social and biological networks may take place along multiple sometimes suboptimal paths. Another proximity measure that captures the effect of multiple paths is the maximum flow between two nodes [10]. However, it does not penalize longer paths and can be sensitive to small perturbations, since it depends on the capacity of the bottle-neck between the source and destination.

Network proximity has also been modeled by *effective conductance (EC)* in an electrical circuit corresponding to the network graph [13], [20], [29]. As Faloutsos and colleagues point out in [13], *EC* does not account for large degree nodes, connected to multiple small degree nodes. Drawbacks of EC,

are addressed by either introducing a universal sink [13], [29] or by considering a cycle-free version of the effective conductance [20]. Different from such topology augmentations, we propose a measure that handles large degree nodes without the need of additional parameters.

The stationary probability of *random walks with restarts (RWR)* has also been used as a proximity measure [31], [32]. Tong et al. [31] proposed a RWR-motivated approach that can capture both negative and positive personalization drifts. In a follow-up approach [32], the authors improve the running time of the method for bipartite graphs. Both algorithms are based on the pre-computation of a low-rank approximation of the adjacency matrix (or Laplacian) of the graph, originally proposed in [30]. These methods are not applicable to dynamic graphs as the low-rank approximation procedure is too expensive to perform online. In addition, the RWR stationary probability is not well-suited for a proximity measure due to its bias towards high degree nodes. Our measure Effective Importance is related to the RWR probability, but addresses the problem of high degree neighbors and captures community structure.

The use of random walk average commute time as proximity measure was proposed by Sarkar et al. [27], [28]. *GRanch* [27] adopts a truncated version of the commute time, in which walks are restricted to a threshold length, and discovers closest neighbors according to this measure. In a later work, the authors propose a better running time algorithm for truncated commute time, based on sampling [28]. The latter method introduces probabilistic guarantees for the sampling approximation and demonstrates a significant speed-up suitable for online search of the top neighbors. Our kNN algorithm based on Effective Importance shares this query-time local computation property, but it does not involve dynamic programming, and the top-k guarantees we provide are deterministic as opposed to probabilistic. Moreover, our method examines a query node's locality adaptively based on decreasing proximity as opposed to defining a fixed hop-based truncation of the random walks adopted by *GRanch*.

The problem of scalable *RWR* computation has been addressed in the World Wide Web and data mining communities [6], [9], [12], [14], [15], [18], [33]. Haveliwala et al [15] showed that biasing the random walks to a set of restart nodes is equivalent to a linear combination of the proximity to each of the targets. There are three major optimization directions for RWR from a single target: off-line index construction that facilitates on-line proximity queries [18], inexact evaluation [6], [9], [12] or a combination of the two [14].

Indexing approaches are not suitable for composite networks, in which every distinct type of edges is weighted according to a user query, since the index, would need to be precomputed for every mixture of data sources. Although the methods presented in [6], [9], [12] produce experimentally very accurate RWR probability approximation, no theoretical guarantees are provided. A recent method [33] provides an aggregate bound on the $L1$ error of the approximated stationary distribution, but no node-wise guarantees. In addition, the above methods are tailored towards approximating the actual shape of the stationary distribution of RWR, while we aim at computing the top neighbors with guarantees.

Our proximity measure is related to a recent graph partitioning method proposed by Andersen et al [3]. The method approximates a random walk with restarts and performs a sweep on the *degree-normalized* RWR vector to obtain a small-conductance cut. The degree-normalized stationary probabilities were originally used by Lovasz et al to prove a mixing rate result for Markov Chains [22]. This normalized stationary probability renders nodes from the same network cluster closer than nodes from different clusters. We call this quantity *Effective Importance (EI)* and adopt it as a proximity measure. We establish new theoretical properties of EI that enable our fast online and index-free kNN search algorithm.

III. Effective Importance and kNN search

In this section we define and evaluate our proximity measure and establish its theoretical properties. We leverage these properties to construct efficient locally-computed bounds to the proximity of candidate neighbors to a query node. The bounds are then employed in the design of a scalable online kNN algorithm for large dynamic networks and network overlays.

We represent a network as a weighted undirected graph $G(V, E, W)$, where V is the set of nodes, E is the set of edges, and W is a mapping of the edges to real weights $W : (i, j) \rightarrow w_{ij}, i, j \in V, (i, j) \in E$. The volume of a node, denoted w_i is the sum of the weights of its adjacent edges $w_i = \sum_{(i,j)\in E} w_{ij}$. The transition probability of a random walk from node i to node j is defined as $\frac{w_{ij}}{w_i}$. If the Markov chain corresponding to the graph random walk is ergodic then its stationary behavior is governed by a unique distribution of state visit probabilities, called *stationary distribution*. The stationary probability for RWR of node j, if restarting to node r with probability α, is denoted as $\pi_r^\alpha(j)$. This can be expressed according to the balance condition as:

$$\pi_r(i) = \begin{cases} (1-\alpha)\sum_{(i,j)\in N}(w_{ji}/w_j)\pi_r(j) & \text{if } i \neq r \\ \alpha + (1-\alpha)\sum_{(i,j)\in E}(w_{ji}/w_j)\pi_r(j) & \text{if } i = r \end{cases} \quad (1)$$

We will omit the α superscript when the context allows it.

A. Effective importance (EI): a measure to capture community structure

This subsection introduces and evaluates the quality of our proposed proximity measure—*Effective importance (EI)*. We discuss (i) previous theoretical results that guarantee EI's quality and (ii) empirical evaluation on four networks of different genres.

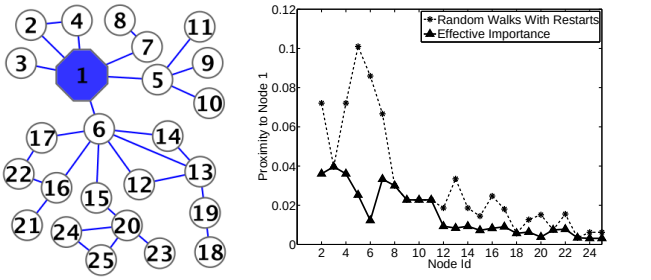
Definition 1: The *Effective Importance (EI)* of a network node is its volume-normalized RWR stationary probability, defined as:

$$q_r^\alpha(i) = \frac{\pi_r^\alpha(i)}{w_i} \quad (2)$$

For zero restart probability ($\alpha = 0$) on undirected graphs, EI is a constant $\frac{1}{\sum_{j\in V} w_j}$ for all nodes [21]. For increasing α, EI is higher for the restart node's neighbors that keep the

walker in the restart node's proximity. Such close neighbors receive *more visits per adjacent edge* than nodes further away or nodes that let the walker "escape" to other parts of the network.

We propose the use of *EI* as a proximity measure of the restart node to all other nodes. Compared to the RWR probability which has also been used as a proximity measure ([31], [32]), EI assigns high proximity to nodes that connect to the restart node and its neighbors, as opposed to nodes of sheer high degree. It captures the effect of network clusters, rendering nodes within the same cluster closer than nodes in different clusters.

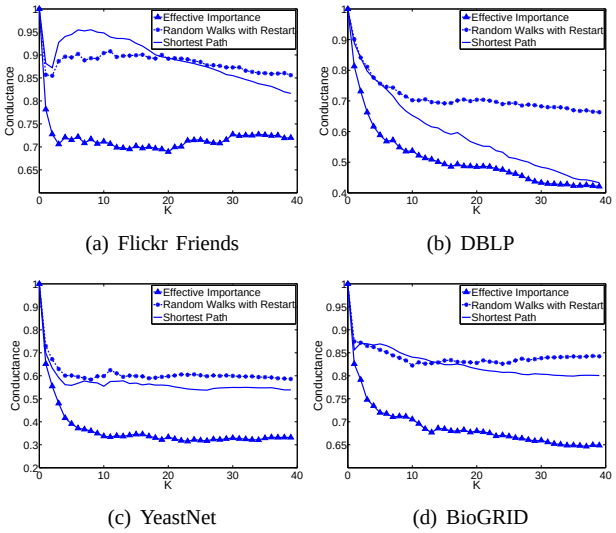


(a) Network example (b) Comparison of the proximity to node 1 based on EI and RWR

Fig. 1. A network example (a) and the proximity to node 1 measured by RWR and Effective Importance (b).

We illustrate the effect of using EI as a proximity measure in a small example graph and further quantify this effect in real-world networks. Figure 1(a) shows a small network for which we compare the proximity of all nodes to node 1 based on RWR and EI (Figure 1(b)). Edges between nodes designate similarity and for the sake of clarity they are unweighted in this example. RWR ranks nodes 5 and 6 as the nearest neighbors of 1, due to their high degree compared to the other neighbors of 1. Intuitively, node 6 should not be among the closest and hence most similar nodes to 1 as it is part of a different cluster, connecting to multiple non-neighbors of 1. In other words, 6 is similar to a number of nodes, which are neither similar to node 1, nor to 1's neighborhood. Conversely, EI renders nodes 3, 2 and 4 closer to 1 as they connect and are similar exclusively to 1 and among themselves. EI measures the number of visits per adjacent edge and thus ranks high nodes that connect well with the query and contains the random walk in its vicinity.

The ability of EI to rank well-clustered neighbors around a query node has been exploited by a family of recent local partitioning methods [4], [3], [2]. All above methods seek to obtain a good-quality local cut around a target node by performing a sweep based on EI. All nodes are ordered by their decreasing EI from the target and a cut at every position is evaluated and the minimum of all such retained. Particularly Andersen and colleagues [3] show that for any set of nodes C of conductance γ, they can produce a cut of conductance $O(\sqrt{\gamma \log \sum_{i \in C} w_i})$ using the EI proximity vector from a node in C. While the above methods use EI as a ranking function, their goal is local partitioning. In this work, we



(a) Flickr Friends (b) DBLP

(c) YeastNet (d) BioGRID

Fig. 2. Conductance of cuts based on Effective Importance, Random walks with restarts and Shortest paths. Closest EI neighbors form better clusters around the query as compared to the closest RWR or SP neighbors for *Flickr[25]* (a), *DBLP co-authors* (b), *YeastNet[24]* (c) and *BioGRID[1]* (d). Lower cut corresponds to better separation of the top k neighbors from the rest of the network.

prove useful local properties of EI that enable its efficient computation and its applicability for kNN proximity search in large dynamic graphs.

Next, we evaluate the utility of using EI as a proximity measure in real world biological and collaboration networks. To quantify the difference of proximity rankings, we evaluate the tightness of the clusters induced by the subgraph containing the closest neighbors. We consider the top k EI, RWR and shortest path neighbors of randomly chosen nodes and measure their separation from the rest of the network in terms of cut conductance [8]. The conductance of a cut is defined as:

$$\Phi_q(Q) = \frac{\sum_{i \in Q, j \in V \setminus Q} w_{ij}}{\min(\sum_{i \in Q} w_i \ , \ \sum_{j \in V \setminus Q} w_j)}, \qquad (3)$$

where Q is the set of top neighbors of a query node q. The conductance measures the fraction of cut edges to all edges on the smaller size of the cut in terms of volume. It is a well-established measure of the optimality of a two-way clustering [8], [19], [3]. The smaller the conductance the better the separation of the nodes on both sides of the cut.

Figure 2 presents a comparative analysis of employing EI, RWR and Shortest paths (SP) as proximity measures in social, co-authorship and gene interaction networks [25], [1], [24] (for details on the datasets refer to the experimental Section IV-A). We measure the conductance around a random target node for sets Q comprised of the closest k neighbors. We report the average conductance for 50 randomly chosen target nodes in each of the networks. EI achieves consistently better conductance than RWR and SP for increasing position of the cut k. RWR's worse performance can be explained by its preference for high degree nodes. The good performance of SP on DBLP (2(b)) is due to the natural clusters corresponding to

academic labs with high degree among the lab members and not many connections to other labs. The SP measure suffers from instability due to multiple tied nodes in unweighted networks (we order such nodes arbitrarily) and its inability to account for multiplicity of paths.

EI can also be interpreted as the odds ratio of the visit probability of a random walk *with restarts* versus a random walk *without restarts*.

$$q_r^\alpha(i) = \pi_r^\alpha(i)/w_i \propto (\pi_r^\alpha(i)/w_i)/\sum_{j\in V} w_j = \pi_r^\alpha(i)/\pi_r^0(i) \tag{4}$$

Odds of visit for nodes closer to the restart node are higher than nodes far from it. Note that the normalization factor of the total network volume in (4) is common for all nodes, so comparing nodes by EI is equivalent to comparing them by *restart/no-restart* odds ratio.

Evidence for the advantages of using EI as proximity measure are both based on theoretical analysis [3] and our empirical evaluation in Fig. 2. The ability of EI to rank high well-clustered neighbors of the query coupled with the theoretical properties we derive in the following sections make it a useful graph measure for large dynamic networks and network overlays.

B. Bounds on the Effective Importance

Performing RWR with $\alpha > 0$ creates a bias in the Effective Importance, causing neighbors of the restart node to receive more visits per adjacent edge. As a result, every node in the network, excluding the restart node r, has at least one neighbor of higher EI.

Lemma 1: **(Unbalanced EI)** Consider a random walk with restart $\alpha > 0$ to node r in a weighted graph. Then,

$$\forall i \neq r,\ \exists j\{(j,i) \in E, q_r(i) \leq (1-\alpha)q_r(j)\}. \tag{5}$$

Proof: Available in the appendix. ■

Lemma 1 is central to deriving a bound to all nodes in the network based on a subgraph around the query as outlined in the following Theorem 1.

For the rest of our results, we assume a partitioning of the nodes in the graph in three sets: (i) a set of known (active) nodes K; (ii) a set of fringe nodes F, directly connected to nodes in K; and (iii) the set of all other nodes U. The restart node r is a member of K and there is no direct edge between K and U. Our kNN algorithm will operate on the set K, using lower and upper bounds to all network nodes, also computed within K. The smaller the active set K, the lower the online search running time.

If the EI values of nodes in F are available, we can established an upper bound on the EI of all nodes in U.

Theorem 1: **(Bound on EI in U)**

$$q_r(u) \leq (1-\alpha)\max_{f\in F} q_r(f),\ \forall u \in U. \tag{6}$$

Proof: Available in the appendix. ■

Note that according to Theorem 1, there are no restrictions on how the set $K \bigcup F$ is chosen as long as it contains the restart node. This theorem provides a powerful mechanism for pruning irrelevant candidates residing in U without the need of considering them as part of the active graph used at query time. If we find a way of obtaining the largest actual EI within the fringe set F, we can use $(1-\alpha)$-fraction of it as an upper bound to all unobserved nodes in U. Obtaining the actual EI in F, however, requires computing the stationary distribution of the whole graph. Instead, we will apply this theorem using an upper bound to the EI in F derived based on analyzing the subgraph induced by $K \bigcup F$.

Now we are ready to introduce our lower and upper bound constructions for the values of EI within the active set K.

Theorem 2: **(Lower bound)** For a modified network graph G_{lb} in which every outgoing edge $(f,x) \in E, f \in F$ is replaced with a self-edge (f,f), such that $w_{ff} = \sum_{(f,x)\in E} w_{f,x} = w_f$,

$$\pi_{lb}^\alpha(i) \leq \pi^\alpha(i),\ \forall i \in K \tag{7}$$

Proof: Available in the appendix. ■

The above lower bound construction transforms all fringe nodes into sinks, as their incoming edges are kept but their outgoing edges are redirected to themselves. The random walker in this augmented network cannot progress to the nodes in U and hence $\pi_{lb}^U = 0$. As a result, we can compute the values of π_{lb}^K in the sub-graph containing only nodes in $K \bigcup F$ by solving for the stationary distribution of this smaller graph. The lower bound values π_{lb}^K approach the actual values π^K as $|K|$ approaches $|V|$. Refining the lower bound estimates is possible at the price of solving for the stationary distribution of larger sub-networks. This "pay-as-you-go" property of the construction allows for flexibility in defining the set K. In our experiments, the lower bound is very tight with respect to the actual value for small sizes of K. The bound for EI is obtained from the stationary distribution bound as follows:

$$q_{lb}(i) = \frac{\pi_{lb}(i)}{w_i},\ \forall i \in K \tag{8}$$

In order to provide guarantees for the top k neighbors of a given node, we also need to bound each node's EI from above. Our upper bound construction is inspired by a push-based algorithm that was originally proposed to obtain an approximation of the RWR importance [6], [18] and later modified for performing efficient local graph partitioning [3]. We transform the approximation algorithm into a upper-bounding procedure. The method in [3] computes an ϵ-approximation of the RWR distribution.

Definition 2: An ϵ-approximate RWR stationary distribution for $\pi_{\vec{r}}^\alpha$ is another distribution $\pi_{\vec{r}-\vec{s}}^\alpha$, computed by restarting according to the restart vector $\vec{r}-\vec{s}$, where $0 \leq \vec{s}(i) \leq \epsilon w_i$.

Note that here we overload the adopted notation for restarting to a single node by changing it to a vector specifying multiple restart nodes. In the notation above, $\vec{r}$ is a vector containing 1 in the position of the restart node r and 0 elsewhere and $\vec{s}$ is a vector containing small non-negative values. The original approach [3] fixes the desired value of ϵ and computes the approximation in a size-unconstrained subgraph of the original graph. We turn the problem around

for our upper bound construction by evaluating the accuracy of the approximation for a fixed subgraph. We answer the question: *"If the approximate vector is computed in a fixed subgraph, what is the maximal node-wise deviation ϵ?"*.

Our upper bound is computed by transferring mass of decaying magnitude along the edges in our active set of nodes K of the network until the mass left to transfer from every node becomes negligibly small. In this process we maintain two vectors: the approximation vector $\vec{p}$ and the push vector $\vec{s}$, both of length $|K|$. We construct an upper bound on the RWR distribution vector based on the vectors $\vec{p}$ and $\vec{s}$.

Algorithm 1 Upper Bound

1: Input: $r,\ \alpha,\ K \bigcup F$
2: Output: π^{α}_{ub}
3: $\alpha_{lazy} = \frac{\alpha}{2-\alpha}$
4: Initialize $\vec{p}(i) = 0, \forall i \in K \bigcup F$,
5: Initialize $\vec{s}(i) = \begin{cases} 1 & \text{if } i = r \\ 0 & \text{if } i \in K \bigcup F / \{r\} \end{cases}$
6: **while** $\vec{p}$ and $\vec{s}$ have not converged **do**
7: Choose any $i \in K$
8: $\vec{p}(i) \mathrel{+}= \alpha_{lazy} s(i)$
9: **for** $\forall j, (i,j) \in E$ **do**
10: $\vec{s}(j) \mathrel{+}= \frac{(1-\alpha_{lazy})}{2} \vec{s}(i) \frac{w_{ij}}{w_i}$
11: **end for**
12: $\vec{s}(i) = \frac{(1-\alpha_{lazy})}{2} \vec{s}(i)$
13: **end while**
14: $\epsilon = \max_{i \in K \bigcup F} (\frac{\vec{s}(i)}{w_i})$
15: $\pi^{\alpha}_{ub}(i) = \vec{p}(i) + \epsilon w_i,\ \forall i \in K \bigcup F$
16: **return** π^{α}_{ub}

We first present our upper bound procedure in Algorithm 1 and later provide justification for its correctness. The input to the algorithm includes the restart node r, the explored subnetwork $K \bigcup F$, and the restart probability α. The push algorithm is performed on a "lazy" version of the adjacency matrix, in which every node has a self edge of weight equal to the volume of the node. As a result, at every node a random walker may (i) move to the restart node with probability α, (ii) stay in the same node with probability $(1-\alpha)/2$ or (iii) follow a random outgoing edge with probability $(1-\alpha)/2$. Andersen et al. [3] show that performing a *"lazy"* RWR with restart probability α_{lazy} is equivalent to computing a *non-lazy* random walk with restart probability $2\alpha_{lazy}/(1+\alpha_{lazy})$. As our goal is to bound the distribution for a given input restart probability, we first determine the corresponding "lazy" restart probability *(line 3)*. Next, the approximation $\vec{p}$ and push $\vec{r}$ vectors are initialized *(lines 4,5)*. Push steps are performed until $\vec{p}$ and $\vec{r}$ converge.

Every push operation increases the approximation value of the current node *(line 8)*, increases the push values of all neighbors *(lines 9-11)* and finally decreases the push value of the current node *(line 12)*. We estimate the deviation of the approximation vector from the actual stationary distribution, based on the maximal ratio of remaining push value per unit mass *(line 14)*. An upper bound to the stationary probabilities is obtained by adding the approximation value and the maximal deviation *(line 15)*. Similar to the lower bound, the upper bound to the EI is obtained by normalizing the importance upper bound computed with the push algorithm by every node's volume:

$$q_{ub}(i) = \frac{\pi_{ub}(i)}{w_i},\ \forall i \in K \bigcup F. \tag{9}$$

Lemma 2: Let $\vec{p}^{(*)}$ and $\vec{s}^{(*)}$ be the resulting vectors after applying a single push operation from Algorithm 1 on the previous $\vec{p}$ and $\vec{s}$. Then

$$\vec{p} = \pi^{\alpha}_{\vec{s}-\vec{r}} \Longrightarrow \vec{p}^{(*)} = \pi^{\alpha}_{\vec{s}-\vec{r}^{(*)}} \tag{10}$$

Proof: The push operation we perform is the same as in [3]. The only difference is that we work with generally weighted graphs, however the same proof applies here as well. ■

Notice that after performing a push operation on all nodes in K, the corresponding ϵ decreases. Formally, the initial pair $\vec{p} = \vec{0}$ is a valid trivial approximation of the stationary distribution, however its corresponding approximation deviation is

$$\epsilon = \max_{i \in K \bigcup F} \frac{\vec{s}(i)}{w_i} = \frac{1}{w_i}, \tag{11}$$

which is too loose to be practically used for bounding the EI. During the push operation, mass is moved from the push vector and added to the approximation vector. As a result, the norm of $\vec{p}$ increases monotonically while the one of $\vec{s}$ decreases monotonically by the same amount. When this transfer of mass becomes close to zero, we terminate the push operations.

Theorem 3: **(Upper bound)** The vector $\pi^{\alpha}_{ub}(i)$, computed by Algorithm 1 provides a node-wise upper bound to $\pi^{\alpha}(i)$.

Proof: Available in the appendix. ■

We present our upper and lower bound constructions for the case of single restart node, for the purpose of simplicity of presentation. Due to the linear combination property of stationary distributions of RWR showed by Haveliwala et al [15], all results carry over to restarting to multiple nodes with different restart probabilities.

The computational complexity for the evaluation of both our bounds is $O(c|E|_K)$, where $|E|_K$ is the number of edges adjacent to nodes in K. The c term depends on the mixing rate of the Markov Chain, corresponding to the adjacency matrix of the nodes in K. In our experiments c is typically a constant in the order of 100. The size and density of the active graph induced on K is the dominant component of the complexity. Therefore, being able to determine the top neighbors using a small active subgraph is crucial for the small online query time.

In this section, we derived upper and lower bounds to the effective importance of nodes within a predefined subnetwork that contains the query node. In addition, in Theorem 1, we showed that if the EI is available for nodes in a fixed subnetwork, we can bound (from above) the EI of all nodes in the rest of the network. We combine these results for pruning infeasible neighbor candidates.

C. Bound-Driven Candidate Pruning

We employ our lower and upper bound constructions for pruning nodes that are not among the k nearest neighbors. We can also use the same node-wise bounds to determine the exact ranking among the top k neighbors if the application demands this.

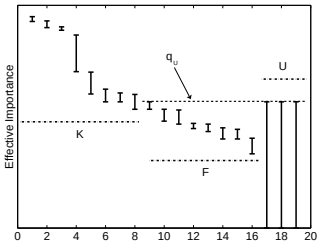


Fig. 3. Pruning of infeasible candidates.

Figure 3 shows a running example of our pruning criterion, for effective top-k searching. The example assumes that the active set K is available. Lower and upper bounds to the EI in K are computed and feasibility intervals are formed for each node. These intervals are shown using vertical error bars in Figure 3 and nodes are sorted by decreasing lower bound. The first 8 nodes comprise the known part of the network K, nodes 9-16 comprise the fringe set F and nodes 17 and on belong to the unknown part of the network. Since the nodes in U are unknown, together with the edges among them, we can only bound them from above according to Theorem 1. Note that in order to apply Theorem 1 we need the actual maximum EI in the fringe set F. Since obtaining the actual values requires computing the exact RWR stationary distribution on the whole graph, we can use their upper bounds instead, obtained according to Equation (9):

$$(1-\alpha)\max_{f\in F} q_{ub}(f) \geq (1-\alpha)\max_{f\in F} q(f) \geq q_u, \forall u \in U. \quad (12)$$

If the query for this example is 4-NN, we can guarantee that the first 4 nodes are the actual top-4 neighbors as all of their lower bounds dominate the upper bounds of the rest of the nodes in $K \bigcup F$ and also dominate the upper bounds of nodes in U. Note that we can give this guarantee, regardless of how many nodes comprise U and without exploring more nodes than the ones in $K \bigcup F$. The actual ordering of the top 4, is not certain in this case as some of their feasibility intervals overlap and their actual order may possibly be different than the one shown.

In the same running example, however, we cannot provide guarantees for the exact 8 nearest neighbors since the feasibility interval of the eight top candidate in K overlaps with those of nodes in F and U. In this case we can expand K by including more nodes with full information about their neighbors. Subsequent expansions of K would refine the bounds estimation and shrink the feasibility intervals making guarantees possible for the nearest neighbors set.

Algorithm 2 Online kNN Search

1: Input: r, α, k and G
2: Output: Ordered set of top-k nodes
3: Initialize $K = \{r\} \bigcup \{j, (i,j) \in E\}$
4: Compute q^{α}_{lb} and q^{α}_{ub}
5: Compute $q^{\alpha}_{ub}(u), u \in U$
6: **while** Top k cannot be guaranteed **do**
7: Extend K with the nodes highest q^{α}_{lb}
8: Refine q^{α}_{ub}, q^{α}_{ub} and $q^{\alpha}_{ub}(u), u \in U$
9: **end while**
10: **return** Top-k nodes

D. Online kNN Search

The refinement of the feasibility intervals, discussed in the previous section, comes at the cost of re-computing the lower and upper bounds for a larger instantiation of K. We would like to obtain the minimal set K that allows us to answer a specific query. An exhaustive search procedure for an optimal K would have to evaluate all subsets of connected nodes that contain the query. For our applications on large and composite networks, any attempt to find an optimal subgraph would add an impractical overhead. We define a greedy procedure that uses the previous bound estimates to direct expansion. We add a fixed number of nodes from the current F to K that have the highest lower bound estimates.

Our online kNN search is outlined in Algorithm 2. The input consists of the query node r, the restart probability, the number of top neighbors k and the network. The set K is initialized with the query node and its immediate neighbors *(line 3)*. Next, we compute the lower and upper bounds *(line 4)* of the EI of nodes in the subgraph $G_{K \bigcup F}$, according to the constructions and algorithm in Section III-B. The upper bound for all unexplored nodes (part of U) is computed based on Theorem 1 *(line 5)*. A series of expansion and refinement steps is performed until the top-k list can be guaranteed using the feasibility intervals of candidate nodes *(lines 6-9)*.

The size of the set K, sufficient for determining the exact kNN, depends on the network structure around the query. Particularly, structures that result in close EI values of the k-th and $(k+1)$-th neighbors, demand a large number of expansions due to persistent overlap between their feasibility intervals. Our algorithm can be easily relaxed to overcome such situations when very fast response is demanded and a small uncertainty of the top-k set is tolerable. In order to adapt our algorithm to *m-tolerant kNN* search, we terminate at *line 6* when less than m candidates are left to prune. The result set contains at most $k+m$ nodes including the actual top-k neighbors. Evaluation of the computational savings from such a relaxation in our experimental section shows that significant online time is spent in pruning the last few candidates.

E. kNN in Composite Networks

Composite network overlays model the connections of a node in multiple networks in which it participates. We consider

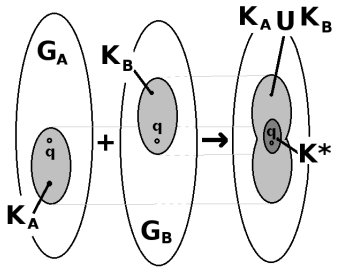


Fig. 4. *Naive* active set compared to an optimal composition-aware counterpart.

kNN search according to user prioritization of the networks in the overlay. We define the composite kNN query as the triple $< r,\ k,\ \vec{\beta}\ >$, where r and k are the query node and the number of desired NN and $\vec{\beta}$ is a vector that specifies the user-defined weight of each layer. We assume there exists a one-to-one mapping between nodes that represent the same entity in different layers.

One naive approach to performing composite kNN search is to (i) apply the online kNN procedure (Algorithm 2) to each layer separately, (ii) discover the corresponding relevant subnetworks K_n in each network layer G_n and then (iii) compose a network, induced by $\{\bigcup_n K_n\}$ with links, weighted according to the mixture vector $\vec{\beta}$. This approach is pictorially presented in Figure 4 for two-network overlay. Although it would scale much better than performing a mix of the whole networks in each layer, it could lead to inefficient selection of the relevant subnetwork in the overlay. We refer to this approach as *Naive*.

The overlap of relevant nodes in each layer may be small for uncorrelated layers. For example, friends might not always have the same taste in music. Moreover, the expansion in each layer should be driven according to its weight in $\vec{\beta}$. High-priority layers should be expanded more aggressively than low-priority ones.

In order to select a small relevant subnetwork in the overlay, we push the mixture vector $\vec{\beta}$ into the expansion step of our kNN search (*step 7* of Algorithm 2). We expand with priority-aware best expansion candidates, taking into account feasibility intervals computed according to all layers at the previous expansion iteration. As a result, we use a smaller active set K^* (in Figure 4) as opposed to a union of separate layer active sets produced by *Naive*.

IV. Experimental evaluation

This section is dedicated to evaluating the scalability and performance of our kNN search algorithm based on Effective Importance. We describe the datasets that we use for experimentation and next we measure the savings due to our bound-based pruning.

A. Data

We experiment with four real world datasets: two from the social information networks domain and two from Biology.

The DBLP co-author network consists of collaboration links between scientific authors based on joint papers ($|V| \approx 700k, |E| \approx 4.5m$). This network is created from the public DBLP[1] dataset by adding a link between two authors if they have joint publications. Link weights are based on the count of joint papers for the adjacent authors.

Another large scale network we evaluate contains a *three-million-user* sample of the Flickr social graph. We also infer a second Flickr network layer based on common photo favorite bookmarks of users. We use data provided by the authors of [25]. The dataset contains a list of $24,885,921$ friendship links connecting $14,648,975$ anonymized users. We work with the largest connected component of *3 million* users and $14,648,975$ connections termed *FRIENDS* in the experiments.

The Flickr dataset also contains information about $11,267,320$ photos and favorite bookmarked photos by users. Using this data we construct a second network based on shared bookmarks by users. To score the similarity of user tastes we use the *Dice* set similarity coefficient.

$$s(u_1, u_2) = \frac{2|B_{u_1} \cap B_{u_2}|}{|B_{u_1}| + |B_{u_2}|}, \tag{13}$$

where u_1 and u_2 are two users and B_{u_1} and B_{u_2} are their corresponding sets of bookmarked photos. We further threshold the similarities, keeping only values greater than 0.01. For overlay experiments on Flickr we sample overlapping users that are both in the largest connected component in the friendship network and in the similarity network termed *FAV* in the experiments.

Other experimental networks we use for evaluation come from genomic research. We experiment with a functional yeast interaction network BioGRID[2]. Nodes represent gene products and links represent interaction between them weighted by the interaction strength. The network contains $35,630$ edges and 4913 genes.

Another biological network is *YeastNet*, a functional gene network overlay of 10 data sources available due to McGary et al. [24]. Each layer corresponds to interactions detected using a different experimental methodology. The overlay contains 5400 genes and more than $250,000$ interactions in all 10 layers.

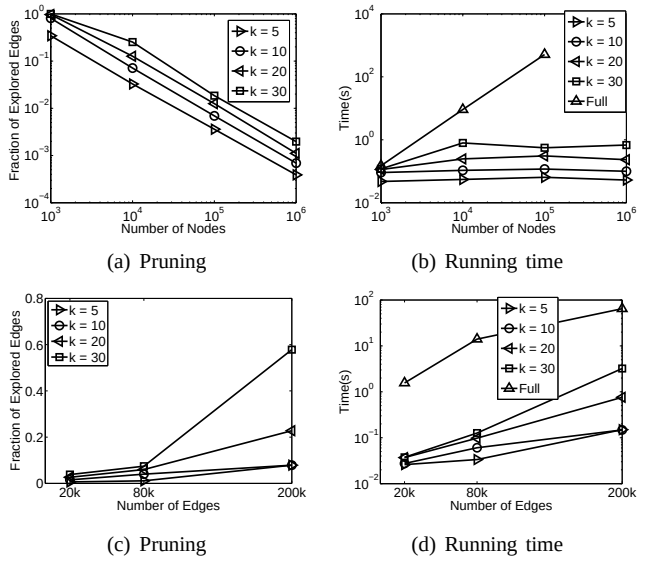
All synthetic networks are constructed according to Barabasi's preferential attachment generative model [5].

B. Scalability

We study the performance of our kNN in terms of running time and pruning power. All experiments are performed on a single machine with $2GB$ of main memory and $3GHz$ dual-core processor. Our proposed algorithms are implemented in C++.

Our natural control comparison when reporting query time is the evaluation of kNN on the full networks (traces marked

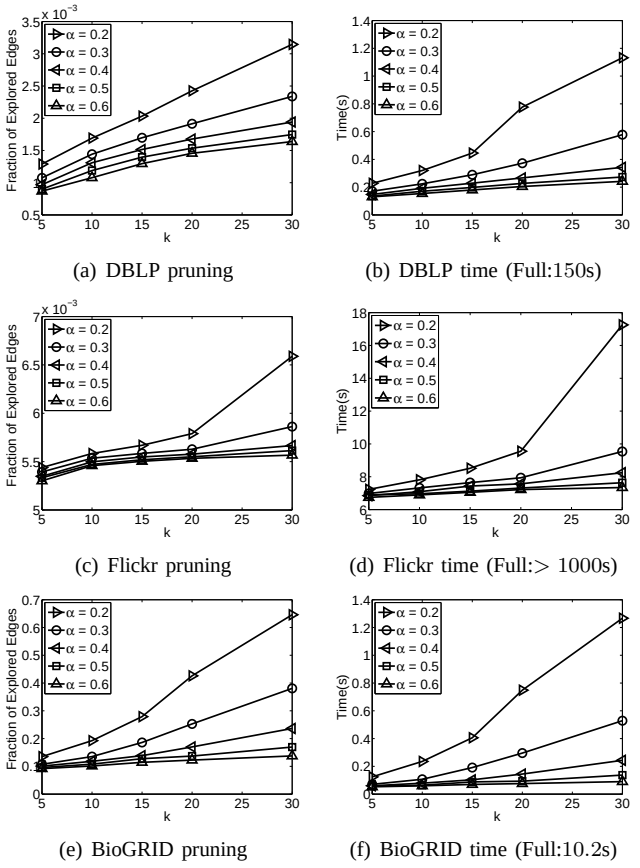
[1] http://dblp.uni-trier.de/xml/dblp.xml
[2] www.biogrid.com



(a) Pruning (b) Running time

(c) Pruning (d) Running time

Fig. 5. Average performance for increasing number of nodes and fixed average degree of 6 (a), (b); and for increasing number of edges and fixed number of $10k$ nodes (c), (d)($\alpha = 0.3$).



(a) DBLP pruning (b) DBLP time (Full:150s)

(c) Flickr pruning (d) Flickr time (Full:$>$ 1000s)

(e) BioGRID pruning (f) BioGRID time (Full:10.2s)

Fig. 6. Average pruning power (a), (c), (e) and online running time (b), (d), (f) for DBLP, Flickr and BioGRID. Running time for computing the measure on the whole network for Flickr is projected based on DBLP size due to the inability to fit the whole network in the main memory (2GB).

$Full$ in the figures). Another machine-independent performance metric is the fraction of edges kNN explores to evaluate a query, i.e. the pruning power of our lower and upper bounds.

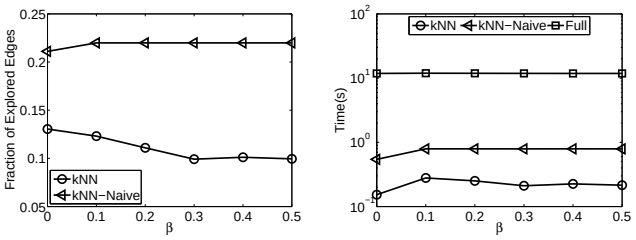
a) Synthetic networks: We use controlled synthetic networks to evaluate the scalability with nodes and edge density. Figures 5(a) and 5(b) present the scalability of kNN for increasing number of nodes, while keeping the average degree in a synthetic network fixed to 6. The expected growth behavior of scale-free graphs is in line with this experiment, since such graphs are typically characterized by a large number of small-degree nodes and a small number of high-degree ones. The size of the active subnetwork K, sufficient to answer the kNN query, remains constant for increasing network sizes. We observe this both in the pruning traces 5(a) that decrease linearly on a *log-log* scale and from the running time which remains constant 5(b). In comparison, the exact stationary distribution (denoted Full 5(b)) for a 100 thousand nodes network takes close to 9 minutes to compute. Our algorithm answers kNN queries in less than a second for k up to 30, which makes it ideal for online analysis.

Next, we evaluate the scalability of our approach for a single synthetic network as it becomes denser (Fig. 5(c), 5(d)). We fix the number of nodes to $10k$ and increase the total number of edges. The average size of the active edge set K is 6% for $80k$ edges, but increases to more than half of all edges when the average degree reaches 20. Note that $10k$ nodes and $200k$ edges corresponds to a dense scenario in which computing the exact EI in the whole network takes $50s$ (trace *Full* in 5(d)).

b) kNN search in real-world networks: Our performance evaluation on single-layer real-world networks is presented in Fig. 6. We achieve sub-second search time in DBLP for values of k up to 30, while using less than 0.2% of the network edges. The actual EI in the whole network takes more than $150s$ to compute. The top neighbor search for the Flickr friendship graph takes 7 seconds on average, while pruning more than 99.4% of the network edges for $\alpha \geq 0.3$. The increased complexity, compared to the DBLP graph is due to the higher density and the bigger size of Flickr. The kNN search expands the known graph to 40% and higher in the BioGRID network due to its smaller size. The actual number of used edges is about $10k$, allowing for sub-second evaluation. For all three networks we tolerate at most 3 additional candidate nodes that are not pruned ($m = 3$), which eliminates corner cases of very close k-th and $(k+1)$-th neighbors.

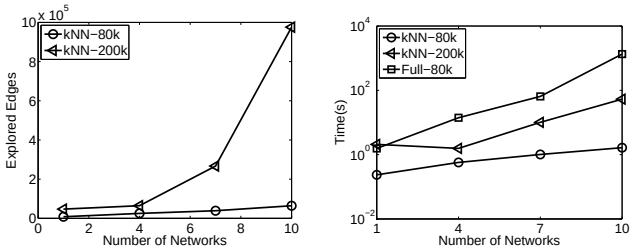
Restart values $\alpha \geq 0.3$ in Fig. 6 allow for practical (close to 1s) search performance. If α is too low, the random walker explores almost the whole network. If α is too high (exceeding 0.6), the walker is restrained to the immediate neighbors of the query and does not capture the deeper community structure. A value of α should reflect the balance between these two extremes.

c) Composite networks: Next, we measure the performance of kNN in composite networks. Fig. 7(a) shows a comparison of the pruning power of a *Naive* composition search (trace *kNN-Naive*) and our tier-optimized kNN for overlay of two synthetic networks ($|V| = 10k, |E| = 80k, k = 20$). We choose a mixture vector $\vec{\beta}$ of norm 1 and increase the



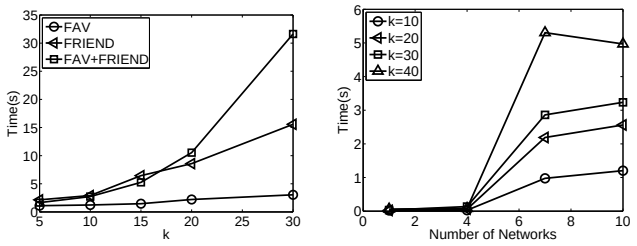
(a) Pruning (b) Running time

Fig. 7. Average performance under different mixing conditions of a two-network overlay ($\alpha = 0.3$).



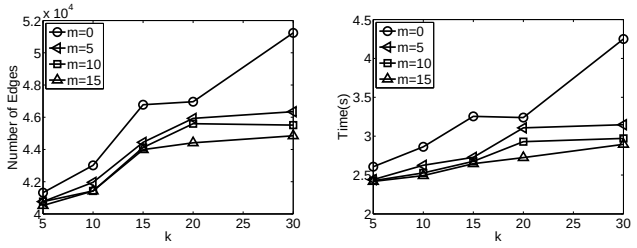
(a) Number of edges (b) Running time

Fig. 8. Performance for increasing number of overlaid networks ($\alpha = 0.3$).



(a) Flickr (Full(1 layer):$> 1000s$) (b) YeastNet (Full(10 layers): $82s$)

Fig. 9. Search time for (a) Flickr friends and favorite photos($\alpha = 0.4,\ m = 5$) and (b) YeastNet ($\alpha = 0.3,\ m = 0$).



(a) Number of edges (b) Running time

Fig. 10. Performance for increasing tolerance m ($\alpha = 0.2$).

weight of one of the networks from 0 to 0.5 while decreasing the weight of the other correspondingly. The kNN algorithm explores two times fewer edges than its*Naive* counterpart and is twice as fast. For the same overlay, computing the exact EI (*Full* in 7(b)) is 100 times slower.

We also measure the performance for increasing number of equally-weighted networks in an overlay (Fig. 8(a) and 8(b)). We generate the networks by computing random permutation of the nodes in single power-law network and reconnect the permuted nodes using the original edges. In this respect, the presented results are pessimistic, as the separate networks have unrelated (orthogonal) edge sets. The number of nodes is fixed to $10k$ and edges of each separate overlay network are $80k$ (trace kNN-$80k$) and $200k$ (trace kNN-$200k$). For $80k$ edge overlays, our online kNN completes in a second, while the respective full computation (trace Full-$80k$) is 1000 times slower on average. When mixing denser networks ($200k$ edges in $10k$ node networks), the search time increases to ten seconds for 7-network overlay. Even when adding 10 dense networks in an overlay, thus forcing the resulting network to have 2 million edges (20% of *all possible* edges), the kNN search takes on average less than 100 seconds.

Search time for real-world composite networks is reported for Flickr (Fig. 9(a)) and YeastNet (Fig. 9(b)). The Flickr composite search time (trace *FAV+FRIEND*) is similar to that of the *FRIEND* layer on its own, for k up to 20, while further neighbors become harder to compute. The similar bookmarks (trace *FAV*) layer is sparser (similarity is thresholded to 1%, details in Appendix) and hence the lower running time. For the gene overlay YeastNet 9(b), we iteratively add layers and compute neighbors for k up to 40. Regardless of the relatively small number of nodes ($5k$), the density of the composite network results in search times up to 5 seconds, while computing the actual EI for the whole network takes more than 80 seconds.

d) Relaxed kNN: We study the effect of relaxing the kNN search to m-tolerant kNN search for values of m up to 15 (Fig. 10(a) and 10(b)). Significant savings both in running time and pruning are observed for small $m = 5$ as compared to exact evaluation. This is due to typically few nodes of very similar EI situated around the k-th position. On average, 30% more online time is required to separate the feasibility intervals of these few nodes, via 15% increased expansion of edges. For large networks and overlays, this small tolerance can enable an order of magnitude performance improvement.

V. Conclusion

We address the growing need for online, index-free search algorithms tailored to dynamic and multi-tier networks across multiple genres. We propose a novel and intuitive proximity measure called Effective Importance that captures the community structure around a query node. Our proposed solution for the pivotal problem of kNN search is scalable and preserves result quality without using precomputed indices. Our experiments on real world and synthetic networks reveal up to 100 times running time improvement of our kNN, compared to exact computation of the proximity measure. Our method provides a practical on-the-fly search solution for real world dynamic networks with accuracy guarantees.

VI. Acknowledgements

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained

in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] Biogrid: General repository for interaction datasets. *http://www.thebiogrid.org/*, 2006.

[2] R. Andersen and F. Chung. Detecting Sharp Drops in PageRank and a Simplified Local Partitioning Algorithm. *Theory and Applications of Models of Computation*, 4484/2007:1–12, 2007.

[3] R. Andersen, F. Chung, and K. Lang. Local Graph Partitioning using PageRank Vectors. *FOCS*, pages 475–486, 2006.

[4] R. Andersen, S. Diego, L. Jolla, F. Chung, K. Lang, and S. Clara. Using PageRank to Locally Partition a Graph. *FOCS*, pages 1–23, 2006.

[5] A.-L. Barabasi. *Linked: the new science of networks*. Perseus Publishing, 2002.

[6] P. Berkhin. Bookmark-coloring algorithm for personalized pagerank computing. *Internet Math*, 3:2006.

[7] P. Bogdanov and A. K. Singh. Molecular Function Prediction Using Neighborhood Features. *TCBB*, 7(December):1–11, 2010.

[8] B. Bollobas. *Modern Graph Theory*. Springer, July 1998.

[9] Y.-Y. Chen, Q. Gan, and T. Suel. Local methods for estimating pagerank values. In *CIKM*, pages 381–389. ACM, 2004.

[10] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. Introduction to Algorithms. *MIT Press*, 1990.

[11] P. Csermely, V. Agoston, and S. Pongor. The efficiency of multi-target drugs: the network approach might help drug design. *Trends Pharmacol. Sci.*, 26:178–182, Apr 2005.

[12] J. V. Davis and I. S. Dhillon. Estimating the global pagerank of web communities. In *SIGKDD*, pages 116–125. ACM, 2006.

[13] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. *KDD*, 2004.

[14] D. Fogaras and B. Racz. Towards scaling fully personalized pagerank. In *WAW*, pages 105–117, 2004.

[15] T. H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *TKDE*, 15:2003, 2003.

[16] H. Hu, D. L. Lee, and J. Xu. Fast Nearest Neighbor Search on Road Networks. In *EDBT*, pages 186–203, 2006.

[17] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *SIGKDD*, pages 538–543. ACM, 2002.

[18] G. Jeh and J. Widom. Scaling personalized web search. *WWW*, 2003.

[19] R. Kannan, S. Vempala, and a. Veta. On clusterings-good, bad and spectral. *Proceedings 41st Annual Symposium on Foundations of Computer Science*, 51(3):367–377, 2004.

[20] Y. Koren, S. C. North, and C. Volinsky. Measuring and extracting proximity in networks. *KDD*, 06, 2006.

[21] L. Lovász. Random Walks on Graphs: A Survey. *Combinatorics, Paul Erdos is Eighty*, 2:1–46, 1993.

[22] L. Lovasz and M. Simonovits. The mixing rate of markov chains, an isoperimetric inequality, and computing the volume. In *SFCS*, 1990.

[23] A. Lubiwt. Some NP-complete Problems Similar To Graph Isomorphism. *SIAM Journal of Computing*, 10(1):11–21, 1981.

[24] K. L. McGary, I. Lee, and E. M. Marcotte. Broad network-based predictability of Saccharomyces cerevisiae gene loss-of-function phenotypes. *Genome biology*, 8(12):R258, 2007.

[25] A. Mislove, H. S. Koppula, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Growth of the flickr social network. In *WOSN*, 2008.

[26] H. Samet, J. Sankaranarayanan, and H. Alborzi. Scalable network distance browsing in spatial databases. In *SIGMOD*, 2008.

[27] P. Sarkar and A. W. Moore. A tractable approach to finding closest truncated-commute-time neighbors in large graphs. In *UAI*, 2007.

[28] P. Sarkar, A. W. Moore, and A. Prakash. Fast incremental proximity search in large graphs. In *ICML*, New York, New York, USA, 2008.

[29] H. Tong, C. Faloutsos, and Y. Koren. Fast direction-aware proximity for graph mining. *SIGKDD*, 2007.

[30] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast Random Walk with Restart and Its Applications. *ICDM*, 2006.

[31] H. Tong, H. Qu, and H. Jamjoom. Measuring Proximity on Graphs with Side Information. *ICDM*, 2008.

[32] H. Tong, H. Qu, H. Jamjoom, and C. Faloutsos. iPoG: Fast Interactive Proximity Querying on Graphs. *CIKM*, pages 1673–1676, 2009.

[33] Y. Wu and L. Raschid. Approxrank: Estimating rank for a subgraph. *ICDE*, pages 54–65, 2009.

APPENDIX

A. Proofs and Additional Results

Proof of Lemma 1

Proof:

$$\begin{aligned} q_r(i) &= \frac{\pi_r(i)}{w_i} \\ &= \frac{1}{w_i}(1-\alpha)\sum_{j\in N(i)}(w_{ji}/w_j)\pi_r(j) \\ &= \frac{1}{w_i}(1-\alpha)\sum_{j\in N(i)} w_{ji}q_r(j) \\ &\leq \frac{1}{w_i}(1-\alpha)\max_j q_r(j)\sum_{j\in N(i)} w_{ji} \\ &= \frac{1}{w_i}(1-\alpha)w_i\max_j q_r(j) \\ &= (1-\alpha)\max_j q_r(j) \end{aligned}$$

The first three equations follow from the definition of EI, and the balance equation. The inequality follows from the algebraic inequality $ab + cd \leq max(b,d)(a+c), a, b, c, d \geq 0$. Finally, we use the definition of the volume of a node. ■

Proof of Theorem 1.

Proof: Choose $\hat{u}$ to be the node in U with largest $q_r(\hat{u})$. Then from Lemma 1 and using the fact that $u \in U \rightarrow u \neq r$ it follows that $\exists \hat{f}\{(\hat{f},\hat{u}) \in E, (1-\alpha)q_r(\hat{f}) >= q_r(\hat{u})\}$. Due to the choice of $\hat{u}$ as the node of maximum EI in U, it follows that $\hat{f} \notin U$. The node $\hat{f}$ is also not in K since there are no edges between U and K. The only possibility is that $\hat{f} \in F$ as the three sets are mutually exclusive. Then we get the following chain of inequalities:
$q_r(u) \leq q_r(\hat{u}) \leq (1-\alpha)q_r(\hat{f}) \leq (1-\alpha)\max_{f\in F} q_r(f)$, for every $u \in U$. ■

Lower and Upper Bound Proofs

Before we prove our lower bound correctness, we will introduce additional notation.

The stationary probability of RWR can be also expressed as an infinite sum of tour probabilities. A tour t in the network is a sequence of traversed nodes $t : v_1 \rightarrow v_2 \rightarrow \cdots v_n$, denoted also as $t : v_1 \rightarrow v_n$. Each tour is associated with a length $l(t) = n-1$ and probability of traversal

$$P(t) = \prod_{i=1}^{n-1} \frac{w_{v_i v_{i+1}}}{w_{v_i}}. \tag{14}$$

Jeh et al. [17], [18] introduce the *inverse p-distance* and show that it is equivalent to the stationary probability of a RWR.

Theorem 4: **(Equivalence of inverse p-distance and π)**

$$d_r(j) = \sum_{t:r\rightarrow j} P(t)\alpha(1-\alpha)^{l(t)} = \pi_r(j), \tag{15}$$

where $d_r(j)$ is the *inverse-p distance*, r and j are nodes in V, t is a tour starting from node r and ending at node j allowing cycles; and the sum is over all possible distinct tours t.

Proof: Available in Jeh et al. [18] for unweighted graphs. The proof trivially extends to weighted graphs. ■

Proof of Theorem 2

Proof: Consider a node $k \in K$. Let us denote the total probability of all paths in G from r to k that stay in K and do not include a node in $F \bigcup U$ as $T^K = \sum_{t_K : r \to k} P(t_K)\alpha(1-\alpha)^{l(t_K)}$. Similarly let us denote the total probability of paths from r to k that include at least one node in $F \bigcup U$ as T^{FU}. As the above two sets of paths are mutually exclusive and span the whole space of paths from r to k we have:

$$T^K + T^{FU} = \sum_{t:r \to k} P(t)\alpha(1-\alpha)^{l(t)} \tag{16}$$

Let us denote the total probability of paths from r to k in the perturbed network G_{lb} as T^K_{lb}. Note, that the superscript K is added just to clarify the fact that all possible paths in the perturbed network do not include a vertex outside of K, due to the nature of the modification. There a is one-to-one correspondence from all paths contributing to T^K in G to all paths in G_{lb} that contribute to T^K_{lb}. The latter is true as we have not removed any edges within K, so every path within K in G exists and has the same probability in G_{lb}. We obtain the following equality:

$$T^K = T^K_{lb}. \tag{17}$$

Using the introduced notation, (16) and (17), we have:

$$\begin{aligned}
\pi(k) &= \sum_{t:r \to k} P(t)\alpha(1-\alpha)^{l(t)} \\
&= T^K + T^{FU},\ due\ to\ (16) \\
&\geq T^K \\
&= T^K_{lb},\ due\ to\ (17) \\
&= \pi_{lb}(k).
\end{aligned}$$

■

Proof of Theorem 3

Proof: From Lemma 2 and the fact that the initial assignment of $\vec{p}$ is a $(\frac{1}{w_r})$-approximation of $\pi^\alpha_{\vec{r}}$ it follows that after all iterations of push operations (Step 4) in Algorithm 1 the obtained $\vec{p} = \pi^\alpha_{\vec{r}-\vec{s}}$ is also an ϵ-approximation of the $\pi^\alpha_{\vec{r}}$. According to Definition 2 the corresponding ϵ should dominate all $\frac{\vec{s}(i)}{w_i}, i \in K \bigcup F$, hence (Step 5) computes the corresponding ϵ.

The authors of [3] show that if $\pi^\alpha_{\vec{r}-\vec{s}}$ is an ϵ-approximation of π^α, then

$$\sum_{i \in S} \pi^\alpha_{\vec{r}-\vec{s}}(i) \geq \sum_{i \in S} \pi^\alpha_{\vec{r}}(i) - \epsilon \sum_{i \in S} w_i, \tag{18}$$

where S is any subset of nodes in the network. If we chose the set S as a singleton node in $K \bigcup F$ and by subtracting $\epsilon \sum_{i \in S} w_i$ on both sides, we get:

$$\pi^\alpha_{\vec{r}-\vec{s}}(i) + \epsilon w_i \geq \pi^\alpha_{\vec{r}}(i), \forall i \in K \bigcup F. \tag{19}$$

Since after all push operations $\vec{p}(i) = \pi^\alpha_{\vec{r}-\vec{s}}(i)$ then the vector $\vec{p}(i) + \epsilon w_i$ provides a node-wise upper bound to the elements in π^α. ■

Expressing the RWR probability as an infinite sum of path probabilities allows us to establish another desirable property of EI as a proximity measure.

Lemma 3: (Symmetric EI) The effective importance is symmetric $q_i(j) = q_j(i)$.

Proof of Lemma 3. *Proof:*

$$\begin{aligned}
q_i(j) &= \frac{\pi_i(j)}{w_j} \\
&= \frac{d_i(j)}{w_j} \\
&= \frac{1}{w_j} \sum_{t:i \to j} P(t)\alpha(1-\alpha)^{l(t)} \\
&= \frac{1}{w_j}\frac{w_j}{w_i} \sum_{t_{rev}:j \to i} P(t_{rev})\alpha(1-\alpha)^{l(t_{rev})} \\
&= \frac{r_j(i)}{w_i} \\
&= \frac{\pi_j(i)}{w_i} \\
&= q_j(i).
\end{aligned}$$

The first and last three equalities follow from the introduced definitions and Theorem 4. The fourth equality follows from the existence of one-to-one correspondence between paths of the form $t : i \to j$ and their reverse paths $t_{rev} : j \to i$ traversing the same edges in the opposite direction. Every mapped pair (t, t_{rev}) of paths have the same length and the probabilities of the two paths are related as follows: $P(t) = \frac{w_j}{w_i} P(t_{rev})$. ■

B. Handling tied EI values

One possible corner case for the kNN algorithm is the existence of ties in the EI. If two or more nodes have the same EI with respect to a query node their feasibility intervals will overlap for any instance of K during expansion. If a series of tied nodes happen to overlap with the k-th and $(k+1)$-th positions in the top-k order, the whole network will have to be expanded in order to answer the query. Ties are possible due to nodes that map to each other in graph isomorphisms. Such ties need to be detected early in the expansion as otherwise the exact top-k neighbors will require computing the actual stationary distribution. Detecting a graph isomorphism is NP-complete, since it is polynomial time reducible to the graph isomorphism problem [23]. However, we can cheaply detect a class of automorphisms consisting of nodes, connected "in parallel" to exactly the same set of neighbors. We detect most ties using this efficient check while performing the expansion process.