RoundTripRank: Graph-based Proximity with Importance and Specificity*

Yuan Fang #$, Kevin Chen-Chuan Chang #$, Hady W. Lauw +
fang2@illinois.edu, kcchang@illinois.edu, hadywlauw@smu.edu.sg
# University of Illinois at Urbana-Champaign, 201 N. Goodwin Avenue, Urbana, IL 61801, USA
$ Advanced Digital Sciences Center, 1 Fusionopolis Way, #08-10 Connexis N. Tower, Singapore 138632
+ Singapore Management University, 80 Stamford Road, Singapore 178902

Abstract—**Graph-based proximity has many applications with different ranking needs. However, most previous works only stress the sense of *importance* by finding "popular" results for a query. Often times important results are overly general without being well-tailored to the query, lacking a sense of *specificity*—which only emerges recently. Even then, the two senses are treated independently, and only combined empirically. In this paper, we generalize the well-studied importance-based random walk into a *round trip* and develop RoundTripRank, seamlessly integrating specificity and importance in one coherent process. We also recognize the need for a flexible trade-off between the two senses, and further develop RoundTripRank+ based on a scheme of *hybrid random surfers*. For efficient computation, we start with a basic model that decomposes RoundTripRank into smaller units. For each unit, we apply a novel two-stage bounds updating framework, enabling an online top-K algorithm 2SBound. Finally, our experiments show that RoundTripRank and RoundTripRank+ are robust over various ranking tasks, and 2SBound enables scalable online processing.**

I. Introduction

Graphs are abundant in the real world, such as a *bibliographic network* connecting authors, papers, terms and venues, and a *query log graph* linking search phrases and their clicked URLs. In this work, we study the problem of ranking nodes on a graph by their "proximity" to a given query. Consider a *graph* $G = (V, E)$ with *nodes* V and *edges* E. Edges are directed and possibly weighted, where an undirected edge is treated as bidirectional. As each edge embeds certain semantic relationship, through these edges the proximity of two nodes can be quantified, reflecting a degree of match between the two nodes. Thus, given a *query node* $q \in V$, how to measure the *proximity* of every node $v \in V$ to q? More generally, a query can comprise multiple nodes on the graph.

We motivate graph-based proximity with some examples below, which illustrate quite varying ranking tasks (of different natures, as we will see) even on the same graph.

- *Task A (Expert):* on a bibliographic network, given a paper, who are the experts best suited to review it?

* This material is based upon work partially supported by NSF Grant IIS 1018723, the Advanced Digital Sciences Center of the University of Illinois at Urbana-Champaign, and the Agency for Science, Technology and Research of Singapore. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the funding agencies.

(a) *Importance*-based	(b) *Specificity*-based	(c) *balanced*
SIGMOD	Spatio-Temporal Databases	VLDB
VLDB	Spatio-Temporal Data Mining	Spatio-Temporal Databases
ICDE	Temporal Aspects in Info. Sys.	ACM GIS

Fig. 1. Venues for "spatio temporal data" (real results from Sect. VI).

- *Task B (Venue):* on a bibliographic network, given some terms as a topic, what are the matching venues?
- *Task C (Relevant URL):* on a query log graph, given a search phrase, what are the relevant URLs?
- *Task D (Equivalent search):* on a query log graph, given a search phrase, what are the equivalent phrases for the same concept (*e.g.*, "google mail" and "gmail")?

Taking Task B as an example, given a query, say q = "spatio temporal data" (comprising three term nodes), what are the matching venues? One effective family of works build upon random walks on the graph, such as Personalized PageRank (PPR) [1], [2] and its variants [3], [4], which find venues that can be easily reached from q through the edges. Fig. 1(a) shows venues found by PPR in our actual experiments—they are well-known venues like VLDB where the topic in q appears. As pioneered by PPR, ranking nodes by their *reachability from* the query intuitively captures the sense of *importance*.

However, such importance-based ranking is not ideal in many cases, as a node can be easily reached from q simply due to its popularity for being linked from a broad range of nodes, without being particularly tailored to q. As we re-examine Fig. 1(a), the venues are only categorically related to our query as general data-centric topics. In fact, the same venues would be ranked as important for almost any data-centric topic, *e.g.*, "information integration." How about those venues especially tailored to the query, as Fig. 1(b) shows?

Thus, we argue that the sense of *specificity* is missing, as the results in Fig. 1(a) are important, but not specific, to the query. As a key insight, proximity shall naturally encompass the dual senses of importance and specificity. A balanced ranking in the two senses is often more useful, such as Fig. 1(c). Among the venues, VLDB is important, Spatio-Temporal Databases is specific, and the previously undiscovered ACM GIS is balanced between the two senses.

Although intuitive and appealing, dual-sensed graph proximity with both importance and specificity is not explored until recently by Hristidis *et al* [5]. Their work hypothesizes various forms of specificity, such as the inverse of node degree, the inverse of global ObjectRank [3], and Inverse ObjectRank [5] (which is ObjectRank on the graph with reversed edges). Each form of specificity is then combined with the reachability-based importance in different ways. While a novel insight, their approach suffers from two fundamental drawbacks.

First, they treat the two senses independently and combine them heuristically, *lacking a unifying model* to coherently capture both. As they model specificity upon quite disparate concepts, there is no definitive view on the best form. Moreover, it is unclear which way of combination with importance is superior. In particular, the form of specificity (*e.g.*, inverse of node degree [5]) can inherently differ from the reachability-based importance, and thus their combination appears ad-hoc. Our experiments also reveal that, even empirically, they tend to perform unevenly on different ranking tasks.

Second, they pursue a *fixed trade-off* between importance and specificity, while many applications require flexible trade-offs, as different users or ranking tasks often involve different objectives. Let us revisit the example tasks given earlier.

- Task A (Expert): Reviewers balanced between importance and specificity are preferred. An important but broad expert may miss some latest development, while a very specific researcher like a student may lack authoritativeness.
- Task B (Venue): Different scenarios require varying senses. For instance, to build some background on a topic, a specific book chapter may be preferred. In contrast, to submit one's best work, important venues are often sought after.
- Task C (Relevant URL): Users often prefer important URLs for monetary transactions, such as booking a hotel.
- Task D (Equivalent search): Equivalent phrases are inherently specific, as they ideally represent the same concept.

As our experiments validated, it is rare to involve only a single sense in a task. The above tasks actually require some trade-off between the two senses, *e.g.*, Task C values importance, but it still needs some specificity—when booking a hotel, an important "travel" site is expected. However, what sense weighs more varies across tasks as we have analyzed. We note that the benefit of a flexible trade-off has not been recognized in previous works, and thus they miss the potential to improve ranking by catering to different objectives.

Thus, as our main thesis, we believe that an effective dual-sensed graph proximity measure must entail:

- **Unified modeling.** Instead of combining the independent forms of importance and specificity, the dual senses must be seamlessly integrated under a single unifying process—coherently based on the same random walk principle as the well-studied importance-based ranking.
- **Customizable trade-off.** Instead of a "one-size-fits-all" solution, the trade-off between importance and specificity must be flexible—the unified modeling shall also allow for customizable trade-offs between the dual senses.

Towards a unified modeling, as the **first contribution** (Sect. III), we propose *RoundTripRank* to integrate the dual

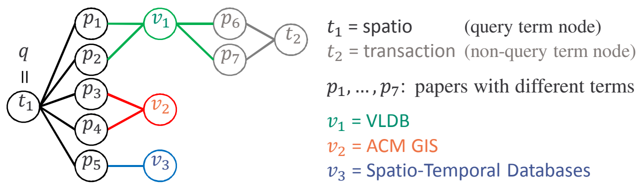
q
$\|$
t_1
p_1
p_2
p_3
p_4
p_5
v_1
v_2
v_3
p_6
p_7
t_2
t_1 = spatio (query term node)
t_2 = transaction (non-query term node)
$p_1, \dots, p_7$: papers with different terms
v_1 = VLDB
v_2 = ACM GIS
v_3 = Spatio-Temporal Databases

Fig. 2. A toy graph for a bibliographic network.

senses of importance and specificity in a round trip. Given a node q as the query, a *round trip* w.r.t. a target node v starts from q, goes through v, and finally returns to q. As we believe importance and specificity naturally exist in symmetric forms, this round trip generalizes the reachability-based importance, capturing not only the reachability from the query (to the target) as importance, but also the reachability to the query (from the target) as specificity.

As an example, let us rank venues v_1, v_2, v_3 on Fig. 2 for query $q = t_1$. v_1 is intuitively more important than v_3, as v_1 has two papers p_1, p_2 on t_1, but v_3 has only one p_5. Observe that from t_1 it is easier to reach v_1 than v_3. However, v_1 is less specific than v_3, as v_1 accepts two off-topic papers p_6, p_7, but v_3 does not. Observe that it is easier to return to t_1 from v_3 than from v_1. Thus, a round trip—from t_1, through a target venue v_i, and back to t_1—integrates both senses, favoring v_2 for being not only important (with two papers p_3, p_4 about t_1), but also specific (without off-topic papers).

Next, towards a customizable trade-off, as the **second contribution** (Sect. IV), we make a further generalization and develop *RoundTripRank+* based on a novel scheme of *hybrid random surfers*. We consider random surfers of different minds who may "shortcut" different parts of the round trip to reflect their preference—some prefer importance, some prefer specificity, and others prefer a balance. In particular, we use different compositions of hybrid random surfers to mimic the varying objectives in importance and specificity.

A potential drawback of RoundTripRank is its computational overhead. Each "outgoing trip" from a query node to every target node can be conceptually paired with many different "returning trips" to complete a round trip, causing an exponential blow-up to the possible round trips. Fortunately, we are able to decompose RoundTripRank into smaller units, where each unit can be computed without actually considering the large number of round trips. Moreover, since users are often interested in getting a small number of top results quickly, as the **third contribution** (Sect. V), we introduce a top-K algorithm called 2SBound, building upon our decomposition of RoundTripRank as well as a novel two-stage bounds updating framework. To handle very large graphs, we further adapt 2SBound to a distributed environment.

Finally, as the **fourth contribution** (Sect. VI), we conduct extensive experiments on two real-world graphs. In terms of effectiveness, we consistently and significantly outperform existing baselines over varying ranking tasks. On average, RoundTripRank improves over existing mono-sensed measures by 10%, and RoundTripRank+ improves over existing dual-sensed measures by 7%. For efficiency, 2SBound is 2–10 times faster than various baseline solutions.

II. Related Work

Importance and specificity. While the sense of importance to a query has been studied since PPR [1], [2], the sense of specificity is only recently explored [5] with a few heuristic forms such as the inverse of node degree, the inverse of global ObjectRank [3], and Inverse ObjectRank. We pursue a different approach by generalizing the importance-based random walk to a round trip, which coherently captures both importance and specificity in a single measure.

Mono-sensed proximity. Most existing graph-based proximity measures are *mono-sensed*, matching the query in only one sense—importance, specificity, or simply a vague sense of "closeness" without a finer interpretation. Examples include (a) importance: PPR [1], [2] and its variants [3], [4], [6]; (b) specificity: backward random walk [6] and Inverse ObjectRank [5]; (c) no finer interpretation: AdamicAdar [7], SimRank [8] and escape probability [9], [10]. As Sect. I motivated, they are inadequate since most tasks require some trade-off between importance and specificity.

Dual-sensed proximity. Some recent works, such as truncated commute time [11], the harmonic mean of "precision" and "recall" [12], [13], and ObjSqrtInv [5], can be deemed *dual-sensed* measures. We *first* note that none of them, except [5], connects explicitly to importance and specificity. While [11] does not argue any finer interpretation, [12] and [13] measure probabilistic precision and recall. Unlike importance and specificity which are directly defined on a query, precision and recall are indirectly measured against an underlying "relevant set" of nodes for a query. Despite the contrast, precision and recall seem to intuitively parallel specificity and importance, which may warrant further investigation. *Second*, most works [12], [13], [5] heuristically combine the two senses as two independent measures, lacking an underpinning model to unify them coherently. We instead develop a round trip to directly entail both senses in a single measure. *Third*, all of them ignore different trade-offs between importance and specificity across tasks. In contrast, our round trip model can be generalized to mimic the varying trade-offs using a scheme of hybrid random surfers, allowing for a customizable trade-off.

Efficient query processing. We also study online top-K processing for RoundTripRank. While we adopt a branch-and-bound graph expansion algorithm [14] as the backbone, significant innovations are still required to realize it, which include our original bounds decomposition, two-stage framework for bounds updating, and a cluster-based distributed architecture for scaling up. Our realization not only enables online search, but also requires no precomputation, unlike many previous graph proximity works [2], [3], [15], [16].

III. RoundTripRank: Walking in Round Trips

In this section, we develop RoundTripRank to integrate importance and specificity in a coherent round trip, followed by a basic computational model.

A. RoundTripRank: Balancing Importance and Specificity

Towards a dual-sensed measure, we generalize PPR [1], [2] for its effectiveness in measuring importance. The generalization would capture both importance and specificity in a coherent random walk, instead of treating them independently and combining them heuristically.

Personalized PageRank (PPR). To prepare the generalization, we first review PPR [1], [2] as an effective measure of importance. On a graph $G = (V, E)$, a random surfer is initially at a given query node $q \in V$. (We ignore multi-node queries for now.) At each step, she has a probability $1 - \alpha$ to move to a neighbor randomly, and a probability α to teleport to q. Her stationary probability at node v is v's PPR for q, denoted PPR(q, v), indicating v's importance to q.

Trip-view of PPR. To generalize PPR, we need an alternative view with the notions of a "trip" and a "target." Let L be a random variable (of some distribution). Starting from q, the surfer takes L random steps on the graph. She then teleports to q to restart the walk. This L-step walk is a *trip* from q, reaching v as the *target*. We call the probability that v is the target of a trip from q the *F-Rank* of v for q ($\underline{\text{rank}}$ by reachability $\underline{\text{f}}$rom query), denoted $f(q, v)$. Representing a trip as a sequence of visited nodes $W_0, \ldots, W_L$, we have:

$$f(q, v) \triangleq p(W_L = v | W_0 = q). \tag{1}$$

The walk length L captures the range of influence from the starting node—how far the surfer can walk until the influence becomes too weak and requires a restart. In particular, if L is geometric with parameter $\alpha \in (0, 1)$, *i.e.*, $p(L = \ell) = (1 - \alpha)^\ell \alpha$, F-Rank is equivalent to PPR.

Proposition 1 (From Fogaras et al. [17]): For a query node q, a node v's PPR with teleporting probability α equals its F-Rank with walk length $L \sim \text{Geo}(\alpha)$:

$$f(q, v) \equiv \text{PPR}(q, v). \tag{2}$$

In general, a geometric L is effective as it gives longer walk lengths smaller probabilities, agreeing with the intuition that the influence from a node weakens along a path. Unless otherwise stated, we will assume a geometric L for F-Rank, which captures importance just as PPR does.

Our proposal: from importance to specificity. F-Rank treats importance as reachability from q in the L steps before teleportation. We can interpret this notion of importance by viewing a directed edge $a \to b$ as "a cites b." Note that, more generally, depending on the context a cites b may mean a (paper) mentions b (term), or a (venue) accepts b (paper), or a (paper) supports b (venue), *etc.* Naturally, if a node v is more important to q than another node v' is, q is more likely to cite v than v', directly or indirectly. In other words, the more likely v can be reached from q via a directed path, the more important v is to q. As an example, in Fig. 2 where $q = t_1$, observe that v_1 or v_2 (each with two papers about t_1) is intuitively more important to t_1 than v_3 (with only one such paper). Indeed, from q it is easier to reach v_1 or v_2 than v_3.

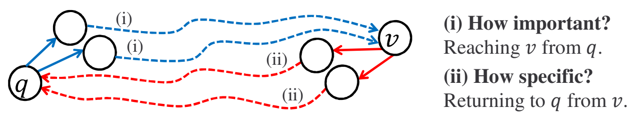

Fig. 3. Illustration of round trips. A dotted line indicates a random walk path that may pass through other nodes not shown here.

Interestingly, the citation analogy enables us to treat specificity in a symmetric form to importance. Imagine two nodes v and v', where v is more specific to q than v' is. Naturally, v tends to cite a focused set of nodes which are particularly tailored to q, whereas v' tends to cite a diverse set of nodes which may not match q at all. Hence, q is more likely to be cited by v than by v', directly or indirectly. In other words, the more likely q can be reached from v via a directed path, the more specific v is to q. In Fig. 2 where $q = t_1$, v_2 or v_3 (accepting no off-topic papers) is intuitively more specific than v_1 (accepting two such papers). Observe that it is more likely to reach t_1 from v_2 or v_3 than from v_1.

As we will see next, the importance-based random walk for F-Rank does leave room to encompass the symmetric view of specificity, unifying both senses in one coherent model.

RoundTripRank. To integrate specificity, we analyze the formulation of F-Rank. The surfer not only reaches some target v from q in L steps, but also returns to q from v by teleportation (to restart the walk). We can thus view this process, at least conceptually, as a *round trip* from q to v, then back to q. However, the surfer goes back to q from v via a trivial teleportation. Suppose that, instead of teleporting to q, the random surfer takes an actual walk back to q in L' steps. The more likely to return to q from v, the more specific v is to q. Hence, the materialization into an actual walk embeds the sense of specificity into these additional L' steps, symmetric to importance embedded in the first L steps.

Thus, we consider a round trip starting and ending at q, through v, as depicted in Fig. 3. Intuitively, if v is important to q, the surfer will easily reach v from q in L steps; once at v, if v is specific to q, the surfer will easily return to q in L' steps. By finding out how likely a round trip goes through v, we capture both senses in one coherent random walk.

As another degenerated case symmetric to F-Rank, we can make the first L steps a trivial teleportation instead of an actual walk. We will further exploit it to enable a customized trade-off between importance and specificity in Sect. IV.

Now we formally introduce the round trip concept, upon which RoundTripRank can be defined.

Definition 1 (Round Trip): A *round trip* is a random walk of $L + L'$ steps starting and ending at the same node, *i.e.*, $W_0 = W_{L+L'}$ (L, L' are i.i.d. geometric random variables). The node after the first L steps, W_L, is the *target* of the round trip. ■

Definition 2 (RoundTripRank): RoundTripRank of a node v for a query node q, denoted $r(q, v)$, is defined as: given that a random surfer starting at q completes a round trip, the probability that this round trip has the target v.

$$r(q, v) \triangleq p(W_L = v | W_0 = W_{L+L'}, W_0 = q) \qquad (3)$$

Target v	Round trip from t_1	Probability	RoundTripRank $r(t_1, v)$
v_1	$t_1 \to p_1 \to v_1 \to p_1 \to t_1$	0.0125	$\propto 0.0125 \times 4 = 0.05$
	$t_1 \to p_1 \to v_1 \to p_2 \to t_1$	0.0125	
	$t_1 \to p_2 \to v_1 \to p_1 \to t_1$	0.0125	
	$t_1 \to p_2 \to v_1 \to p_2 \to t_1$	0.0125	
v_2	$t_1 \to p_3 \to v_2 \to p_3 \to t_1$	0.025	$\propto 0.025 \times 4 = 0.1$
	$t_1 \to p_3 \to v_2 \to p_4 \to t_1$	0.025	
	$t_1 \to p_4 \to v_2 \to p_3 \to t_1$	0.025	
	$t_1 \to p_4 \to v_2 \to p_4 \to t_1$	0.025	
v_3	$t_1 \to p_5 \to v_3 \to p_5 \to t_1$	0.05	$\propto 0.05$
t_1	$t_1 \to p_1 \to t_1 \to p_1 \to t_1$	0.01	$\propto 0.01 \times 25 = 0.25$
	24 more …	0.01 each	
others	none	0	0

Fig. 4. RoundTripRank for the toy example with constant $L = L' = 2$.

More generally, a query can consist of multiple nodes, and the round trip can start from any of them. Similar to the *Linearity Theorem* [2] for PPR, RoundTripRank for a multi-node query can be equivalently expressed as a linear function of RoundTripRank for each node in the query. Hence, the rest of our discussion only considers single-node queries.

Toy example. To illustrate RoundTripRank, we revisit the toy graph in Fig. 2. For simplicity, we assume a constant walk length $L = L' = 2$, and equal weights for all edges. Applying Bayes' law to Eq. 3, RoundTripRank of each target v for the query node t_1 can be computed as follows:

$$\begin{aligned} r(t_1, v) &= \frac{p(W_0 = W_{L+L'}, W_0 = t_1, W_L = v)}{p(W_0 = W_{L+L'}, W_0 = t_1)} \\ &\propto p(W_0 = W_{L+L'}, W_0 = t_1, W_L = v) \end{aligned} \qquad (4)$$

where the denominator is v-independent and thus does not affect ranking ("$\propto$" stands for equivalence in ranking).

The detailed computation is shown in Fig. 4. We list all round trips starting at t_1, grouped by their targets. The probability of a round trip can be found using node degrees, *e.g.*, $p(t_1 \to p_1 \to v_1 \to p_1 \to t_1) = \frac{1}{5} \cdot \frac{1}{2} \cdot \frac{1}{4} \cdot \frac{1}{2} = 0.0125$.

To see the intuition, let us compare v_1, v_2, v_3. As discussed earlier, v_2 is as important as but more specific than v_1, and v_2 is as specific as but more important than v_3. Hence, v_2 has larger RoundTripRank than v_1 and v_3 for being both important and specific. Also note t_1 itself has the largest RoundTripRank, which is intuitive that self-proximity is high.

B. Basic Computational Model For RoundTripRank

In Fig. 4, we compute RoundTripRank by summing the round trips for each target. However, the number of such round trips increases exponentially on larger graphs, calling for a more practical computational model.

To begin with, we examine how PPR/F-Rank can be computed. One simple method applies iterative computation [1], which is linear in the number of nodes and edges:

$$f^{(i+1)}(q, v) = \alpha I(q, v) + (1 - \alpha) \sum_{v' \in \mathrm{In}(v)} M_{v'v} f^{(i)}(q, v'), \qquad (5)$$

where $I(q, v) = 1$ if $q = v$, or 0 if $q \neq v$; $\mathrm{In}(v)$ is the set of v's in-neighbors; $M_{v'v}$ is the one-step transition probability from v' to v; α is the teleporting probability. Convergence is guaranteed on an irreducible and aperiodic graph [18].

Thus, we wonder if RoundTripRank can also be computed iteratively. Simply put, iterative computation is possible for F-Rank as it can be defined recursively using its in-neighbors—to reach a node v from q, the surfer must first reach one of v's in-neighbors v'. In other words, as the target v is at an "end" of a walk, we can reduce the walk by one step to utilize the F-Rank of v'. However, in RoundTripRank, a round trip merely passes through v which is not at an "end" of the walk, and thus no apparent iterative computation exists.

Consequently, we resort to divide and conquer. Can we decouple a round trip into smaller units with the target at the "end" of each unit, such that we can iteratively compute each? It turns out we can rewrite RoundTripRank defined in Eq. 3 into an equivalent form with two decoupled units.

Proposition 2: The RoundTripRank of v for a query node q can be expressed as follows with rank equivalence:

$$r(q,v) \propto p(W_L = v | W_0 = q) p(W_{L'} = q | W_0 = v) \quad (6)$$

PROOF:

$$\begin{aligned} r(q,v) &\overset{1}{=} p(W_L = v, W_0 = W_{L+L'} | W_0 = q) / p(W_0 = W_{L+L'} | W_0 = q) \\ &\overset{2}{\propto} p(W_L = v, W_{L+L'} = q | W_0 = q) \\ &\overset{3}{=} p(W_{L+L'} = q | W_L = v, W_0 = q) p(W_L = v | W_0 = q) \\ &\overset{4}{=} p(W_{L+L'} = q | W_L = v) p(W_L = v | W_0 = q) \\ &\overset{5}{=} p(W_{L'} = q | W_0 = v) p(W_L = v | W_0 = q) \end{aligned}$$

In step 2, the v-independent denominator is dropped without altering ranking. In step 4, given W_L, $W_{L+L'}$ is conditionally independent of W_0 in a Markovian walk. In step 5, a shift in time by $-L$ does not affect the transition probability. ■

In particular, $p(W_L = v|W_0 = q)$ is the F-Rank (Eq. 1), *i.e.*, the reachability from q to v. Similarly, $p(W_{L'} = q|W_0 = v)$ is the reachability from v to q, which we call *T-Rank* (rank by reachability to query), denoted $t(q, v)$. Hence,

$$r(q,v) \propto f(q,v) t(q,v) \quad (7)$$

While F-Rank can be computed iteratively (see Eq. 5), T-Rank can be computed in a symmetric way, where $\text{Out}(v)$ is the set of v's out-neighbors:

$$t^{(i+1)}(q,v) = \alpha I(q,v) + (1-\alpha) \sum_{v' \in \text{Out}(v)} M_{vv'} t^{(i)}(q,v'). \quad (8)$$

As a minor caveat, if a directed path exists from q to v but not from v to q, $f(q,v) > 0$ but $t(q,v) = 0$. Hence, it may be counter-intuitive that $r(q,v) = 0$ no matter how large $f(q,v)$ is. Fortunately, this does not happen on an irreducible (*i.e.*, strongly connected) graph. In practice, we can always make a graph irreducible by adding some dummy edges [18].

IV. ROUNDTRIPRANK+: CUSTOMIZING THE TRADE-OFF

Although RoundTripRank balances importance and specificity, different tasks often require varying trade-offs, as Sect. I motivated. To this end, we propose RoundTripRank+.

A. RoundTripRank+ with Hybrid Random Surfers

Recall that we generalize F-Rank by materializing the teleportation from v back to q into an actual walk. Symmetrically, in a round trip we can reduce the actual walk from q to each v to a teleportation. In other words, the surfer may shortcut the first L steps by teleporting to each v, ignoring importance. She continues the next L' steps for specificity.

Now consider some *hybrid random surfers* Ω consisting of different-minded surfers in three disjoint groups: (a) Ω_{11}, seeking targets balanced between importance and specificity, *e.g.*, ACM GIS for q = "spatio temporal data"; (b) Ω_{10}, seeking important targets, *e.g.*, VLDB for q; (c) Ω_{01}, seeking specific targets, *e.g.*, Spatio-Temporal Databases for q.

To reflect their intents, the surfers potentially take variants of round trips by shortcutting as discussed earlier. Given a query node q and a target v, the surfers in Ω_{11} take regular round trips. However, the surfers in Ω_{10}, after reaching v in L steps, shortcut the next L' steps by teleporting to q, *i.e.*, $\forall \omega \in \Omega_{10}, p(W^\omega_{L+L'} = q | W^\omega_L = v) = 1$ where $\{W^\omega_\ell : \ell \geq 0\}$ represents the sequence of nodes visited by ω. Likewise, the surfers in Ω_{01} shortcut the first L steps by teleporting to v, *i.e.*, $\forall \omega \in \Omega_{01}, p(W^\omega_L = v | W^\omega_0 = q) = 1$.

Using different compositions of Ω, *i.e.*, different distributions of the three groups, we can customize the trade-off between importance and specificity with RoundTripRank+.

Definition 3 (RoundTripRank+): Given hybrid random surfers Ω, *RoundTripRank+* of v for a query node q, denoted $r_\Omega(q,v)$, is defined as: given that each random surfer in Ω *independently* completes a round trip from q with a common target, the probability that the common target is v.

$$r_\Omega(q,v) \triangleq p(x = v | \forall_{\omega \in \Omega} : W^\omega_0 = W^\omega_{L+L'} = q, W^\omega_L = x) \ (9)$$

We illustrate the customizability of RoundTripRank+ with some special cases. (a) $\Omega = \Omega_{11}$, *i.e.*, $\Omega_{10} = \Omega_{01} = \emptyset$. It reduces to (the non-customizable) RoundTripRank for a balance between the two senses, since all the surfers only walk in regular round trips. (b) $\Omega = \Omega_{10}$ reduces to F-Rank and captures only importance, since all the surfers teleport back to q from v. (c) Likewise, $\Omega = \Omega_{01}$ reduces to T-Rank and captures only specificity.

However, to customize the trade-off we need to adjust the compositions of Ω, which involves three parameters $|\Omega_{11}|, |\Omega_{10}|, |\Omega_{01}|$. To simplify it, we will introduce a computational model with only one parameter.

B. Basic Computational Model for RoundTripRank+

We first rewrite RoundTripRank+ in Eq. 9 into an equivalent form, similar to Proposition 2.

Proposition 3: Given hybrid random surfers Ω consisting of $(\Omega_{11}, \Omega_{10}, \Omega_{01})$, the RoundTripRank+ of v for a query node q can be expressed as follows with rank equivalence:

$$r_\Omega(q,v) \propto f(q,v)^{|\Omega_{11}|+|\Omega_{10}|} \cdot t(q,v)^{|\Omega_{11}|+|\Omega_{01}|}. \quad (10)$$

PROOF (SKETCH): As each surfer walks independently, we can factor $r_\Omega(q,v)$ into the product of individual surfers. Then, for each surfer, apply Proposition 2. Since some surfers shortcut the round trip, *e.g.*, $p(W^\omega_{L+L'} = q | W^\omega_L = v) = 1$ for $\omega \in \Omega_{10}$, and similarly for $\omega \in \Omega_{01}$, we will get Eq. 10. ■

We can further normalize the exponents in Eq. 10 without altering ranking, since the power function is monotonic:

$$r_\Omega(q,v) \propto \left(f(q,v)^{|\Omega_{11}|+|\Omega_{10}|} \cdot t(q,v)^{|\Omega_{11}|+|\Omega_{01}|}\right)^{\frac{1}{|\Omega|+|\Omega_{11}|}}$$
$$= f(q,v)^{\frac{|\Omega_{11}|+|\Omega_{10}|}{|\Omega|+|\Omega_{11}|}} \cdot t(q,v)^{\frac{|\Omega_{11}|+|\Omega_{01}|}{|\Omega|+|\Omega_{11}|}} \quad (11)$$

Letting $\beta \triangleq \frac{|\Omega_{11}|+|\Omega_{01}|}{|\Omega|+|\Omega_{11}|} \in [0,1]$, we can rewrite Eq. 11 below:

$$r_\Omega(q,v) \propto r_\beta(q,v) = f(q,v)^{1-\beta} \cdot t(q,v)^\beta \quad (12)$$

Intuitively, β is the fraction of objectives by all surfers that are specificity (note that each surfer in Ω_{11} has two objectives). We call β the *specificity bias*—a large β favors specificity over importance. A unique ranking can be determined by a given β, which is itself uniquely determinable from any given hybrid surfers Ω. Hence, we can adjust β directly to customize the trade-off between importance and specificity. That is, we compute RoundTripRank+ as $r_\beta(q,v)$ in Eq. 12 for a chosen β. Choosing β for each ranking task enables a customized trade-off, which corresponds to, as its physical meaning, adjusting the composition of hybrid surfers. As special cases, $\beta = 0$ (or 1) reduces to F-Rank (or T-Rank) for only importance (or specificity); $\beta = 0.5$ reduces to RoundTripRank.

To find the optimal β for a ranking task, we may use some development queries, or consult domain experts. Users may also specify β directly. As a last resort, we can always fall back to the default $\beta = 0.5$, which outperforms the extreme cases of $\beta = 0$ or 1 in our experiments.

V. Online Top-K Processing

The basic computational models discussed earlier require computing F-Rank and T-Rank. However, their iterative computation in Eq. 5 and 8 is not scalable, as it involves multiple passes of the entire graph for each query.

In many applications, users only need to quickly get a small number of top results, which could be a close approximation. Hence, we propose an online approximate top-K method 2SBound, followed by a distributed solution to handle larger graphs. Our discussion only covers RoundTripRank, but extending to RoundTripRank+ is straightforward.

A. Approximate Top-K Processing: 2SBound

We propose Two-Stage Bounding, or 2SBound, for online top-K processing. As its backbone, we adopt the standard branch-and-bound expansion on graphs (*e.g.*, [14]). However, its realization for RoundTripRank still requires significant innovations, hinging on our original basic computational model and two-stage bounds updating framework.

We outline 2SBound in Algorithm 1. Given a query node q, we maintain a *neighborhood* S, which is a subset of the nodes: $S \subseteq V$. We refer to those nodes currently in S as *seen* nodes, and those outside S as *unseen* nodes. We also maintain a set of bounds Δ for RoundTripRank: (a) each seen node is sandwiched by an upper bound $\hat{r}(q,v)$ and a lower bound $\check{r}(q,v)$, *i.e.*, $\check{r}(q,v) \le r(q,v) \le \hat{r}(q,v)$, $\forall v \in S$; (b) all unseen nodes have a common *unseen* upper bound $\hat{r}(q)$, *i.e.*, $r(q,v) \le \hat{r}(q)$, $\forall v \notin S$. Starting with an empty neighborhood S, we repeatedly expand S and update the bounds Δ. After each expansion and updating, we obtain a candidate top K ranking T_K according to the lower bounds $\check{r}(q,*)$. The process stops if T_K satisfy the top-K conditions (see Sect. V-A1).

Lastly, to concretize the algorithm for RoundTripRank, we must further solve the core problem of bounds updating, which relies on our bounds decomposition (see Sect. V-A2) and two-stage bounds updating framework (see Sect. V-A3).

Algorithm 1: Two-Stage Bounding (2SBound)

```
Input: graph $G$; query node $q$; number of desired results $K$
Output: top-$K$ ranking $T_K$ of nodes $v$ on $G$ by $r(q,v)$
1 Neighborhood $S \leftarrow \emptyset$;
2 repeat
3     Two-stage bounds updating framework:        // Sect. V-A3
4         Stage I: Expand $S$ and initialize bounds $\Delta$;
5         Stage II: Iteratively refine $\Delta$ over $S$;
6     $T_K \leftarrow$ current top-$K$ by lower bounds;
7 until $T_K$ satisfies top-$K$ conditions;      // Sect. V-A1
8 return $T_K$.
```

1) Top-K Conditions

After each neighborhood expansion and updating, we try to decide the top-K nodes if $|S| \ge K$. The nodes in S are sorted as $v_1, \ldots, v_{|S|}$ such that $\check{r}(q,v_1) \ge \ldots \ge \check{r}(q,v_{|S|})$. $T_K = \langle v_1, \ldots, v_K \rangle$ is a candidate top-K ranking, which will be further judged by two conditions below, assuming $\epsilon = 0$ for the moment. *First*, Eq. 13 ensures that T_K contains correct top K nodes (in no particular order), since their lower bounds are no less than the largest upper bound of all other nodes. *Second*, Eq. 14 ensures that T_K is ordered correctly.

$$\check{r}(q,v_K) > \max\left\{\hat{r}(q,v_{K+1}), \ldots, \hat{r}(q,v_{|S|}), \hat{r}(q)\right\} - \epsilon \quad (13)$$
$$\check{r}(q,v_i) > \hat{r}(q,v_{i+1}) - \epsilon, \quad \forall i \in \{1,\ldots,K-1\} \quad (14)$$

In many applications, it is desirable to speed up the computation by slightly sacrificing ranking quality. Hence, we relax the conditions in Eq. 13–14 with a positive *slack* parameter ϵ. Accordingly, we obtain an ϵ-approximate top-K ranking, which (a) does not miss any node whose score exceeds v_K's by at least ϵ; (b) does not swap the order of two nodes if their scores differ by at least ϵ.

2) Bounds Decomposition

Our computational model (Eq. 7) implies that the bounds for RoundTripRank can be computed based on the bounds for F-Rank and T-Rank. Thus, we maintain two neighborhoods: the f-neighborhood S_f for F-Rank, and the t-neighborhood S_t for T-Rank. In general $S_f \neq S_t$, because their expansions differ as we shall see. We ultimately define the r-neighborhood S for RoundTripRank using S_f and S_t.

Specifically, $\forall v \in S_f$, let $\hat{f}(q,v)$ and $\check{f}(q,v)$ denote the upper and lower bounds for F-Rank. Similarly, $\forall v \in S_t$, let $\hat{t}(q,v)$ and $\check{t}(q,v)$ denote the upper and lower bounds for T-Rank. By defining the r-neighborhood as the set of nodes common in both the f- and t-neighborhoods, *i.e.*, $S = S_f \cap S_t$, we can compute the bounds for RoundTripRank, $\forall v \in S$:

$$\check{r}(q,v) = \check{f}(q,v)\check{t}(q,v); \qquad \hat{r}(q,v) = \hat{f}(q,v)\hat{t}(q,v) \quad (15)$$

Additionally, to compute the unseen upper bound $\hat{r}(q)$, we also maintain its counterparts $\hat{f}(q)$ and $\hat{t}(q)$ for F-Rank and T-Rank, respectively. However, $\hat{r}(q)$ does not simply decompose as $\hat{f}(q)\hat{t}(q)$. Since $S = S_f \cap S_t$, some unseen nodes by S may be "seen" by either S_f or S_t, but not both. These nodes $v \in S_f \backslash S$ or $v \in S_t \backslash S$ have individual upper bound $\hat{f}(q,v)$ or $\hat{t}(q,v)$. Hence, we can compute $\hat{r}(q)$ as follows:

$$\hat{r}(q) = \max\left\{\hat{f}(q)\hat{t}(q), \max_{v \in S_f \backslash S} \hat{f}(q,v)\hat{t}(q), \max_{v \in S_t \backslash S} \hat{f}(q)\hat{t}(q,v)\right\} \quad (16)$$

3) Two-Stage Bounds Updating Framework

As just explained, to obtain the bounds for RoundTripRank, we only need those for F-Rank and T-Rank. Given the graph and the current neighborhood (S_f or S_t) as input, we propose a novel two-stage framework common to both S_f and S_t, although their realizations differ. In particular, we update bounds for each seen node by considering not only the node itself for *initialization* (Stage I), but also its relationship with its neighbors for *iterative refinement* (Stage II).

To simplify notations in the common framework, let S_x denote either the f- or t-neighborhood, *i.e.*, x refers to either f or t. Likewise, $\check{x}(\cdot)$ and $\hat{x}(\cdot)$ denote the lower and upper bounds for either F-Rank or T-Rank.

Stage I: Expansion and initialization. Expand the given neighborhood S_x. Assign lower and upper bounds $\check{x}^{(0)}(q,v)$ and $\hat{x}^{(0)}(q,v)$ for each $v \in S_x$, and the unseen upper bound $\hat{x}^{(0)}(q)$ for all nodes $v \notin S_x$. We are essentially initializing the bounds—as the superscript (0) indicates—for each node individually, before further refining them iteratively using neighbors in Stage II. To compute such initial values, we can leverage previous works (*e.g.*, [19], [16], [20]).

Stage II: Iterative refinement. We further improve the initial bounds from Stage I by exploiting the relationships between nodes. As implied by the naïve computation in Eq. 5 (or Eq. 8), the F-Rank (or T-Rank) of a node v can be expressed in terms of its in- (or out-) neighbors' values. Since Eq. 5 and 8 are monotonic additions, by using the lower or upper bound of each summand that involves a neighbor v' of v, we will get a lower or upper bound on the overall sum for v. Hence, we can iteratively update the bounds of each seen node using its neighbors. To tighten the bounds, we only decrease an upper bound or increase a lower bound in any update. Subsequently, the initial bounds from Stage I can be iteratively refined over the neighborhood S_x: in iteration $i+1$ (where $i = 0$ corresponds to the initialization in Stage I), $\forall v \in S_x$,

$$\check{x}^{(i+1)}(q,v) = \max\left\{\begin{array}{l}\check{x}^{(i)}(q,v), \\ \alpha I(q,v) + (1-\alpha)\sum_{v' \in N_x(v)} X_{vv'}\check{x}^{(i)}(q,v')\end{array}\right\} \quad (17)$$

$$\hat{x}^{(i+1)}(q,v) = \min\left\{\begin{array}{l}\hat{x}^{(i)}(q,v), \\ \alpha I(q,v) + (1-\alpha)\sum_{v' \in N_x(v)} X_{vv'}\hat{x}^{(i)}(q,v')\end{array}\right\} \quad (18)$$

where $N_x(v) = \text{In}(v)$, $X_{vv'} = M_{v'v}$ for F-Rank, and $N_x(v) = \text{Out}(v)$, $X_{vv'} = M_{vv'}$ for T-Rank. If a neighbor v' of v is unseen, we simply use a lower bound zero and the unseen upper bound for it.

We terminate the iterative refinement when the bounds converge, which is guaranteed because $\{\check{x}^{(i)}(q,v)\}_{i=0}^{\infty}$ and $\{\hat{x}^{(i)}(q,v)\}_{i=0}^{\infty}$ are bounded monotone sequences due to the $\max\{\cdot\}$ and $\min\{\cdot\}$ functions, respectively.

In general, improving the bounds even for a single node may further improve the bounds for its neighbors, and this effect propagates recursively on the neighborhood. As the neighborhood is often far smaller than the entire graph, it is feasible to iteratively refine the bounds over it.

Next, we discuss the specific realization of the two-stage framework for F-Rank and T-Rank, respectively.

Realization of F-Rank. As F-Rank is equivalent to PPR (see Proposition 1), we can leverage an existing work on PPR for Stage I, namely the Bookmark-Coloring Algorithm (BCA) [19]. BCA iteratively spreads the "residual" starting from the query node over the graph to form the PPR at each node. It maintains two values for each node $v \in V$ w.r.t. a query node q: the current estimated PPR score $\rho(q,v)$ and the residual $\mu(q,v)$. Initially, $\forall v, \rho(q,v) = 0$. Moreover, there is initially a total of one unit residual, all of which is concentrated at q, *i.e.*, $\mu(q,q) = 1$ and $\forall v \neq q, \mu(q,v) = 0$. Subsequently, BCA picks the node v_{max} with the largest residual currently, and performs *BCA processing* on v_{max}:

- α portion of its residual adds to its current estimated PPR, *i.e.*, $\rho(q,v_{\text{max}}) \leftarrow \rho(q,v_{\text{max}}) + \alpha \cdot \mu(q,v_{\text{max}})$;
- the remaining $1-\alpha$ portion spreads to its out-neighbors, *i.e.*, $\forall v' \in \text{Out}(v_{\text{max}}), \mu(q,v') \leftarrow \mu(q,v') + (1-\alpha)\mu(q,v_{\text{max}})M_{vv'}$;
- reset its residual to zero: $\mu(q,v_{\text{max}}) \leftarrow 0$.

This procedure—picking v_{max} and apply BCA processing—is repeated until the total amount of remaining residual becomes zero, which occurs asymptotically. We refer readers to the original work [19] for full details.

Stage I. We concretize the f-neighborhood as the set of nodes currently with non-zero estimated PPR in BCA: $S_f \triangleq \{v \in V : \rho(q,v) > 0\}$. The precondition of BCA dictates $\rho(q,v) = 0, \forall v \in V$, which results in an initially empty S_f.

As the first step in Stage I, to expand S_f, instead of picking one node v_{max} with maximal residual as in the original BCA, we generally pick m nodes by some strategy (which we will discuss next), and apply BCA processing to each of them. After the processing, their $\rho(q,*)$ become non-zero, and thus they are included into S_f.

The question now is how to select the m nodes. *First*, to tighten the bounds quickly, we need to reduce the total residual (as we will see soon in Eq. 19–21). Thus, it is preferred to select a node v with a large residual. *Second*, it is better to select nodes with few out-neighbors, since the BCA processing time of a node is linear in its number of out-neighbors. Factoring in both criteria, we define the *benefit* of a node v as $\mu(q,v)/|\text{Out}(v)|$. Subsequently, we pick up to m nodes with the largest non-zero benefits. Note that the first expansion will only bring in the query node q, since it is the only node with non-zero residual initially.

In the above expansion model, m controls the granularity of the expansion—a small m may result in too frequent bounds updating, while a large m may miss the opportunity to stop early. We use $m = 100$ based on some trial queries. The performance is not sensitive to small changes in m.

As the second step in Stage I, we initialize the bounds $\check{f}^{(0)}(q,v)$ and $\hat{f}^{(0)}(q,v)$ for each seen node $v \in S_f$, as well as the unseen upper bound $\hat{f}^{(0)}(q)$. The initializations are based on the current $\rho(q,*)$ and $\mu(q,*)$ in BCA, as below.

Proposition 4: Given the current $\mu(q,*)$ and $\rho(q,*)$ as maintained by BCA, the following bounds hold:

$$\hat{f}^{(0)}(q) = \frac{\alpha}{2-\alpha}\max_{u\in V}\mu(q,u) + \frac{1-\alpha}{2-\alpha}\sum_{u\in V}\mu(q,u) \quad (19)$$

$$\check{f}^{(0)}(q,v) = \rho(q,v), \forall v \in S_f \quad (20)$$

$$\hat{f}^{(0)}(q,v) = \rho(q,v) + \hat{f}^{(0)}(q), \forall v \in S_f \quad (21)$$

PROOF (SKETCH): It is already known that Eq. 20 holds [19]. Thus, we only need to prove Eq. 19, upon which Eq. 21 can be built. For any node v, consider an upper bound U for the sum of residual that is already at v or may spread to v for the first time. An α portion of U adds to $\rho(q,v)$, while the remaining $(1-\alpha)$ spreads to v's out-neighbors. Then, the same mechanism repeats—residual spread to neighbors may come back to v for a second or third time, adding α portion to $\rho(q,v)$ each time, totaling to $U\cdot[\alpha + (1-\alpha)^2\alpha + (1-\alpha)^4\alpha + \ldots] = U/(2-\alpha)$. Thus, $U/(2-\alpha)$ is the maximum residual that may add to $\rho(q,v)$ for any node v, *i.e.*, an unseen upper bound. It can be simplified to Eq. 19 since $U \leq \mu(q,v) + (1-\alpha)\sum_{u\neq v}\mu(q,u)$. Next, Eq. 21 can be easily obtained by adding a seen node v's current $\rho(q,v)$ to $\hat{f}^{(0)}(q)$. ■

Note that we consider the residual that may repeatedly spread to and from a node, and thus obtain better upper bounds than the work by Gupta *et al.* [16]. The latter only accounts for residual that may spread to a node for the first time.

Stage II. For each seen node, the iterative refinement of its bounds (Eq. 17 and 18) is applied as is.

Realization of T-Rank. Our realization of Stage I and II hinges on the concept of *border nodes* [14], [20]. A border node of the t-neighborhood S_t has at least one of its in-neighbors outside S_t. Thus, to reach q from any unseen node $v \notin S_t$, we must first pass through a border node. Let u be a border node with the largest upper bound. Then, the unseen upper bound, which is the largest probability of reaching q from $v \notin S_t$, can be achieved by reaching u with probability 1 in one step, and then continuing from u to q:

$$\hat{t}(q) = (1-\alpha)\max_{u\in\partial(s_t)}\hat{t}(q,u), \quad (22)$$

where $\partial(s_t)$ is the set of border nodes of S_t. Note that reaching u in one step dampens the probability by a factor $1-\alpha$ due to the geometrically distributed walk length.

Stage I. In the first expansion, let $S_t = \{q\}$ with $\check{t}^{(0)}(q,q) = \alpha$ due to Eq. 8, and $\hat{t}^{(0)}(q,q) = 1$. The unseen upper bound is $\hat{t}^{(0)}(q) = 1-\alpha$ due to Eq. 22.

In subsequent expansions, we aim to reduce the unseen upper bound, which will further improve the bounds of the seen nodes owing to the propagating effect of the iterative refinement in Stage II. We pick up to m border nodes with the largest upper bounds, and bring all of their in-neighbors into S_t. This makes these m nodes no longer border nodes, and thus reduces the unseen upper bound (which is based on the border node with the largest upper bound, see Eq. 22). As we discussed for F-Rank, m controls the granularity of expansion, and it can be set empirically ($m = 5$ in our experiments).

To initialize the bounds for nodes already in S_t before this expansion, we use their bounds from the last expansion. For each newly included node, we use a lower bound of zero, and the unseen upper bound from the last expansion. Finally, we initialize the current unseen upper bound using Eq. 22.

Stage II. We iteratively refine the bounds for each seen node as in Eq. 17–18. In addition, based on Eq. 22 we can also refine the unseen upper bound in each iteration: $\hat{t}^{(i)}(q) = (1-\alpha)\max_{u\in\partial(S_t)}\hat{t}^{(i)}(q,u)$.

B. Distributed Solution for 2SBound

We have so far assumed that the entire graph resides in the main memory of a single machine. To scale 2SBound to very large graphs, we design a solution based on the observation of the *active set* and the technique of *data striping*.

1) Active Set

For any query, 2SBound only needs to maintain a subset of the nodes and edges in G rather than the entire G. As our realization is decoupled into F-Rank and T-Rank, we maintain the nodes in their respective neighborhood, and the set of edges for these nodes. We call the nodes and edges to be maintained the *active nodes* and *active edges* respectively, which collectively form the *active set*.

Memory requirement. The active set is the minimum working set that must reside in the main memory. Otherwise, frequent page swappings occur during iterative refinement. Fortunately, in terms of the absolute space cost, the active set is usually a minute subset of the entire graph. In terms of the scalability, as the graph grows, the active set scales at a slower rate, as our analysis below and the experiments show. Thus, it is practical to fit the active set in the main memory.

Orders of growth. Assume a graph G with $|V|$ nodes and average degree $\bar{D}$. As observed by Leskovec *et al.* [21], the average degree can be modeled by power laws: $\bar{D} \approx c|V|^{a-1}$, where c and a are graph-dependent constants and $1 < a < 2$ on most real-world graphs. Hence, G incurs $O(|V| + |V|\bar{D}) = O(c|V|^a)$ space, where $|V|\bar{D}$ is the number of edges.

Next, we examine the active set. In each neighborhood expansion, we pick m nodes and bring their neighbors into the active set. After n expansions we have $O(nm\bar{D})$ active nodes. Since m is a constant and we assume n is also a constant for a given slack ϵ and graph G, the number of active nodes is $O(\bar{D})$. Hence, the active set incurs $O(\bar{D} + \bar{D}^2) = O(c^2|V|^{2(a-1)})$ space. To compare the orders of growth of the active set and the graph, we find:

$$\lim_{|V|\to\infty}\frac{c^2|V|^{2(a-1)}}{c|V|^a} = \lim_{|V|\to\infty} c|V|^{a-2} = 0,\ \forall a \in (1,2), \quad (23)$$

implying that the active set grows slower than the graph.

2) *Distributed Architecture*

Our architecture includes one *active processor*, which is connected to multiple *graph processors* over a network.

Active processor (AP). Starting with the query node, AP expands the neighborhoods by picking some nodes as in Sect. V-A, whose neighbors are subsequently brought into the active set. But instead of pulling them directly from the graph (which is not in its main memory), AP queries the graph processors over a network, which are responsible for identifying and sending back the new active nodes and edges. Subsequently, AP incrementally assembles the active set from the responses, updates the bounds, and proceeds to the next expansion until the top K nodes can be determined.

Graph processors (GP). When the graph does not fit into the main memory of a single machine, we rely on *data striping* [22], a technique to segment data over multiple storage units. In our case, the graph is segmented across multiple GPs—each GP stores a subset of the nodes and edges in its main memory. In particular, we assign nodes (along with their edges) in the graph to GPs in a round-robin fashion.

Such striping presents several benefits. *First*, it aggregates the fast storage (main memory) of GPs to handle large graphs. *Second*, it enables parallel access to different parts of the graph. *Third*, apart from segmenting the graph which involves minimal work, there is no offline precomputation like the original 2SBound. In general, using a cluster of commodity computers is more flexible, cost effective, and reliable (if with redundancy) than using a single powerful machine.

Upon an expansion request from AP during query processing, each GP identifies the requested active nodes and edges stored in it, and sends them back to AP. AP can then incrementally assemble the active set, as described earlier.

VI. Experiments

We aim to evaluate the effectiveness of RoundTripRank and RoundTripRank+, as well as the efficiency of the top-K algorithm 2SBound, on the two real-world datasets below.

Bibliographic network (BibNet). There are 2 million nodes (papers, authors, terms, venues) and 25 million edges (paper-paper, paper-term, paper-venue, paper-author), extracted from DBLP and Citeseer. The paper-paper citation edges are directed, while the others are undirected. The edge weights are set following a previous work [14].

Query log (QLog). We use a search engine query log [12]. After removing search phrases and clicked URLs that only appear once, we construct a graph with 2 million nodes and 4 million edges—the search phrases and clicked URLs are the nodes, where an undirected edge is drawn between them if the former has a click landing on the latter. The count of such clicks is used as the edge weight.

A. *Effectiveness of RoundTripRank and RoundTripRank+*

We evaluate the ranking of the results to validate our hypotheses. *First*, most ranking scenarios require some trade-off between importance and specificity, as RoundTripRank pursues. *Second*, the optimal trade-off varies from task to task, depending on the user preference or task nature. Hence, an effective proximity measure should cater to different tasks in a flexible manner, as RoundTripRank+ pursues.

Hence, after describing the experimental settings, we first compare RoundTripRank with a few mono-sensed baselines to demonstrate the need for the dual senses. Next, we show that task-oriented customization is indeed beneficial, and compare RoundTripRank+ with various dual-sensed baselines.

Subgraphs. As we evaluate the effectiveness rather than efficiency here, we use the iterative method in Eq. 5 and 8 for the exact ranking to eliminate the effect of approximation. However, some baselines (*e.g.*, SimRank and TCommute) are very expensive to compute exactly on the full graphs. Thus, as previous works [23], [24], we use smaller subgraphs for the effectiveness evaluation. The full graphs will be used in Sect. VI-B for the efficiency study.

For BibNet, we focus on 28 hand-picked major venues in four related areas (DB/DM/IR/AI), resulting in a subgraph of 20545 nodes and 252272 edges. For QLog, we start with 200 random nodes, and expand to their neighbors for three hops, resulting in a subgraph of 23665 nodes and 74504 edges.

Experiment methodology. In our experiments, we reserve some nodes with known association to the query, and then test whether a proximity measure can rank these nodes highly without the knowledge of the association. In other words, such reserved nodes form the ground truth, which we aim to re-discover. This methodology makes the ground truth easily available for large-scale evaluation, which is often used as a benchmark test (*e.g.*, [14]) for graph proximity. If a measure performs well on such ground truths, presumably it can also rank other matching results highly.

Thus, we use the ranking scenarios in Task 1–4 below, which are adapted from Task A–D in Sect. I, such that the ground truth nodes for each query are automatically known. To test the ability to recover the ground truth, we remove all direct edges between the query and ground truth nodes. As these tasks mimic the different trade-offs between importance and specificity (which we will analyze in Sect. VI-A2), the need for a customizable trade-off can be justified.

- *Task 1 (Author) / Task 2 (Venue)*: On BibNet, given a paper as the query node, find all its authors/venue.
- *Task 3 (Relevant URL)*: On QLog, given a search phrase as the query node, find a randomly chosen clicked URL.
- *Task 4 (Equivalent search)*: On QLog, given a search phrase as the query node, find its equivalent phrases. We deem two phrases equivalent if they contain the exact same non-stop words (*e.g.*, “the apple ipod” and “ipod of apple”).

Evaluation. For each task, we randomly sample 1000 nodes as the *test queries*. To assess the ranking for a query, we filter out the query node itself and nodes not of the target type. We then evaluate the filtered ranking against the ground truth using NDCG@K with ungraded judgments. Statistical significance is verified using two-tail paired t-tests.

	Task 1			Task 2			Task 3			Task 4			Average		
	$K=5$	$K=10$	$K=20$	$K=5$	$K=10$	$K=20$	$K=5$	$K=10$	$K=20$	$K=5$	$K=10$	$K=20$	$K=5$	$K=10$	$K=20$
RoundTripRank	**0.3798**	**0.4189**	**0.4534**	**0.7573**	**0.7853**	**0.7974**	**0.3750**	**0.4112**	**0.4357**	**0.4876**	**0.5378**	**0.5763**	**0.4999**	**0.5383**	**0.5657**
F-Rank/PPR	0.3276	0.3649	0.3985	0.7498	0.7835	0.7940	0.3694	0.4079	0.4348	0.3777	0.4312	0.4757	0.4561	0.4969	0.5257
T-Rank	0.3041	0.3425	0.3817	0.6864	0.7306	0.7436	0.1957	0.2312	0.2701	0.4523	0.5093	0.5527	0.4096	0.4534	0.4870
SimRank	0.1981	0.2236	0.2512	0.5747	0.6197	0.6310	0.0988	0.1214	0.1451	0.4365	0.4953	0.5404	0.3270	0.3650	0.3919
AdamicAdar	0.1689	0.1765	0.1831	0.2133	0.2425	0.3028	0.0000	0.0000	0.0006	0.4192	0.4713	0.5185	0.2004	0.2226	0.2512

Fig. 5. NDCG@K of RoundTripRank and mono-sensed baselines. The best in each column is bolded, and the runner-up is underlined.

(a) F-Rank/PPR	(b) T-Rank	(c) RoundTripRank
Comp. Res. Repository	Spatio-Temporal DBs	Spatio-Temporal DBs
SIGMOD	Temporal DBs, Dagstuhl	Temp. Repr. & Reasoning
VLDB	Spatio-Temporal Data Mining	GeoInformatica
ICDE	Temporal Aspects in Info. Sys.	ACM GIS
Temp. Repr. & Reasoning	Ana. & Retr. in Video Streams	VLDB

Fig. 6. Ranking venues for "spatio temporal data."

(a) F-Rank/PPR	(b) T-Rank	(c) RoundTripRank
World Conf. on WWW	Wkshp. on Semantic Web	Intl. Semantic Web Conf.
IEEE Intelligent Sys.	Trends in Web Sci.	Intl. Conf. on Web Services
Intl. Conf. on Web Services	Semantic Grid	IEEE Intelligent Sys.
Commun. ACM	J. Web Engineering	J. Web Semantics
Comp. Res. Repository	Semantic Tech.	Euro. Semantic Web Conf.

Fig. 7. Ranking venues for "semantic web."

1) RoundTripRank and Mono-Sensed Baselines

As the first dimension, we validate that dual-sensed RoundTripRank outperforms mono-sensed baselines.

Quantitative results. We set $\alpha = 0.25$ for RoundTripRank, *i.e.*, $L, L' \sim \text{Geo}(0.25)$. Its ranking is stable for a wide range of α between 0.1 and 0.5. As baselines, we use importance-based F-Rank/PPR and specificity-based T-Rank with the same α. We also use SimRank [8] with $C = 0.85$ (as recommended, which we find robust) and AdamicAdar [7], both of which capture some form of "closeness" as Sect. II explained.

As reported in Fig. 5, RoundTripRank consistently outperforms all the baselines across all four tasks. On average, it improves NDCG@5 over the runner-up (F-Rank/PPR) by 10%, with statistical significance ($p < 0.01$). Hence, some balance between importance and specificity is indeed necessary.

Illustrative results. We illustrate the top 5 results of two queries on the full BibNet graph. Given a query consisting of some term nodes, its matching venues are ranked. It is relatively objective to judge the importance and specificity of venues as compared to other types of nodes.

We show the results of the first query, "spatio temporal data" (three term nodes), in Fig. 6. F-Rank finds important venues in the database area. However, most of them are too general, and will be ranked highly for any data-centric topic. In contrast, T-Rank favors venues specifically tailored to the topic, but they are less well-known. Lastly, RoundTripRank's ranking is comprehensive and balanced, with not only important (*e.g.*, VLDB) or specific venues (*e.g.*, Spatio-Temporal DBs), but also venues that are themselves a balance of the two senses (*e.g.*, ACM GIS). Similar observations can be made on Fig. 7 for the second query "semantic web".

2) RoundTripRank+ and Dual-Sensed Baselines

As the second dimension, we investigate the benefit of a customizable trade-off between importance and specificity. In particular, we study the effect of varying β on RoundTripRank+, and compare it to various dual-sensed baselines.

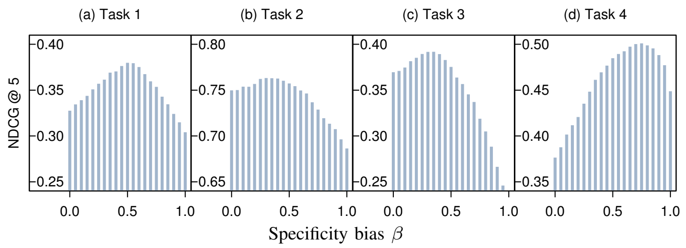

Fig. 8. Effect of the specificity bias.

Effect of specificity bias β. Recall that $\beta \in [0, 1]$ is used to customize the trade-off between importance and specificity in RoundTripRank+. Fig. 8 shows the performance when varying β between 0 and 1. We only present NDCG@5, as the same trends are observed @10 and @20.

We *first* observe that extreme β values (close to 0 or 1) result in poor performance. When $\beta = 0$ (or 1), RoundTripRank+ only captures importance (or specificity). This reconfirms our findings in Sect. VI-A1 that we need both senses.

Second, different tasks have varying optimal values β^*, naturally reflecting the different trade-offs required in the tasks. In Task 1 (Author), $\beta^* \approx 0.5$, as paper authors can be either important (*e.g.*, the faculty) or specific (*e.g.*, the student). In Task 2 (Venue), $\beta^* < 0.5$, as paper submissions tend to favor important venues. In Task 3 (Relevant URL), $\beta^* < 0.5$, as users are often biased to click on important and well-known sites. In Task 4 (Equivalent search), $\beta^* > 0.5$, as equivalent search phrases ideally refer to the exact same concept, and thus are inherently specific to each other. The varying β^* values imply no "one-size-fits-all" solution, and thus a customizable trade-off is beneficial.

For a given task at hand, the optimal β can be tuned using some development queries, or predetermined by identifying the underlying objective as we analyzed above, or directly specified by users to reflect their needs. If such options are not available, we can use the default $\beta = 0.5$, which still fares better than $\beta \to 0$ or 1 as just observed.

Comparing to existing dual-sensed baselines. We set $\alpha = 0.25$ for RoundTripRank+ as before. For each task, to choose the optimal β, we use 1000 randomly sampled *development queries* that do not overlap with the test queries.

We use existing dual-sensed baselines, namely truncated commute time or TCommute [11], [14] with $T = 10$ (as recommended, which we find robust), and ObjSqrtInv [5] with $d = 0.25$ (like α, the ranking is stable for a wide range of d). Additionally, as our computational model is actually a geometric mean of F-Rank and T-Rank, we also compare with their harmonic [12], [13] and arithmetic means. These existing works [5], [11], [12], [13] neither recognize the benefit of, nor implement, customizable trade-offs for different tasks.

	Task 1			Task 2			Task 3			Task 4			Average		
	$K = 5$	$K = 10$	$K = 20$	$K = 5$	$K = 10$	$K = 20$	$K = 5$	$K = 10$	$K = 20$	$K = 5$	$K = 10$	$K = 20$	$K = 5$	$K = 10$	$K = 20$
RoundTripRank+	**0.3798**	**0.4189**	**0.4534**	**0.7630**	**0.7945**	**0.8045**	**0.3856**	**0.4188**	**0.4446**	**0.5035**	**0.5560**	**0.5944**	**0.5080**	**0.5470**	**0.5742**
TCommute	0.3381	0.3800	0.4151	0.7514	0.7860	0.7965	0.3718	0.4104	0.4350	0.4322	0.4873	0.5297	0.4734	0.5159	0.5441
ObjSqrtInv	0.3308	0.3694	0.4070	0.7556	0.7849	0.7963	0.3589	0.3979	0.4236	0.4045	0.4590	0.5016	0.4624	0.5028	0.5321
Harmonic	0.3155	0.3535	0.3926	0.7000	0.7426	0.7546	0.3012	0.3384	0.3683	0.4931	0.5439	0.5833	0.4524	0.4946	0.5247
Arithmetic	0.3364	0.3756	0.4114	0.7499	0.7841	0.7948	0.3693	0.4104	0.4320	0.4211	0.4799	0.5223	0.4692	0.5125	0.5401

Fig. 9. NDCG@K of RoundTripRank+ and existing dual-sensed baselines. The best in each column is bolded, and the runner-up is underlined.

	Task 1	Task 2	Task 3	Task 4	Average
RoundTripRank+	**0.3798**	**0.7630**	**0.3856**	**0.5035**	**0.5080**
TCommute+	0.3614	0.7514	0.3770	0.4603	0.4876
ObjSqrtInv+	0.3228	0.7560	0.3691	0.4478	0.4740
Harmonic+	0.3262	0.7450	0.3602	0.4960	0.4818
Arithmetic+	0.3560	0.7508	0.3728	0.4576	0.4843

Fig. 10. NDCG@5 of RoundTripRank+ and customized dual-sensed baselines. The best in each column is bolded, and the runner-up is underlined.

As reported in Fig. 9, RoundTripRank+ consistently outperforms all the baselines in all four tasks. On average, it improves NDCG@5 by 7% over the runner-up (TCommute) with statistical significance ($p < 0.01$). The results clearly highlight the advantage of flexible trade-offs across tasks.

Comparison to customized dual-sensed baselines. Recall that existing dual-sensed baselines [5], [11], [12], [13] apply a fixed trade-off across tasks. To give them the benefit of a flexible trade-off, we customize each of them with a tunable $\beta \in [0, 1]$, putting weights $1 - \beta$ and β on their two sub-measures, respectively. To choose the optimal β for each task, we use the same development queries of RoundTripRank+. We stress that the customizations are implemented by us, and existing works are unaware of such a need.

Nonetheless, RoundTripRank+ still performs the best consistently, as summarized in Fig. 10 where each customized baseline is marked with "+". For brevity only NDCG@5 is shown, but the conclusion is similar @10 or @20. On average, RoundTripRank+ improves over the runner-up (TCommute) by more than 4%, with statistical significance ($p < 0.01$). In addition, the baselines perform unevenly across tasks—the runner-up varies from task to task, providing no evidence to argue one or another even empirically.

We attribute the better performance of RoundTripRank+ to the coherent integration of importance and specificity in a round trip. In contrast, most baselines combine their sub-measures for the two senses in a somehow crude manner. For example, the arithmetic mean is simply the expectation of two *independent* trials, one for each sense, lacking coherence in their integration. Others such as TCommute, as Sect. II discussed, lack an interpretation in terms of the two senses. In particular, the semantics of the two sub-measures in TCommute has not been thoroughly studied. That being said, the connections to importance and specificity may still exist, but possibly in a weaker form.

B. *Efficiency of 2SBound*

We evaluate the approximate top-K algorithm 2SBound for RoundTripRank on the two full graphs. In particular, we examine its query time and approximation quality on a single machine, and its scalability on our distributed architecture. In all the experiments we use $K = 10$.

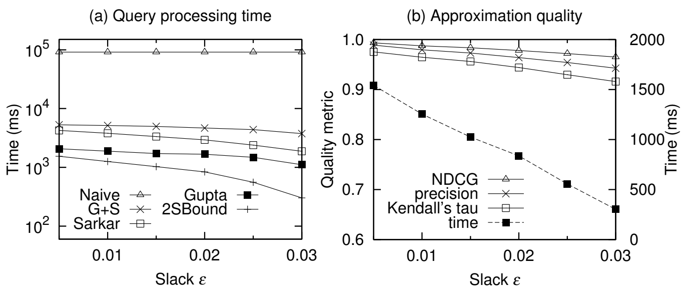

Fig. 11. Time and quality of 2SBound on BibNet under varying slacks.

1) Query Time and Approximation Quality

As assumed in Sect. V-A, we load the entire graph into the main memory of a single computer, which has 8GB capacity. From each graph, we randomly sample 1000 nodes as queries. Due to space limit, we only report the results on BibNet, as we make similar observations on both graphs.

Query time. We compare the query time of 2SBound with that of a naïve baseline and three weaker schemes of 2SBound:

- Naive: The naïve iterative method (Eq. 5 and 8).
- G+S: A weaker scheme of 2SBound. We apply the work by Gupta *et al.* [16] to update the bounds for F-Rank, and the work by Sarkar *et al.* [20] for T-Rank, which are their respective state-of-the-art algorithms.
- Gupta: Same as G+S, but using our two-stage framework for T-Rank, while still using Gupta's method for F-Rank.
- Sarkar: Same as G+S, but using our two-stage framework for F-Rank, while still using Sarkar's method for T-Rank.

We show their average query time in Fig. 11(a). As expected, a large slack ϵ greatly reduces the query time (except Naive which does not use ϵ). Notably, 2SBound is two orders of magnitude faster than Naive, and 2–10 times faster than the others. Its query time is also stable across queries, *e.g.*, its 99% confidence interval is 1255 ± 154ms at $\epsilon = 0.01$.

Approximation quality. To evaluate the approximation quality, we compare 2SBound's ranking with the exact ranking using NDCG, precision and Kendall's tau [15].

We present the approximation quality of 2SBound in Fig. 11(b), along with its query time for reference. While query processing speeds up five-fold as the slack increases, the quality drops slightly. Nevertheless, all metrics are still above 0.9 when the time is only 300ms. Most drops are observed in precision and Kendall's tau, which penalize mistakes equally regardless of their ranks. In contrast, NDCG has a smaller drop, meaning that mistakes are rare at high ranks.

In summary, 2SBound enables real-time query processing with a close approximation to the exact ranking.

(a) BibNet snapshots

Time-stamp	Snapshot size/MB	Active set size/MB	Query time/ms
1994	93	$1.2\pm.1$	728 ± 48
1998	165	$1.5\pm.1$	842 ± 53
2002	284	$1.8\pm.1$	1031 ± 66
2006	510	$2.3\pm.2$	1311 ± 101
2010	692	$2.4\pm.2$	1415 ± 102

(b) QLog snapshots

Time-stamp	Snapshot size/MB	Active set size/MB	Query time/ms
5/6	264	$.041\pm.001$	51 ± 1
5/12	495	$.045\pm.002$	55 ± 2
5/18	713	$.049\pm.002$	57 ± 2
5/24	905	$.053\pm.002$	61 ± 3
5/31	1114	$.054\pm.002$	66 ± 3

Fig. 12. Active set size and query time on growing graphs.

2) *Scalability*

We study the scale-up of the distributed solution for 2SBound. The goal is scaling to larger graphs with minimal increase in the active set and query time.

As both BibNet and QLog grow over time, we model their growth by taking five snapshots at different timestamps—every four years on BibNet from 1994 to 2010, and about every six days on QLog during May 2006. To simulate our AP/GP architecture, for each graph, we assume its i-th snapshot requires i GPs, $\forall i \in \{1, \dots, 5\}$. Note that all snapshots are cumulative, and thus they are larger at later timestamps. On each snapshot, we randomly sample 1000 nodes as queries, and set a slack $\epsilon = 0.01$.

Efficiency. We report the 99% confidence intervals of the active set size and the query time in Fig. 12. Recall that the active set is the minimum working set on AP.

We *first* observe that the active set remains small even on the largest snapshot. For instance, the active set of the 2010 snapshot on BibNet is merely 2.4MB or 0.3% of that snapshot. *Second*, active set size and query time are strongly correlated, since the two-stage bounds updating framework iterates over the active set. *Third*, QLog has larger snapshots but smaller active sets than BibNet. In Sect. V-B, we show that the space cost of the active set, $O(\bar{D}+\bar{D}^2)$, is correlated with the average degree $\bar{D}$, which is smaller on QLog.

Rate of growth. We compare the rate of growth of the three quantities, namely snapshot size, active set size and query time. For each graph, we normalize the three quantities on all snapshots by their corresponding values on the first snapshot. Thus, for each graph, we obtain a rate of growth w.r.t. the first snapshot for each quantity, shown in Fig. 13.

As Sect. V-B discussed, the active set grows much slower than the snapshot. In particular, while the snapshot grows by a factor of 7.4 on BibNet and 4.2 on QLog over the entire period, the active set only grows by a factor of 1.9 and 1.3, respectively. Furthermore, query time depends only on the active set size, and thus has a similar rate of growth. Based on these trends, we can scale query processing to even larger graphs with both space and computational efficiency.

VII. Conclusion

In this paper, we developed RoundTripRank, a graph-based proximity that coherently integrates importance and specificity in a round trip. We further generalized it to RoundTripRank+ using a scheme of hybrid random surfers for a flexible trade-off between the two senses. Empirically, we showed them to be effective on two real-world graphs across various ranking

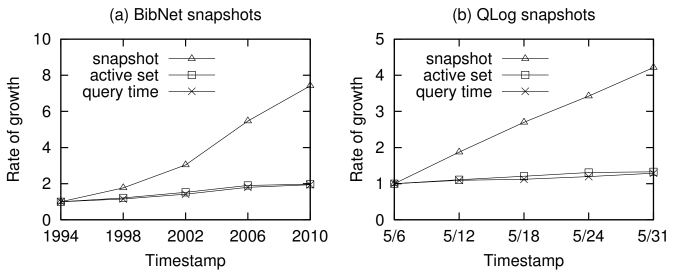

Fig. 13. Rate of growth in snapshot, active set and query time.

tasks. Finally, we proposed a top-K algorithm 2SBound, enabling online processing with a close approximation.

References

[1] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Stanford Univ., Tech. Rep., 1999.

[2] G. Jeh and J. Widom, "Scaling personalized web search," in *WWW*, 2003, pp. 271–279.

[3] A. Balmin, V. Hristidis, and Y. Papakonstantinou, "ObjectRank: Authority-based keyword search in databases," in *VLDB*, 2004, pp. 564–575.

[4] Z. Nie, Y. Zhang, J. Wen, and W. Ma, "Object-level ranking: Bringing order to web objects," in *WWW*, 2005, pp. 567–574.

[5] V. Hristidis, H. Hwang, and Y. Papakonstantinou, "Authority-based keyword search in databases," *ACM TODS*, vol. 33, no. 1, pp. 1–40, 2008.

[6] N. Craswell and M. Szummer, "Random walks on the click graph," in *SIGIR*, 2007, pp. 239–246.

[7] L. Adamic and E. Adar, "Friends and neighbors on the web," *Social networks*, vol. 25, no. 3, pp. 211–230, 2003.

[8] G. Jeh and J. Widom, "SimRank: a measure of structural-context similarity," in *SIGKDD*, 2002, pp. 538–543.

[9] Y. Koren, S. North, and C. Volinsky, "Measuring and extracting proximity in networks," in *SIGKDD*, 2006, pp. 245–255.

[10] H. Tong, C. Faloutsos, and Y. Koren, "Fast direction-aware proximity for graph mining," in *SIGKDD*, 2007, pp. 747–756.

[11] P. Sarkar and A. Moore, "A tractable approach to finding closest truncated-commute-time neighbors in large graphs," in *UAI*, 2007, pp. 335–343.

[12] G. Agarwal, G. Kabra, and K. C.-C. Chang, "Towards rich query interpretation: walking back and forth for mining query templates," in *WWW*, 2010, pp. 1–10.

[13] Y. Fang and K. C.-C. Chang, "Searching patterns for relation extraction over the web: rediscovering the pattern-relation duality," in *WSDM*, 2011, pp. 825–834.

[14] P. Sarkar, A. Moore, and A. Prakash, "Fast incremental proximity search in large graphs," in *ICML*, 2008, pp. 896–903.

[15] S. Chakrabarti, "Dynamic personalized pagerank in entity-relation graphs," in *WWW*, 2007, pp. 571–580.

[16] M. Gupta, A. Pathak, and S. Chakrabarti, "Fast algorithms for top-k personalized pagerank queries," in *WWW*, 2008, pp. 1225–1226.

[17] D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós, "Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments," *Internet Mathematics*, vol. 2, no. 3, pp. 333–358, 2005.

[18] T. Haveliwala, "Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search," *TKDE*, vol. 15, no. 4, pp. 784–796, 2003.

[19] P. Berkhin, "Bookmark-coloring algorithm for personalized pagerank computing," *Internet Mathematics*, vol. 3, no. 1, pp. 41–62, 2006.

[20] P. Sarkar and A. Moore, "Fast nearest-neighbor search in disk-resident graphs," in *SIGKDD*, 2010, pp. 513–522.

[21] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *SIGKDD*, 2005, pp. 177–187.

[22] K. Salem and H. Garcia-Molina, "Disk striping," in *ICDE*, 1986, pp. 336–342.

[23] Y. Sun, J. Han, X. Yan, P. Yu, and T. Wu, "PathSim: Meta path-based top-k similarity search in heterogeneous information networks," in *VLDB*, 2011, pp. 992–1003.

[24] Y. Sun, J. Han, J. Gao, and Y. Yu, "iTopicModel: Information network-integrated topic modeling," in *ICDM*, 2009, pp. 493–502.