RESEARCH-ARTICLE

UPPR+: Scaling Uncertain Personalised PageRank Computation on Billion-Sized Graphs with Mutually Exclusive Edges

MIN ZHANG, University of Warwick, Coventry, West Midlands, U.K.

WEIREN YU, University of Warwick, Coventry, West Midlands, U.K.

Published: 13 July 2025

Citation in BibTeX format

SIGIR '25: The 48th International ACM SIGIR Conference on Research and Development in Information Retrieval
July 13 - 18, 2025
Padua, Italy

Conference Sponsors:
SIGIR

UPPR+: Scaling Uncertain Personalised PageRank Computation on Billion-Sized Graphs with Mutually Exclusive Edges

Min Zhang
University of Warwick
Coventry, UK
min.zhang.2@warwick.ac.uk

Weiren Yu
University of Warwick
Coventry, UK
weiren.yu@warwick.ac.uk

Abstract

While Personalised PageRank (PPR) is widely used for ranking nodes in certain graphs, research on PPR for uncertain graphs remains limited. Real-world graphs often exhibit uncertainty in some edges with interdependent probabilities. The best-of-breed work by Kim *et al.* [13] proposed a fast approximate algorithm, UPPR, leveraging the Sherman-Morrison formula with singular value decomposition. However, UPPR lacks error guarantees, and struggles to scale on large graphs due to the high cost to precompute block matrix inverses over the certain part of the graph.

To address these problems, we propose UPPR+, an efficient scheme to retrieve PPR on billion-scale uncertain graphs. 1) We first propose an efficient approach that groupifies uncertain edges to remove duplicates of source node set and maps uncertainties into a small subspace using low-dimensional embeddings. 2) To further accelerate the computation, we devise a novel uncertain subspace reduction method and advance the aggregation of PPRs over all possible worlds within the subspace via "subset embedding". 3) We provide theoretical guarantees on the exactness of UPPR+ and analyse its time and space complexities. Extensive empirical studies on various real datasets validate that UPPR+ is 23–106x faster than state-of-the-art competitors, and scales well on billion-edge graphs without compromising accuracy.

CCS Concepts

• **Information systems** → **Retrieval models and ranking**.

Keywords

Link Analysis; Uncertain Graphs; PageRank; Ranking Algorithms

ACM Reference Format:
Min Zhang and Weiren Yu. 2025. UPPR+: Scaling Uncertain Personalised PageRank Computation on Billion-Sized Graphs with Mutually Exclusive Edges. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3726302.3730113

1 Introduction

Large-scale interlinked graphs are ubiquitous in our daily lives. A key task in graph data management is to efficiently quantify node importance based on link structure. Personalised PageRank (PPR) is a renowned Google algorithm, with many real applications (*e.g.,* document clustering [1], link recommendation [25]). Given a set of preference seeds, random surfers in PPR either move to one of the out-neighbours with probability α, or teleport to the preference seeds with probability $(1-\alpha)$, where α is *a damping factor*. The steady-state probability distribution of the surfers defines PPR.

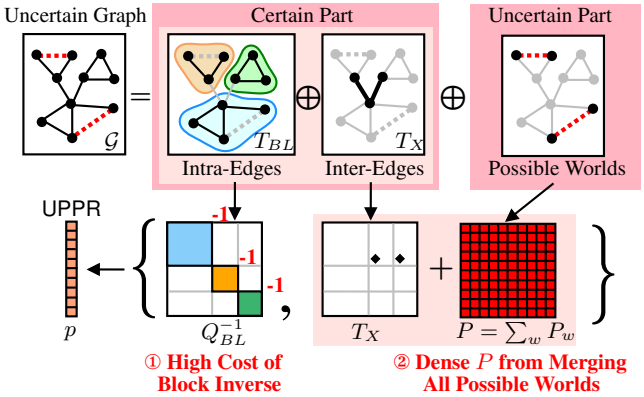


Figure 1: Two Barriers of UPPR [13] for Large-Scale Graphs

Most prior studies on PPR focus on certain graphs [7, 28, 29]. However, real graphs often involve uncertainties, which represents confidence between objects through edge existence probabilities. Existing research on retrieving PPRs for uncertain graphs remains limited. The state-of-the-art work by Kim *et al.* [13] proposed an appealing model for uncertain graphs, which considers dependencies among groups of edges, unlike other probabilistic models [23, 36] that assume independent edge probabilities. This model is useful when we know that an edge exists within specific regions of a graph but are uncertain about which pairs of nodes are connected.

Application (Entity Linking in Uncertain Graphs.) The task of entity linking involves assigning the correct identity to entities (*e.g.,* names, locations) mentioned in the text. Generally, we know that a named entity in a document corresponds to one of several candidate identifiers in a knowledge base, but we are uncertain of the best match. For example, the mention of "apple" could refer to "an edible fruit", "New York City", or "Apple Inc.", depending on the context. The uncertain graph model [13] is useful for representing this ambiguity in named entity disambiguation, where the existence probabilities between a mention (source node) and candidate entities (target nodes) are interdependent. □

Although Kim *et al.* [13] proposed an effective representation of uncertain graphs, it presents significant challenges to efficiently computing uncertain PPR on a large scale due to the growing size of these graphs. The straightforward approach to computing PPR in an uncertain graph $\mathcal{G}$ involves calculating the PPR values for each possible world of $\mathcal{G}$ and then averaging these values. This average is defined as *uncertain PPR* in $\mathcal{G}$.

SIGIR '25, Padua, Italy

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3730113

Motivations. To accelerate uncertain PPR computation, Kim *et al.* [13] devised UPPR (**U**ncertain **PPR**), an approximate algorithm that sacrifices accuracy. Nonetheless, UPPR suffers from two limitations:

Limitation 1 (No Scalability). UPPR is not scalable for large graphs. The largest dataset evaluated in [13] is the Berkeley-Stanford graph, with 650K nodes. Two main barriers hinder UPPR from scaling to billion-sized graphs, as illustrated in Example 1.

Example 1. *Figure 1 depicts the UPPR process for computing uncertain PPR p on an uncertain graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Edges are divided into certain and uncertain edges, with the certain part further split into intra-edges T_{BL} (within k blocks) and inter-edges T_X (across blocks).*

For the certain part, UPPR requires $O(|\mathcal{V}|^3/k^2)$ time[1] to compute Q_{BL}^{-1} and $O(|\mathcal{V}|^2/k)$ memory to store inverted blocks. These costs become prohibitive for billion-scale graphs where $|\mathcal{V}| \gg k$.

For the uncertain part, UPPR merges uncertain edges $\sum_w P_w$ across all possible worlds, requiring $O(\sum_w |P_w|)$ time and space. The exponential growth in possible worlds makes $\sum_w P_w$ dense, even if each P_w is sparse. For example, 8 uncertain edges with an average degree of 4 result in $4^8 = 65,536$ worlds, demanding significant storage. □

Limitation 2 (No Guaranteed Accuracy). UPPR lacks provable error bounds to ensure its accuracy. The total error of UPPR, often non-negligible, arises from three main sources:

- Source 1 stems from UPPR using low-rank SVD to approximate $(T_X + P_w) \approx U_w \Sigma_w V_w$ for each possible world w.
- Source 2 occurs when UPPR approximates $(Q_{BL} - \alpha M_w)^{-1} \approx Q_{BL}^{-1}$, ignoring (αM_w) under the assumption $|T_{BL}| \gg |T_X| + |P_w|$.
- Source 3 is due to UPPR ignoring the term $\sum_w P_w Q_{BL}^{-1} P_w$ (see Eq.(3) in [13]), under the assumption that $|T_X| \gg |P_w|$.

Although [13] did not provide error bounds, we argue that the error bound for the first source, stemming from the low-rank SVD, can be derived from [14]. However, for the second and third sources, the error magnitude depends on two assumptions: $|T_{BL}| \gg |T_X| + |P_w|$ and $|T_X| \gg |P_w|$. These assumptions are not always valid, as the sizes of $|T_{BL}|$, $|T_X|$, and $|P_w|$ depend on the graph structure and how it is partitioned. When $|T_X| \gg |P_w|$ does not hold, [13] suggested a hybrid heuristic that flattens uncertain edges far from seed nodes into certain edges. However, this approach introduces additional errors from edge flattening, with no accuracy guarantees.

Contributions. To address these limitations, we propose scalable techniques for uncertain PPR retrieval on billion-sized graphs.

- We first propose an efficient approach that groupifies uncertian edges to remove duplicates of source node set and maps uncertainties into a small subspace using low-dimensional embeddings. (Sections 4.1–4.2)
- To further accelerate the computation, we devise a novel uncertain subspace reduction method and advance the aggregation of PPRs over all possible worlds within the subspace via "subset embedding" operators. (Sections 4.3–4.4)
- We present a complete UPPR+ algorithm for efficient uncertain PPR computation, analyse its time and space complexities, and provide theoretical guarantees on its exactness. (Section 4.5)
- We conduct extensive experiments on various real datasets to validate that UPPR+ is 23–106x faster than its rivals while scaling well on billion-edge graphs. (Section 5)

[1] $|X|$ denotes the cardinality of a set X, or the number of nonzeros in a matrix X.

2 Related Work

Personalised PageRank (PPR) on certain graphs has been extensively studied (see [31] for a survey). These research can be broadly categorised into power iteration [19, 21], Monte Carlo methods [2–4, 16], and local updates [5, 10, 11, 17, 18, 20, 26–30, 33, 37].

However, there is only a paucity of work on PPR for uncertain graphs [8, 13]. The best-of-breed research by Kim *et al.* [13] proposed a compelling model to represent uncertain graphs, and devised a fast approximate algorithm, UPPR, to compute uncertain PPR. UPPR partitions certain edges into intra-block and inter-block edges, and then applies a low-rank SVD approximation to the combined inter-block edges and uncertain edges. However, UPPR is not scalable for large graphs due to the high computational cost of block-wise inversion over intra-block edges and the need to merge all possible worlds of uncertain edges in the original space ($n \times n$). Moreover, the approximation error in UPPR, arising from multiple sources, is not small in practice. In contrast, our method differs from [13] by mapping uncertainties into a small subspace and proposing subspace reduction techniques to advance aggregating PPRs over all possible worlds within the subspace via "subset embedding".

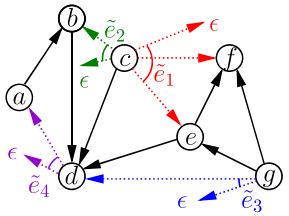
Fushimi *et al.* [8] also proposed an approximation method to evaluate uncertain PPR. To estimate the average PPR (p-ave) over all possible worlds of an uncertain graph, [8] averages the transition matrices of all possible worlds into a single matrix and then applies PPR only once to the merged matrix (s-ave). This approach is similar to the collapse-based strawman baseline used in [13]. The downside is that it produces a dense merged transition matrix, limiting its scalability on sizable graphs. As a result, the largest graph in [8] has only 56K nodes, even smaller than the flattening-based BEAR baseline in [22]. In contrast, our UPPR+ leveraging a subspace reduction method scales well to billion-edge graphs.

There are also heuristic schemes that apply a certain PPR algorithm (*e.g.,* BEAR [22] and B-LIN [24]) to a single merged graph from all possible worlds of an uncertain graph. Two state-of-the-art strawman approaches for merging possible worlds, as suggested in [13], are the collapse-based and flattening-based methods. The collapse-based method averages the transition matrices of all possible worlds into a single matrix before applying a certain PPR algorithm. However, this approach lacks guaranteed accuracy, as detailed in [13]. The flattening-based method encodes uncertainties from all possible worlds into a weighted graph and applies a certain PPR algorithm. This method shows slightly better accuracy than the collapse-based method empirically [22]. However, as shown in our experiments, the aggregation time for collapsing and flattening transition matrices is 1.7–75.4x slower than the UPPR+ runtime, even before accounting for the additional time required by any certain PPR algorithm applied to the merged graph. Thus, existing certain PPR algorithms are inefficient when combined with collapse- and flattening-based methods for large-scale uncertain graphs.

3 Preliminaries

Certain & Uncertain Edges. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an uncertain graph, where $\mathcal{V}$ is the set of nodes, and $\mathcal{E}$ is the set of edges.

- *A certain edge in $\mathcal{E}$*, denoted as $\bar{e} := \langle i, j \rangle$, starts from a well-defined source node i and ends at a well-defined target node j. $\mathrm{src}(\bar{e})$ denotes the source node of $\bar{e}$, and $\mathrm{tar}(\bar{e})$ the target node of $\bar{e}$. The set of all certain edges in $\mathcal{G}$ is denoted as $\bar{\mathcal{E}}$.



(a) $\mathcal{G}$

	a	b	c	d	e	f	g
a	0	0	0	0	0	0	0
b	1	0	0	0	0	0	0
c	0	0	0	0	0	0	0
d	0	1	1	0	1/2	0	0
e	0	0	0	0	0	0	1/2
f	0	0	0	0	1/2	0	1/2
g	0	0	0	0	0	0	0

(b) T_0

Figure 2: (a) uncertain graph $\mathcal{G}$, with solid (*resp.* dotted) arrows representing certain (*resp.* uncertain) edges, and (b) transition matrix T_0 for the certain part of $\mathcal{G}$

- *An uncertain edge in* $\mathcal{E}$, denoted as $\tilde{e}$, starts from a well-defined source node i and ends at one or multiple nodes within a set of potential target nodes. Thus, $\text{src}(\tilde{e}) = \{i\}$, $\text{tar}(\tilde{e}) \subseteq \mathcal{V} \cup \{\epsilon\}$ and $\text{tar}(\tilde{e}) \neq \{\epsilon\}$, where ϵ denotes a non-existent node. The set of all uncertain edges in $\mathcal{G}$ is denoted as $\tilde{\mathcal{E}}$.

Example 2. *Consider the graph $\mathcal{G}$ in Figure 2. Dotted arrows of the same colour originating from a common source represent an uncertain edge. In $\mathcal{G}$, there are 4 uncertain edges, $\tilde{\mathcal{E}} = \{\tilde{e}_1, \tilde{e}_2, \tilde{e}_3, \tilde{e}_4\}$:*

- *$\tilde{e}_1$ (red arrows):* $src(\tilde{e}_1) = \{c\}$, $tar(\tilde{e}_1) = \{e, f, \epsilon\}$.
- *$\tilde{e}_2$ (green arrows):* $src(\tilde{e}_2) = \{c\}$, $tar(\tilde{e}_2) = \{b, \epsilon\}$.
- *$\tilde{e}_3$ (blue arrows):* $src(\tilde{e}_3) = \{g\}$, $tar(\tilde{e}_3) = \{d, \epsilon\}$.
- *$\tilde{e}_4$ (purple arrows):* $src(\tilde{e}_4) = \{d\}$, $tar(\tilde{e}_4) = \{a, \epsilon\}$. □

Possible Worlds of an Uncertain Edge. All possible worlds of an uncertain edge $\tilde{e}$ under mutual exclusion semantics , denoted by $\text{pw}_\text{u}(\tilde{e})$, are defined as

$$\text{pw}_\text{u}(\tilde{e}) = \{\langle i, j\rangle \mid \text{src}(\tilde{e}) = \{i\} \text{ and } \forall j \in \text{tar}(\tilde{e})\}$$

The number of possible worlds in $\text{pw}_\text{u}(\tilde{e})$ is $|\text{pw}_\text{u}(\tilde{e})| = |\text{tar}(\tilde{e})|$. For example, in Figure 2, $\text{pw}_\text{u}(\tilde{e}_1) = \{\langle c, e\rangle, \langle c, f\rangle, \langle c, \epsilon\rangle\}$, where $\langle c, \epsilon\rangle$ implies edge non-existence.

Possible Worlds of a Set of Uncertain Edges. All possible worlds of $\tilde{\mathcal{E}}$ under mutual exclusion semantics, $\text{pw}_\text{u}(\tilde{\mathcal{E}})$, are defined as $\text{pw}_\text{u}(\tilde{\mathcal{E}}) = \bigtimes_{\tilde{e}\in\tilde{\mathcal{E}}} \text{pw}_\text{u}(\tilde{e})$, where $\times$ denotes the Cartesian product.

In Figure 2, there are $|\text{pw}_\text{u}(\tilde{\mathcal{E}})| = 3 \times 2 \times 2 \times 2 = 24$ possible worlds (*e.g.*, $(\langle c, f\rangle, \langle c, b\rangle, \langle g, d\rangle, \langle d, \epsilon\rangle)$).

Possible Worlds of an Uncertain Graph. All possible worlds of an uncertain graph $\mathcal{G}$ semantics are defined as the possible worlds of the entire edge set, $\mathcal{E} = \bar{\mathcal{E}} \cup \tilde{\mathcal{E}}$: $\text{pw}_\text{u}(\mathcal{G}) = \bigtimes_{e\in\mathcal{E}} \text{pw}_\text{u}(e)$.

Uncertain PPR. Given an uncertain graph $\mathcal{G}$ and a seed (preference) vector s, the PPR vector p in $\mathcal{G}$ is defined as

$$p = \underset{w\in\text{pw}_\text{u}(\mathcal{G})}{\text{average}} \text{PPR}\,(T_w, s, \alpha) \tag{1}$$

where T_w is the transition matrix for the possible world w, s is the seed vector, α is the damping factor, and $\text{PPR}\,(T_w, s, \alpha)$ returns the PPR vector p_w for the possible world w *s.t.* $p_w = \alpha T_w p_w + (1-\alpha)s$.

4 Proposed Solution

4.1 Warm-up: Groupifying Uncertainty

Before explaining our methods in detail, we first rewrite all uncertain edges for each possible world w as a source set $\mathcal{I}$ and a target set $\mathcal{J}_w$, which will simplify the presentation of our new approaches.

4.1.1 *Rewriting Uncertainty as Source Set $\mathcal{I}$ and Target Set $\mathcal{J}_w$.*

Consider an uncertain graph $\mathcal{G}$ with l uncertain edges. For each possible world w, the l edges $\tilde{\mathcal{E}}_w := \{\tilde{e}_{1,w}, \cdots, \tilde{e}_{l,w}\}$ can be rewritten as a source set $\mathcal{I}$ and a target set $\mathcal{J}_w$, defined by

$$\mathcal{I} = \{i_1, \cdots, i_l\} \qquad \text{with each } i_k = \text{src}(\tilde{e}_{k,w})$$
$$\mathcal{J}_w = \{j_{1,w}, \cdots, j_{l,w}\} \qquad \text{with each } j_{k,w} = \text{tar}(\tilde{e}_{k,w}) \quad (1 \le \forall k \le l)$$

Example 3. $\tilde{\mathcal{E}}_w = \{\langle c, f\rangle, \langle c, b\rangle, \langle g, d\rangle, \langle d, a\rangle\}$ *under mutual exclusion semantics becomes* $\mathcal{I} = \{c, c, g, d\}$ *and* $\mathcal{J}_w = \{f, b, d, a\}$. □

4.1.2 *Groupifying $\mathcal{I}$ and $\mathcal{J}_w$.*

If there is more than one uncertain edge originating from the same source node, duplicates will exist in $\mathcal{I}$. To address this, we "groupify" the elements in $\mathcal{I}$ and $\mathcal{J}_w$ by removing duplicates in $\mathcal{I}$ and organising related elements in $\mathcal{J}_w$ into nested subsets, thereby simplifying the original structure.

Example 4. *Groupifying* $\mathcal{I} = \{c, c, g, d\}$ *and* $\mathcal{J}_w = \{f, b, d, a\}$ *yields* $\mathcal{I} = \{c, g, d\}$ *and* $\mathcal{J}_w = \{\{f, b\}, d, a\}$. □

Let n $(= |\mathcal{V}|)$ be the number of nodes in $\mathcal{G}$, and l $(= |\mathcal{I}|)$ the number of distinct source nodes in $\mathcal{I}$.[2] Typically, for large-scale graphs, $n \gg l$. We also introduce an uncertain degree vector λ_w to record the cardinality of each k-th $j_{k,w}$ in $\mathcal{J}_w$ as follows:

Uncertain Degree Vector λ_w. Given a target set $\mathcal{J}_w = \{j_{1,w}, \cdots, j_{l,w}\}$, we define an $l\times 1$ uncertain degree vector λ_w, where each k-th entry, $[\lambda_w]_k$, is the number of target nodes in $j_{k,w}$, *i.e.*, $[\lambda_w]_k = |j_{k,w}|$.

Example 5. *For the target set* $\mathcal{J}_w = \{\{f, b\}, d, a\}$, *the uncertain degree vector* $\lambda_w = [2, 1, 1]^\top$. □

4.1.3 *Groupwise Submatrix Extraction.*

For any $n \times n$ matrix X with rows and columns indexed by the node set $\mathcal{V}$, we define an operator of groupwise submatrix extraction:

- $[X]_{*,\mathcal{J}_w}$ denotes the $n \times l$ submatrix of X, where each k-th column is the sum of columns indexed by $j_{k,w}$ in $\mathcal{J}_w$.
- $[X]_{\mathcal{I},\mathcal{J}_w}$ denotes the $l \times l$ submatrix containing all columns of $[X]_{*,\mathcal{J}_w}$ and rows indexed by $\mathcal{I}$.

Example 6. *Let X be a 4×4 matrix, indexed by $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$. Let $\mathcal{J}_w = \{\{v_1, v_3\}, v_2, v_4\}$. Then, $[X]_{*,\mathcal{J}_w}$ is of size 4×3, where the 1st column of $[X]_{*,\mathcal{J}_w}$ is the sum of columns v_1 and v_3 of X:*

$$X = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} \end{array} \Rightarrow [X]_{*,\mathcal{J}_w} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} v_1{+}v_3 & v_2 & v_4 \end{array} \\ \begin{bmatrix} x_{11}+x_{13} & x_{12} & x_{14} \\ x_{21}+x_{23} & x_{22} & x_{24} \\ x_{31}+x_{33} & x_{32} & x_{34} \\ x_{41}+x_{43} & x_{42} & x_{44} \end{bmatrix} \end{array}$$ □

When $\mathcal{J}_w$ has only singletons, a groupwise submatrix extraction operator $[\star]_{\mathcal{I},\mathcal{J}_w}$ reduces to a basic submatrix extraction operator.

4.2 Mapping Uncertainties in Low Dimension

To efficiently compute uncertain PPR, our first insight is that each possible world w is a slight perturbation of the certain part $\mathcal{G}_0$ of $\mathcal{G}$. This perturbation can be captured by using two low-dimensional embedding matrices. Specifically, we first introduce some notations:

Let T_0 be the transition matrix (the transpose of the row-normalised adjacency matrix) for the certain part of $\mathcal{G}$, defined as

$$[T_0]_{u,v} = \begin{cases} 1/d_v, & \text{if a certain edge } \langle v, u\rangle \text{ exists in } \mathcal{G} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

[2] Hereafter, $\mathcal{I}$ and $\mathcal{J}_w$ denote the groupified versions by default.

where d_v is the number of certain edges directed out of v in $\mathcal{G}$. Let D be an $n \times n$ diagonal matrix, where each diagonal entry $[D]_{v,v} = d_v$. Let T_w be the transition matrix for the possible world w of $\mathcal{G}$. Let Λ_w $(= diag(\lambda_w))$ be an $l \times l$ diagonal matrix with the uncertain degree vector λ_w as its diagonal. Let E_n be an $n \times n$ identity matrix.

We now formulate $(T_w - T_0)$ using low-dimensional embeddings.

Theorem 1. *For any possible world w, the transition matrix T_w under mutual exclusion semantics can be represented as*

$$T_w = T_0 + U_w \cdot V_w \tag{3}$$

where the source embedding matrix U_w $(n \times l)$ is

$$U_w = [E_n]_{*,\mathcal{J}_w} - [T_0]_{*,\mathcal{I}} \Lambda_w \tag{4}$$

and the target embedding matrix V $(l \times n)$ is

$$V_w = ([D]_{\mathcal{I},\mathcal{I}} + \Lambda_w)^{-1} [E_n]_{\mathcal{I},*} \tag{5}$$

Proof. Please refer to Appendix A.1 for a detailed proof. □

Example 7. *Recall $\mathcal{G}$ and T_0 in Figure 2. For the possible world, $\tilde{\mathcal{E}}_w = \{\langle c, f\rangle, \langle c, b\rangle, \langle g, d\rangle, \langle d, a\rangle\}$, after groupifying, $\mathcal{I} = \{c, g, d\}$ and $\mathcal{J}_w = \{\{f, b\}, d, a\}$, $\Lambda_w = diag([2, 1, 1]^\top)$. Its transition matrix, T_w, can be represented as $T_w = T_0 + U_w V_w$, where U_w (7×3) and V_w (3×7) are obtained by Eqs.(4) and (5), respectively, as follows:*

$$U_w = \overbrace{\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \end{array}\!\!\begin{array}{c} \overset{f+b}{} \quad d \quad a \\ \left[\begin{array}{ccc} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right] \end{array}}^{[E_7]_{*,\mathcal{J}_w}} - \overbrace{\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \end{array}\!\!\begin{array}{c} \quad c \quad g \quad d \\ \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 0 \end{array}\right] \end{array}}^{[T_0]_{*,\mathcal{I}}} \overbrace{\begin{array}{c} \\ f+b \\ d \\ a \end{array}\!\!\begin{array}{c} \overset{f+b}{} \quad d \quad a \\ \left[\begin{array}{ccc} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \end{array}}^{\Lambda_w} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & -1/2 & 0 \\ 1 & -1/2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$V_w = \left(\overbrace{\begin{array}{c} \\ c \\ g \\ d \end{array}\!\!\begin{array}{c} \quad c \quad g \quad d \\ \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{array}\right] \end{array}}^{[D]_{\mathcal{I},\mathcal{I}}} + \overbrace{\begin{array}{c} \\ f+b \\ d \\ a \end{array}\!\!\begin{array}{c} \overset{f+b}{} \quad d \quad a \\ \left[\begin{array}{ccc} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \end{array}}^{\Lambda_w} \right)^{-1} \overbrace{\begin{array}{c} \\ c \\ g \\ d \end{array}\!\!\begin{array}{c} a \; b \; c \; d \; e \; f \; g \\ \left[\begin{array}{ccccccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right] \end{array}}^{[E_7]_{\mathcal{I},*}} = \begin{bmatrix} 0 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

□

Theorem 1 provides a fast method to get low-dimensional embeddings of $(T_w - T_0)$. Computing U_w via Eq.(4) takes only $O(ld)$ time, where d is the average degree of the certain part of $\mathcal{G}$, while the target embedding matrix V_w via Eq.(5) requires $O(l)$ time.

4.3 Uncertain Subspace Reduction

Based on the low-embeddings of $(T_w - T_0)$ in Theorem 1, we next propose an uncertain subspace reduction method to accelerate large-scale uncertain PPR computation.

Main Idea. Using the low-dimensional matrix V_w from Theorem 1, we begin by projecting the PPR computations for each possible world $(n \times n)$ onto a reduced uncertain subspace $(l \times l)$. In this small subspace, we develop efficient methods to compress all PPR values *w.r.t.* each certain-centred possible world $(w \ominus \mathcal{G}_0)$ into a low-dimensional vector. Finally, we introduce a "subset embedding operator" to map the low-dimensional results $(l \times 1)$ back into a high-dimensional vector $(n \times 1)$, which yields the uncertain PPR.

Subset Embedding Operator. We first introduce the notion of a "subset embedding operator", which embeds the elements of a low-dimensional vector y $(l \times 1)$ into specific positions within a high-dimensional vector x $(n \times 1)$ based on a subset of indices $\mathcal{S}$.

Definition 1. *Given a (short) vector y of length l $(\ll n)$, and a set $\mathcal{S} = \{s_1, \ldots, s_l\}$, where each s_k can represent a singleton node or multiple nodes in $\mathcal{V}$, we define a subset embedding operator: $x \overset{\mathcal{S}}{\longleftarrow} y$, which produces a (long) vector x of length n $(= |\mathcal{V}|)$, with entries indexed by nodes in $\mathcal{V}$. Each v-th element of x is defined as*

$$[x]_v = \begin{cases} \sum_{k \in \mathcal{K}_v} [y]_k & \text{if } v \in \bigcup_{k=1}^{l} s_k \\ 0 & \text{otherwise} \end{cases}$$

where

$$\mathcal{K}_v = \{k \mid v \in s_k,\ s_k \in \mathcal{S},\ 1 \le k \le l\}. \quad \square$$

Intuitively, the set $\mathcal{K}_v$ includes all indices k such that the subset s_k (an element of $\mathcal{S}$) contains the node v. In the special case where all elements of $\mathcal{S}$ are singleton nodes, x becomes a vector of 0s except for the l elements indexed by $\mathcal{S}$, *i.e.,* $[x]_{s_k} = [y]_k$ $(\forall k = 1, \cdots, l)$.

Example 8. *Let $\mathcal{V} = \{a, b, c, d, e, f, g\}$, $\mathcal{I} = \{a, d, f\}$, and $\mathcal{J}_w = \{\{a, b\}, e, \{g, b\}\}$. Given a 3×1 vector $y = [3, 1, 2]^\top$, we have:*

- $x \overset{\mathcal{I}}{\longleftarrow} y$ *embeds y to a 8×1 vector* $x = [\overset{(a)}{3}, 0, 0, \overset{(d)}{1}, 0, \overset{(f)}{2}, 0]^\top$.
- $x \overset{\mathcal{J}_w}{\longleftarrow} y$ *embeds y to a 8×1 vector* $x = [\overset{(a)}{3}, \overset{(b)}{3+2}, 0, 0, \overset{(e)}{1}, 0, \overset{(g)}{2}]^\top$.

Characterising $(p_w - p_0)$ from $(T_w - T_0)$. The subset embedding operator is introduced to accelerate uncertain PPR computation. To achieve this, we first study the relationship between the PPR p_w for any possible world w and the PPR p_0 for the certain part $\mathcal{G}_0$.

Specifically, let T_w (*resp.* T_0) be the transition matrix of w (*resp.* $\mathcal{G}_0$), and let s be a seed vector. We define

$$p_w = \text{PPR}\ (T_w, s, \alpha) \qquad p_0 = \text{PPR}\ (T_0, s, \alpha) \tag{6}$$

Below, we establish the relationship between p_w and p_0.

Theorem 2 (PPR over A Single Possible World). *Given an uncertain graph $\mathcal{G}$, with the PPR vectors p_w and p_0 defined by Eq.(6). Let R be a $n \times n$ matrix, and h_w be a $l \times 1$ vector, defined as*

$$R = (E_n - \alpha T_0)^{-1} \tag{7}$$

$$h_w = ([D]_{\mathcal{I},\mathcal{I}} - \alpha [R]_{\mathcal{I},\mathcal{J}_w} + [R]_{\mathcal{I},\mathcal{I}} \odot (\mathbb{1}\lambda_w^\top))^{-1} [p_0]_{\mathcal{I}} \tag{8}$$

where $\odot$ denotes entry-wise multiplication (i.e., $[XY]_{i,j} = [X]_{i,j}[Y]_{i,j}$), $\mathbb{1}$ is a $l \times 1$ vector of all 1s, and λ_w is a $l \times 1$ uncertain degree vector.

The relationship between p_w and p_0 is as follows:

$$p_w = p_0 + \xi_w + \tfrac{1}{1-\alpha}\, PPR\ (T_0, \eta_w - \xi_w, \alpha) \tag{9}$$

where ξ_w and η_w are obtained from the subset embedding operators:

$$\xi_w \overset{\mathcal{I}}{\longleftarrow} (\lambda_w \odot h_w) \quad \text{and} \quad \eta_w \overset{\mathcal{J}_w}{\longleftarrow} (\alpha h_w) \tag{10}$$

Proof. Please refer to Appendix A.2 for a detailed proof. □

Example 9. *Recall $\mathcal{G}$ in Figure 2. We compute uncertain PPR p_w over a single possible world $\tilde{\mathcal{E}}_w = \{\langle c, \{f, b\}\rangle, \langle g, d\rangle, \langle d, a\rangle\}$.*

First, we evaluate the PPR vector p_0 w.r.t. the certain part of $\mathcal{G}$:

$$p_0 = [0, 0.067, \overset{(c)}{0}, \overset{(d)}{0.091}, 0.093, 0.064, \overset{(g)}{0.067}]^\top$$

Since the source node set $\mathcal{I} = \{c, g, d\}$, it follows that

$$[p_0]_{\mathcal{I}} = \begin{array}{c} c \\ g \\ d \end{array}\!\!\begin{bmatrix} 0 \\ 0.067 \\ 0.091 \end{bmatrix} \quad [D]_{\mathcal{I},\mathcal{I}} = \begin{array}{c} \\ c \\ g \\ d \end{array}\!\!\begin{array}{c} \quad c \quad g \quad d \\ \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{array}\right] \end{array} \quad [R]_{\mathcal{I},\mathcal{I}} = \begin{array}{c} \\ c \\ g \\ d \end{array}\!\!\begin{array}{c} \quad c \quad g \quad d \\ \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.8 & 0.16 & 1 \end{array}\right] \end{array}$$

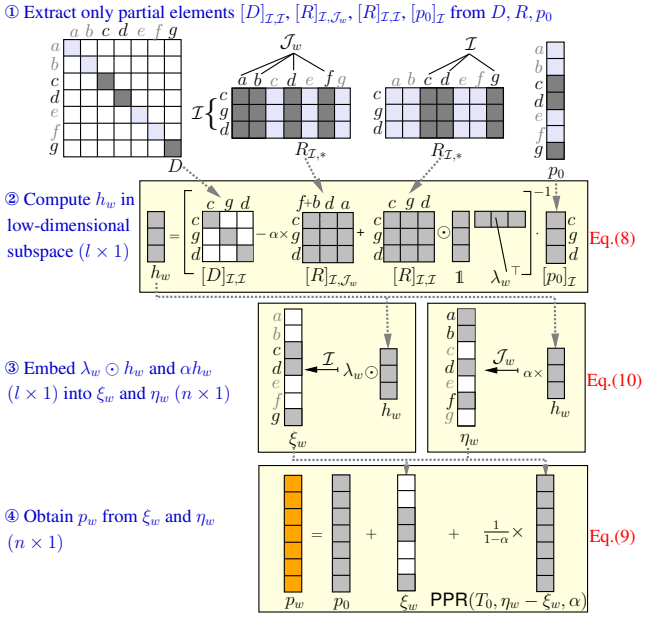


Figure 3: Key Intuition of Theorem 2: Speeding up PPR p_w Computation over any Single Possible World w

Next, we compute λ_w and $[R]_{\mathcal{I},\mathcal{J}_w}$, and obtain h_w via Eq.(8)*:*

$$\lambda_w = \begin{matrix} c \\ g \\ d \end{matrix}\begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \quad [R]_{\mathcal{I},\mathcal{J}_w} = \begin{matrix} \\ c \\ g \\ d \end{matrix}\begin{matrix} \overset{f+b}{} & \overset{d}{} & \overset{a}{} \\ \end{matrix}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.8 & 1 & 0.64 \end{bmatrix} \quad h_w = \begin{matrix} c \\ g \\ d \end{matrix}\begin{bmatrix} 0 \\ 0.022 \\ 0.215 \end{bmatrix} \tag{11}$$

Then, ξ_w and η_w are derived from $\lambda_w \odot h_w$ and αh_w via Eq.(10)*:*

$$\xi_w = [0, 0, \overset{(c)}{0}, \overset{(d)}{0.215}, 0, 0, \overset{(g)}{0.022}]^\top \longleftrightarrow (\lambda_w \odot h_w),$$

$$\eta_w = [\overset{(a)}{0.172}, \overset{(b)}{0}, 0, \overset{(d)}{0}, 0, \overset{(f)}{0.018}, 0]^\top \longleftrightarrow (\alpha h_w),$$

Finally, using ξ_w and η_w, we obtain the PPR p_w from Eq.(9)*:*

$$p_w = [0.172, 0.204, 0, 0.216, 0.084, 0.051, 0.067]^\top \quad \square$$

Intuitively, the key idea of Theorem 2 is pictorially illustrated in Figure 3, where a PPR p_w over a single possible world w is characterised using our uncertain subspace reduction method. It is worth noting that Eq.(8) is performed in a low-dimensional subspace $(l \times l)$ with $l \ll n$, significantly speeding up the computation of p_w. Eq.(8) resorts to two small $l \times l$ grids of R (*i.e.,* $[R]_{\mathcal{I},\mathcal{J}_w}$ and $[R]_{\mathcal{I},\mathcal{I}}$) to obtain a small auxiliary vector h_w $(l \times 1)$. Then, using our subset embedding operators in Eq.(10), two short vectors $(\lambda_w \odot h_w)$ and (αh_w) $(l \times 1)$ are embedded into long vectors ξ_w and η_w $(n \times 1)$, respectively. Finally, p_w is obtained from ξ_w and η_w via Eq.(9).

However, directly applying Theorem 2 over all possible worlds of $\mathcal{G}$ to get uncertain PPR is inefficient due to the following challenge:

Challenge. If we apply Theorem 2 repeatedly to compute each PPR p_w for every possible world w before averaging all PPRs $\{p_w\}$ $(n \times 1)$ to derive the uncertain PPR p, then a high-dimensional space $(n \times 1)$ would be required for the aggregation. Performing this PPR averaging in high-dimensional space across a huge number of possible worlds would be computationally expensive. Thus, the key challenge is how to avoid this high-dimensional space while efficiently aggregating the PPRs $\{p_w\}$ for all possible worlds within the small subspace $(l \times 1)$.

4.4 Advancing PPR Aggregation in Small Space

To address the above challenge, instead of computing the high-dimensional PPR vectors $\{p_w\}$ for all possible worlds $(n \times 1)$ as per Theorem 2 and then averaging them for aggregation, we first aggregate the reduced versions of $\{p_w\}$ into a compact vector in a low-dimensional subspace $(l \times 1)$. Then, this reduced vector is embedded back into the n-dimensional space. The key to advancing PPR aggregation in the reduced subspace is based on two key observations:

Observation 1. *The subset embedding operator in Definition 1 is linearly decomposable. Specifically, let y_1 and y_2 be two short vectors of length l, which are embedded into two long vectors x_1 and x_2 of length n $(\gg l)$, respectively, over the same subset $\mathcal{S}$:*

$$x_1 \overset{\mathcal{S}}{\longleftarrow} y_1 \qquad x_2 \overset{\mathcal{S}}{\longleftarrow} y_2$$

If a linear combination of y_1 and y_2, $(\lambda_1 y_1 + \lambda_2 y_2)$, is embedded into a length-n vector x over $\mathcal{S}$:

$$x \overset{\mathcal{S}}{\longleftarrow} (\lambda_1 y_1 + \lambda_2 y_2)$$

where λ_1 and λ_2 are arbitrary scalars, then it follows that

$$x = \lambda_1 x_1 + \lambda_2 x_2 \quad \square$$

Observation 1 implies the linear decomposability of the subset embedding operator, which allows us to aggregate all reduced versions of $\{p_w\}$ within a compact subspace $(l \times 1)$ before embedding the aggregated result back into the original space $(n \times 1)$.

Observation 2. *The PPR operator is linearly decomposable w.r.t. seed vectors. Specifically, let s_1 and s_2 be two seed vectors, T a transition matrix, and α a damping factor. If*

$$p_1 = PPR\,(T, s_1, \alpha) \qquad p_2 = PPR\,(T, s_2, \alpha)$$

then, for any two arbitrary scalars λ_1 and λ_2, we have

$$\lambda_1 p_1 + \lambda_2 p_2 = PPR\,(T, \lambda_1 s_1 + \lambda_2 s_2, \alpha) \quad \square$$

Observation 2 demonstrates the linear decomposability of the PPR operator *w.r.t.* the seed vector, which allows for the elimination of many redundant computations in uncertain PPR retrieval. From Eq.(9) in Theorem 2, we observe that the PPR vector p_w $(= \mathrm{PPR}(T_w, s, \alpha))$ for each possible world w, given the same seed vector s, can be expressed as the sum of a fixed vector and a scaled PPR vector $\mathrm{PPR}(T_0, s_w, \alpha)$ for the certain graph $\mathcal{G}_0$ *w.r.t.* varying seed vector s_w $(= \eta_w - \xi_w)$. To efficiently compute the uncertain PPR $p = \mathrm{average}_w\{p_w\}$, aggregating $\mathrm{PPR}(T_w, s, \alpha)$ over all possible worlds *w.r.t.* the same seed vector s is equivalent to aggregating $\mathrm{PPR}(T_0, s_w, \alpha)$ for the same certain graph $\mathcal{G}_0$ over the different seed vectors $\{s_w\}$. Using Observation 2, many matrix-vector multiplications $\{T_0 s_w\}$ required for all $\{\mathrm{PPR}(T_0, s_w, \alpha)\}$ can be merged into a single matrix-vector multiplication $(T_0 s)$ (with $s = \sum_w s_w$). This dramatically enhances computational efficiency.

Combining Observations 1–2 and Theorem 2, we next propose our method to efficiently compute uncertain PPR vector p.

Theorem 3 (Uncertain PPR Computation). *Given an uncertain graph $\mathcal{G}$ and a damping factor α, let p_0 be the PPR vector w.r.t. the certain part T_0 of $\mathcal{G}$. For each possible world w of $\mathcal{G}$, h_w is defined by Eq.*(8)*, and λ_w is an uncertain degree vector. The uncertain PPR vector p on $\mathcal{G}$ can be represented as*

$$p = p_0 + \xi + \tfrac{1}{1-\alpha} PPR\,(T_0, \zeta, \alpha) \tag{12}$$

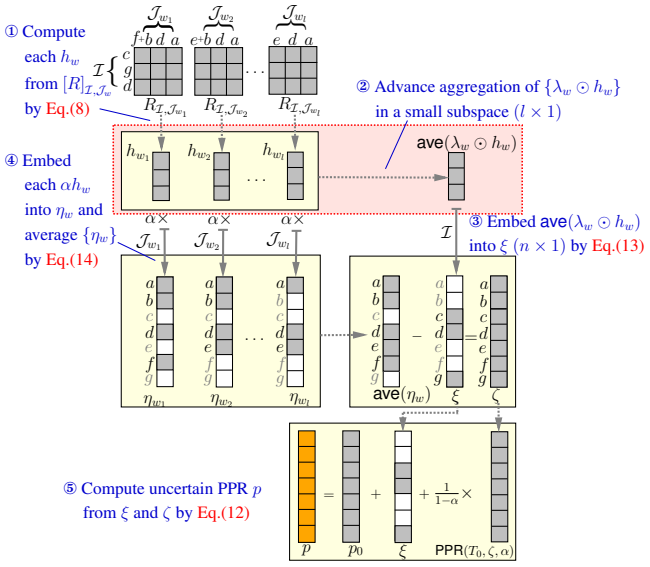


Figure 4: Key Idea of Theorem 3: Advancing Aggregation of PPRs for All Possible Worlds to Get Uncertain PPR p

where ξ is obtained through the subset embedding operation:

$$\xi \xleftarrow{\mathcal{I}} \underset{w \in pw_u(\mathcal{G})}{\text{average}} (\lambda_w \odot h_w) \tag{13}$$

and ζ is derived as follows:

$$\zeta = \underset{w \in pw_u(\mathcal{G})}{\text{average}} (\eta_w) - \xi \quad \textit{with} \quad \eta_w \xleftarrow{\mathcal{J}_w} (\alpha h_w) \tag{14}$$

Proof. Please refer to Appendix A.3 for a detailed proof. □

Theorem 3 presents our efficient model for computing the uncertain PPR p, which addresses the challenge. As visualised in Figure 4, the computation of p via Eq.(12) involves two key components:

(1) To compute ξ, we propose Eq.(13) to achieve a substantial speedup, based on Observation 1. The vectors $\{h_w\}$ from all possible worlds are aggregated (weighted averaged) within a low-dimensional subspace $(l \times 1)$, and the result is then embedded into the high-dimensional vector ξ $(n \times 1)$.
(2) To compute the seed vector ζ for PPR (T_0, ζ, α), we introduce Eq.(14), based on Observation 2. The vectors $\{\eta_w\}$ from all possible worlds, along with ξ, are merged into the seed vector ζ before applying the PPR operator.

Example 10. *Recall $\mathcal{G}$ in Figure 2. We compute uncertain PPR p. After computing each λ_w and h_w as in Example 9, we embed the average of $\lambda_w \odot h_w$ into ξ via Eq.*(13)*, and derive ζ from ξ via Eq.*(14)*:*

$$\xi = [0, 0, 0, 0.215, 0, 0, 0.022]^\top, \quad \zeta = [0.172, 0, 0, -0.197, 0, 0, -0.022]^\top$$

Next, using ζ as a new seed vector, we obtain:

$$PPR\ (T_0, \zeta, \alpha) = [0.034, 0.028, 0, -0.018, -0.002, -0.003, -0.004]^\top$$

Finally, using ξ and ζ, we compute the uncertain PPR p via Eq.(12)*:*

$$p = [0.080, 0.131, 0, 0.149, 0.089, 0.058, 0.067]^\top \quad \square$$

4.5 Putting Them All Together

We now present a complete scheme, UPPR+, in Algorithm 1.

Algorithm 1: UPPR+ $(\mathcal{G}, \alpha, s)$

Input : $\mathcal{G}$: uncertain graph, α: damping factor, s: seed vector
Output: Uncertain PPR p over graph $\mathcal{G}$ *w.r.t.* seed vector s

▶ Phase 1: Precomputation
1 Rewrite and groupify all uncertain edges in $\mathcal{G}$ into source set $\mathcal{I}$, target set $\{\mathcal{J}_w\}$, and uncertain degree vector $\{\lambda_w\}$
2 $\mathcal{J}' := \mathcal{I} \cup (\bigcup_w \mathcal{J}_w)$
3 $T_0 :=$ transition matrix for the certain part of $\mathcal{G}$
4 $[D]_{i,i} :=$ out-degree of node i for the certain part of $\mathcal{G}$ $(\forall i \in \mathcal{I})$
5 Decompose $(E_n - \alpha T_0)$ into L and U via sparse LU factorisation
6 Compute $[R]_{\mathcal{I},\mathcal{J}'}$ from $L, U, \mathcal{I}, \mathcal{J}'$
7 Compute certain PPR $p_0 :=$ PPR (T_0, s, α) by Eq.(6)
▶ Phase 2: Advance PPR Aggregation in Small Subspace
8 Initialise $\eta := 0$ and $h := 0$
9 **foreach** *possible world* w *in* $\mathcal{G}$ **do**
10 $\quad h_w := ([D]_{\mathcal{I},\mathcal{I}} - \alpha[R]_{\mathcal{I},\mathcal{J}_w} + [R]_{\mathcal{I},\mathcal{I}} \odot (\mathbb{1}\lambda_w^\top))^{-1}[p_0]_{\mathcal{I}}$
11 $\quad$ Get $\eta_w \xleftarrow{\mathcal{J}_w} (\alpha h_w)$ via subset embedding
12 $\quad$ Update $h := h + \lambda_w \odot h_w, \quad \eta := \eta + \eta_w$
▶ Phase 3: Revert to Original Space via Subset Embedding
13 Get $\xi \xleftarrow{\mathcal{I}} \left(\frac{h}{|\text{pw}_u(\mathcal{G})|}\right)$ via subset embedding
14 Compute $\zeta := \frac{\eta}{|\text{pw}_u(\mathcal{G})|} - \xi$
15 **return** $p := p_0 + \xi + \frac{1}{1-\alpha}$PPR (T_0, ζ, α)

Theorem 4. *UPPR+ requires $O(|\mathcal{E}|+|L|+|U|+|pw_u(\mathcal{G})|(|\mathcal{V}|+l^2))$ time and $O(|\mathcal{E}|+|\mathcal{V}|(|\mathcal{J}'|+l)+|L|+|U|)$ memory in total, where $|\mathcal{V}|$ is the number of nodes in $\mathcal{G}$, $|\mathcal{E}|$ $(= |\bar{\mathcal{E}}|+|\tilde{\mathcal{E}}|)$ is the number of certain and uncertain edges, l is the number of source nodes for uncertain edges, $|\mathcal{J}'|$ is the number of distinct target nodes for uncertain edges, $|L|$ and $|U|$ are the number of nonzeros in the LU-decomposed matrices L and U, and $|pw_u(\mathcal{G})|$ is the number of possible worlds in $\mathcal{G}$.*

Proof. A line-by-line analysis of Algorithm 1 is provided below.

Line	Description	Time	Space
1-2	get $\mathcal{I}, \{\mathcal{J}_w\}, \{\lambda_w\}, \mathcal{J}'$	$O(\lvert\tilde{\mathcal{E}}\rvert)$	$O(\lvert\tilde{\mathcal{E}}\rvert)$
3-4	initialise T_0, $[D]_{\mathcal{I},\mathcal{I}}$	$O(\lvert\bar{\mathcal{E}}\rvert + l)$	$O(\lvert\bar{\mathcal{E}}\rvert + l)$
5	sparse LU factorisation	$O(\lvert L\rvert + \lvert U\rvert)$	$O(\lvert L\rvert + \lvert U\rvert)$
6	partially inverse $[R]_{\mathcal{I},\mathcal{J}'}$	$O(\lvert L\rvert + \lvert U\rvert + \lvert[R]_{\mathcal{I},\mathcal{J}'}\rvert l)$	$O(\lvert\mathcal{V}\rvert(\lvert\mathcal{J}'\rvert + l))$
7	compute certain PPR	$O(\lvert L\rvert + \lvert U\rvert)$	$O(\lvert\mathcal{V}\rvert + \lvert L\rvert + \lvert U\rvert)$
8	initialise η, h	$O(\lvert\mathcal{V}\rvert + l)$	$O(\lvert\mathcal{V}\rvert + l)$
10	compute h_w	$O(\lvert\text{pw}_u(\mathcal{G})\rvert l^2)$	$O(l^2)$
11	hash η_w	$O(\lvert\text{pw}_u(\mathcal{G})\rvert l)$	$O(\lvert\mathcal{V}\rvert)$
12	update h, η	$O(\lvert\text{pw}_u(\mathcal{G})\rvert l)$	$O(\lvert\mathcal{V}\rvert + l)$
13-14	compute ξ, ζ	$O(\lvert\mathcal{V}\rvert + l)$	$O(\lvert\mathcal{V}\rvert)$
15	get uncertain PPR p	$O(\lvert\mathcal{V}\rvert + \lvert L\rvert + \lvert U\rvert)$	$O(\lvert\mathcal{V}\rvert + \lvert L\rvert + \lvert U\rvert)$

Since real-world graphs are often sparse and large-scale, we have $O(|\bar{\mathcal{E}}|) = O(|L| + |U|)$, $|[R]_{\mathcal{I},\mathcal{J}'}| \leq |\mathcal{I}| \times |\mathcal{J}'|$, and $|\mathcal{V}| \gg l$. Thus, the total complexities of UPPR+ are bounded by $O(|\mathcal{E}| + |L| + |U| + |\text{pw}_u(\mathcal{G})|l^2)$ time and $O(|\mathcal{E}|+|\mathcal{V}|(|\mathcal{J}'|+l)+|L|+|U|)$ memory. □

5 Experiments

5.1 Experimental Settings

Datasets. We adopt the following real-world datasets[3]:

[3] FB, WV, HP, WS, PA are taken from SNAP [15]; UK, AR, WB, IT, SK from LAW [6].

Datasets	Abbr	$n = \|\mathcal{V}\|$	$m = \|\mathcal{E}\|$	m/n	Scale
ego-Facebook	FB	4,039	88,234	21.85	small
wiki-Vote	WV	8,297	103,689	12.45	small
cit-HepPh	HP	27,770	352,807	12.70	medium
web-Stanford	WS	281,903	2,312,497	8.20	medium
cit-Patents	PA	3,774,768	16,518,948	4.38	large
uk-2002	UK	18,520,486	298,113,762	16.10	large
arabic-2005	AR	22,744,080	639,999,458	28.14	large
webbase-2001	WB	118,142,155	1,019,903,190	8.63	large
it-2004	IT	41,291,594	1,150,725,436	27.87	large
sk-2005	SK	50,636,154	1,949,412,601	38.50	large

Compared Algorithms. We compare the following algorithms:

- UPPR+: our proposed approach in Algorithm 1.
- UPPR [13]: the state-of-the-art algorithm to estimate uncertain PPR, using graph partitioning and Sherman-Morrison methods.
- f-BEAR [22]: an efficient Random Walk with Restart algorithm over a flattened transition matrix to evaluate uncertain PPR.
- exh: an exhaustive method that computes PPR over each possible world, and averages results to determine uncertain PPR.
- exhApx: approximate exh by using B_LIN [24] to get PPR for each possible world, and average results for uncertain PPR.
- flatApx: a fast approximate algorithm that flattens uncertain edges to a weighted certain graph, then applies B_LIN.
- collApx: a fast algorithm that collapses transition matrices from all possible worlds into one before applying B_LIN.

As shown in [13], for flattening- and collapsing-based methods, since flatApx and collApx outperform naive flatPPR and collPPR, we select the best competitors, flatApx and collApx, for our comparison.

Comparison with Certain PPR on Collapse- and Flattening-Based Schemes. There are also numerous excellent algorithms to compute certain PPR (*e.g.,* [3, 4, 10, 11, 16, 17, 19, 20, 26–30, 37]). To highlight the superiority of UPPR+ over these certain competitors applied to the merged graph from collapse-based and flattening-based schemes [13], we conduct experiments (Exp-6) to compare the total runtime of UPPR+ with the aggregation time for collapse-based (coll-only) and flattening-based (flat-only) methods excluding the runtime of any certain PPR algorithm on the merged graph.

Parameters. The following parameters are set by default, consistent with [13]: 1) Damping factor $\alpha = 0.8$. 2) For the baselines (exhApx, flatApx, collApx), we set the number of partitions $k = 5, 50, 500$ for small (FB, WV), medium (HP, WS), and large (others) datasets, respectively, and the target low rank for SVD is $r = 50$. 3) The average degree of uncertain edges $\lambda = 4$, and $l = |\mathcal{I}| = 6$.

Seeds Selection. We use a random stratified PageRank sampling method to select seeds, balancing node importance. Nodes are divided into 10 groups based on PageRank values, and 1% of nodes are randomly sampled from each group. The samples are combined to form 10% of the total nodes, ensuring a representative selection.

Uncertain Edges Sampling. To generate uncertain graphs with specified uncertainties, we follow the approach in [13] for uncertain edge sampling. Random edges in the original graph are selected and rendered uncertain by augmenting their target nodes with random nodes. The uncertain edges are concentrated around the seeds (which typically have high PPR scores), as uncertain edges farther from the seeds are deemed less significant according to [13].

Query Sampling. We randomly generated 100 queries per dataset, each following seed selection criteria, to ensure comprehensive coverage of both important and less important nodes. For statistical significance, we evaluate average accuracy, runtime, and memory usage across all queries, and report the average results per query.

Accuracy Measures. We use MAP (Mean Average Precision) and NDCG (Normalized Cumulative Discounted Gain).

Machine Configuration. All the experiments are run on an Intel Core i5-11500 (6 cores/12 threads) with 512GB RAM.

5.2 Experimental Results

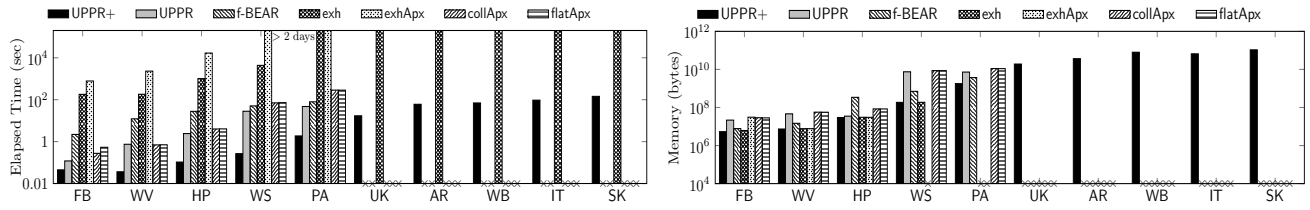
Exp-1: Time Efficiency & Scalablity. Figure 5a compares the computational time of UPPR+ with other algorithms on various real datasets. We discern that: 1) UPPR+ runs consistently 23–106x faster than other competitors on medium-sized graphs while scaling well on billion-sized datasets. This is due to our uncertain subspace reduction method that advances the aggregation of PPRs over all possible worlds via subset embedding, unlike UPPR which requires costly precomputation to invert intra-edge matrices and merge uncertainties across all possible worlds in the large original space. 2) On large graphs, only UPPR+ survives, with its runtime increasing mildly as graph size grows. This is compliant with the complexity analysis in Theorem 4, highlighting the effectiveness of our advanced aggregation technique.

Exp-2: Effect of # of Possible Worlds on Time. We next study how the number of possible worlds affects the runtime of each algorithm. Due to similar trends, we present results for PA and UK under mutual exclusion semantics. We control the number of possible worlds by varying (l, λ), where l is uncertain cardinality and λ is the average degree of uncertain edges.

We first vary l while fixing λ on both PA and UK. The results, shown in Figures 6a and 6b, reveal the following: (1) On PA, UPPR+ remains resilient to the exponential increase in possible worlds, unlike exh and exhApx whose costs rise sharply due to reliance on the original space. This is because UPPR+ efficiently aggregates PPRs within a compact uncertain subspace. (2) UPPR+ consistently outperforms f-BEAR, collApx, and flatApx by 37.8–271.9x. This advantage stems from its advanced subspace-based aggregation. (3) On large UK, only UPPR+ is viable, highlighting its superior scalability as the number of possible worlds grows exponentially. We next vary λ while fixing l on both PA and UK. Similar trends are shown in Figures 6c and 6d.

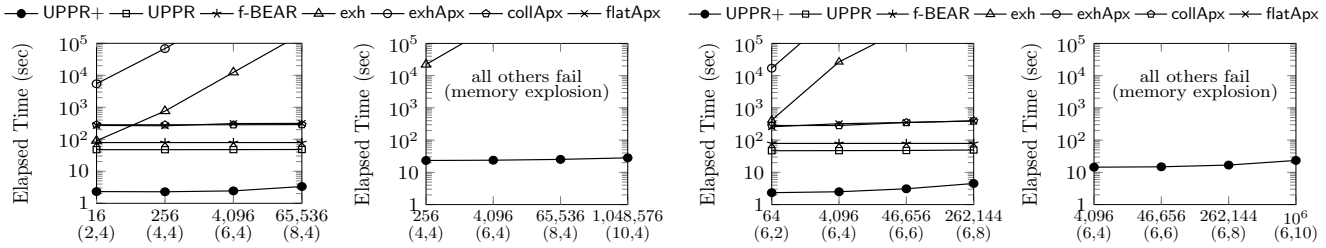
Exp-3: Memory Efficiency. Figure 5b compares the memory of UPPR+ with other algorithms. We observe that: 1) On small and medium graphs, where other algorithms do not fail, UPPR+ uses memory comparable to exh and f-BEAR, and less than other methods. This indicates that UPPR+ achieves substantial speedup without requiring excessive memory. 2) On large datasets, only UPPR+ survives, with its memory usage increasing linearly with graph size, which aligns with the space complexity analysis in Theorem 4.

Exp-4: Effect of # of Possible Worlds on Memory. Figure 7 shows the impact of the number of possible worlds on memory for each algorithm. We first vary l and fix λ, with results shown in Figure 7a and 7b. We notice that: 1) On PA, the memory of UPPR+ is consistently 5-10x less than all rivals, except exh. This is because UPPR+ does not require extra space to store inverted matrices over intra-block edges while aggregating PPRs within a reduced subspace. 2) On PA, when exh does not fail, the memory of UPPR+ is comparable to exh. This is reasonable since UPPR+ only requires a small subspace for uncertain edges aggregation. 3) On UK, only UPPR+ survives, with memory growing linearly as the number of



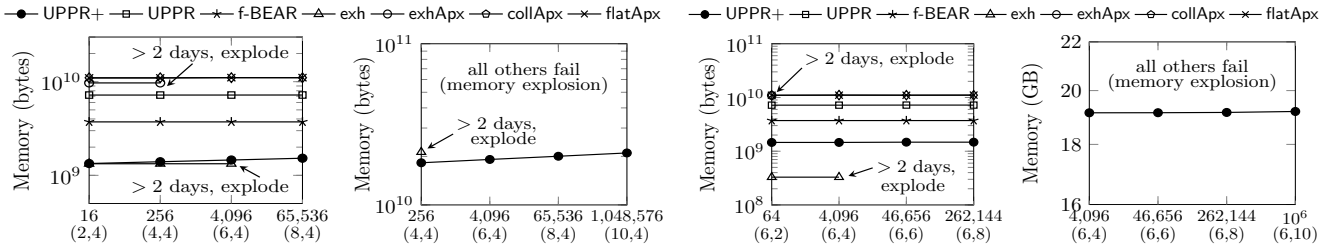
(a) Computational Time & Scalability (b) Memory Space

Figure 5: Computational Time, Scalability, and Memory Space on Various Real Datasets



(a) Varying l on PA (b) Varying l on UK (c) Varying λ on PA (d) Varying λ on UK

Figure 6: Effect of # of Possible Worlds Controlled by (l, λ) on Computational Time



(a) Varying l on PA (b) Varying l on UK (c) Varying λ on PA (d) Varying λ on UK

Figure 7: Effect of # of Possible Worlds Controlled by (l, λ) on Memory

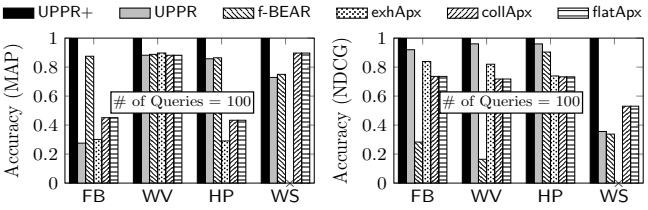


Figure 8: Accuracy/Exactness using MAP & NDCG

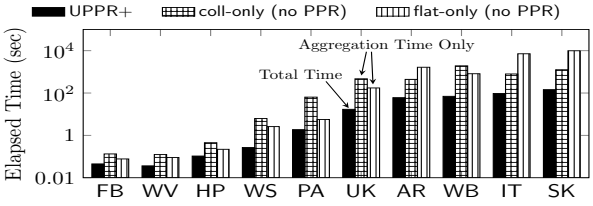


Figure 9: Comparing UPPR+ with Coll. and Flat. Time

possible worlds increases, showing its excellent scalability on large graphs. 4) The memory of UPPR+ shows less sensitivity to the exponential growth of the number of possible worlds.We next vary λ and fix l. The results in Figures 7c and 7d show similar trends.

Exp-5: Accuracy & Exactness. Figure 8 compares the accuracy of UPPR+ with other algorithms, using 2 metrics: MAP, NDCG. The ground truths were generated by performing exh for 100 iterations, ensuring accuracy to 9 decimal places. As exh is not scalable on large graphs, we use FB, WV, HP, WS for comparison. Results on other datasets are similar and omitted due to space constraints. We observe that: 1) UPPR+ consistently achieves the highest accuracy without any error. This validates the correctness of Theorems 1-3. 2) UPPR always exhibits lower accuracy than UPPR+, as it neglects several terms for approximation. 3) The accuracy of f-BEAR varies across different metrics, as the average PPR over all possible worlds is not equal to the PPR applied to the average transition matrix. 4) exhApx, collApx, and flatApx lose significant accuracy due to low-rank SVD approximation based on B-LIN.

Exp-6: UPPR+ vs. Certain PPR on Collapse- and Flattening-Based Schemes. To show that UPPR+ runs faster than any certain PPR algorithm (*e.g.,* [3, 4, 10, 11, 16, 17, 19, 20, 26–30, 37]) directly applied to the merged graph from collapse-based and flattening-based strawman schemes [13], Figure 9 compares the total time of UPPR+ with the aggregation time for collapse-based (coll-only) and flattening-based (flat-only) methods. For fairness, coll-only and flat-only account solely for the time required for aggregating all possible worlds into a single graph, excluding the runtime of any certain PPR algorithm on the merged graph. Results show that UPPR+ is consistently 1.7–75.4x faster than the aggregation time alone for collapsing and flattening transition matrices, with greater speedups on larger graphs. This demonstrates that any certain PPR algorithm applied to these schemes, which requires additional runtime over the merged graph, will be even slower than UPPR+.

6 Conclusion

We propose UPPR+, an efficient uncertain PPR algorithm on large graphs. First, we develop an uncertainty groupifying method and map uncertainties to a compact subspace using low-dimensional embeddings. Next, to accelerate the computation of subspace reduction method, we advance the aggregation of PPRs over all possible worlds in the subspace. Then, we analyse the time and space complexities of UPPR+. Experiments on ten real datasets show that UPPR+ achieves a 23–106x speedup over the best-known competitors with no loss of accuracy while exhibiting superior scalability on billion-edge graphs. As future work, it would be interesting to explore how uncertain semantics can be extended to a wide range of PageRank-like graph-based similarity measures [9, 12, 32, 34, 35].

A Appendix

A.1 Proof of Theorem 1

Proof. Let $\mathcal{I} = \{i_1, \cdots, i_l\}$, and $\mathcal{J}_w = \{j_{1,w}, \cdots, j_{l,w}\}$. If $\exists$ a certain edge $\langle i_k, x\rangle$ in $\mathcal{G}$ or $x = j_{k,w}$, $[T_w]_{x,i_k} = \frac{1}{d_{i_k}+[\Lambda_w]_{i_k,i_k}}$, where d_{i_k} is the number of the certain edges directed out of i_k. Otherwise, $[T_w]_{x,i_k} = 0$. By the definition of T_0 in Eq.(2),

$$[T_w - T_0]_{x,i_k} = \begin{cases} \frac{1}{d_{i_k}+[\Lambda_w]_{i_k,i_k}} - \frac{1}{d_{i_k}} & \text{if } \exists \text{ a certain edge } \langle i_k, x\rangle \quad (15a) \\ \frac{1}{d_{i_k}+[\Lambda_w]_{i_k,i_k}} & \text{if } x = j_{k,w} \quad (15b) \\ 0 & \text{otherwise} \quad (15c) \end{cases}$$

In the following, we will show $U_w V_w = T_w - T_0$:

(i) In case (15a), if $\exists$ a certain edge $\langle i_k, x\rangle$ in $\mathcal{G}$ and $x \neq j_{k,w}$, then $[E_n]_{x,\mathcal{J}_w}$ is a zero row vector, and $[T_0]_{x,i_k} = \frac{1}{d_{i_k}}$. Thus, by the definition of U_w in Eq.(4), it follows that

$$[U_w]_{x,*} = -[\Lambda_w]_{\mathcal{I},\mathcal{I}}[T_0]_{x,\mathcal{I}} = [*, \cdots, \overset{(k)}{-\frac{[\Lambda_w]_{i_k,i_k}}{d_{i_k}}}, *, \cdots, *] \quad (16)$$

Since $([D]_{\mathcal{I},\mathcal{I}} + \Lambda_w)^{-1} = diag(\frac{1}{d_{i_1}+[\Lambda_w]_{i_1,i_1}}, \cdots, \frac{1}{d_{i_l}+[\Lambda_w]_{i_l,i_l}})$ and $[E_n]_{\mathcal{I},i_k} = e_{i_k}$, it follows from the definition of V in Eq.(5) that

$$[V_w]_{*,i_k} = \frac{1}{d_{i_k}+[\Lambda_w]_{i_k,i_k}} e_{i_k} = [0, \cdots, 0, \overset{(k)}{\frac{1}{d_{i_k}+[\Lambda_w]_{i_k,i_k}}}, 0, \cdots, 0]^\top \quad (17)$$

Multiplying Eqs.(16) and (17) together produces case (15a) since

$$[U_w]_{x,*}[V_w]_{*,i_k} = -\frac{[\Lambda_w]_{i_k,i_k}}{d_{i_k}(d_{i_k}+[\Lambda_w]_{i_k,i_k})} = \frac{1}{d_{i_k}+[\Lambda_w]_{i_k,i_k}} - \frac{1}{d_{i_k}}.$$

Similarly, in cases (15b) and (15c), it can be readily verified that $[U_w]_{x,*}[V_w]_{*,i_k} = [T_w - T_0]_{x,i_k}$. We omit the details here. □

A.2 Proof of Theorem 2

Proof. By the definition of PPR, the closed form of p_w is

$$p_w = \mathsf{PPR}(T_w, s, \alpha) \;\Rightarrow\; p_w = (1-\alpha)(E_n - \alpha T_w)^{-1} s \quad (18)$$

We first express p_w in terms of p_0. By Theorem 1, $T_w = T_0 + U_w V_w$. Plugging this into Eq.(18) produces

$$p_w = (1-\alpha)(R^{-1} - \alpha U_w V_w)^{-1} s \;\text{ with }\; R = (E_n - \alpha T_0)^{-1} \quad (19)$$

Applying the Woodbury matrix formula to Eq.(19) yields

$$\begin{aligned} p_w &= \underbrace{(1-\alpha)Rs}_{=p_0} + \alpha \underbrace{RU_w}_{\triangleq B_w} (E_l - \alpha V_w \underbrace{RU_w}_{\triangleq B_w})^{-1} V_w \underbrace{(1-\alpha)Rs}_{=p_0} \\ &= p_0 + \alpha B_w h_w \end{aligned} \quad (20)$$

where $B_w = RU_w$ and $h_w = (E_l - \alpha V_w B_w)^{-1} V_w p_0$.

To simplify the expression of h_w, we apply V_w in Theorem 1 and $[E_n]_{\mathcal{I},*} p_0 = [p_0]_{\mathcal{I}}$, which produces

$$V_w p_0 = ([D]_{\mathcal{I},\mathcal{I}} + \Lambda_w)^{-1} [p_0]_{\mathcal{I}} \quad (21)$$

Thus, substituting Eq.(21) into h_w, we obtain

$$\begin{aligned} h_w &= (E_l - \alpha V_w B_w)^{-1} ([D]_{\mathcal{I},\mathcal{I}} + \Lambda_w)^{-1} [p_0]_{\mathcal{I}} \\ &= \left([D]_{\mathcal{I},\mathcal{I}} + \Lambda_w - \alpha([D]_{\mathcal{I},\mathcal{I}} + \Lambda_w) V_w B_w\right)^{-1} [p_0]_{\mathcal{I}} \end{aligned} \quad (22)$$

Next, we will provide a simple expression of B_w in Eq.(22). Since $R = (E_n - \alpha T_0)^{-1} = \sum_{k=0}^{\infty} (\alpha T_0)^k$, it follows that

$$R[T_0]_{*,\mathcal{I}} = \frac{1}{\alpha} \sum_{k=1}^{\infty} (\alpha T_0)^k [E_n]_{*,\mathcal{I}} = \frac{1}{\alpha}(R - E_n)[E_n]_{*,\mathcal{I}} \quad (23)$$

Plugging Eqs.(4) and (23) into $B_w = RU_w$ yields

$$\begin{aligned} B_w = RU_w &= R[E_n]_{*,\mathcal{J}_w} - \tfrac{1}{\alpha}(R - E_n)[E_n]_{*,\mathcal{I}} \Lambda_w \\ &= [R]_{*,\mathcal{J}_w} - \tfrac{1}{\alpha}[R]_{*,\mathcal{I}} \Lambda_w + \tfrac{1}{\alpha}[E_n]_{*,\mathcal{I}} \Lambda_w \end{aligned} \quad (24)$$

Since Eq.(5) implies that $([D]_{\mathcal{I},\mathcal{I}} + \Lambda_w) V_w = [E_n]_{\mathcal{I},*}$, applying this and plugging Eq.(24) into Eq.(22), we have

$$\begin{aligned} \alpha([D]_{\mathcal{I},\mathcal{I}} + \Lambda_w) V_w B_w &= \alpha [E_n]_{\mathcal{I},*} B_w \\ &= \alpha [R]_{\mathcal{I},\mathcal{J}_w} - [R]_{\mathcal{I},\mathcal{I}} \Lambda_w + \Lambda_w \end{aligned} \quad (25)$$

Combining Eq.(25) with Eq.(22), we get

$$h_w = ([D]_{\mathcal{I},\mathcal{I}} - \alpha[R]_{\mathcal{I},\mathcal{J}_w} + [R]_{\mathcal{I},\mathcal{I}} \Lambda_w)^{-1} [p_0]_{\mathcal{I}} \quad (26)$$

Finally, to obtain the expression of p_w in Eq.(9), we plug Eqs.(24) and (26) into (20), which produces

$$p_w = p_0 + \alpha[R]_{*,\mathcal{J}_w} h_w - [R]_{*,\mathcal{I}} \Lambda_w h_w + [E_n]_{*,\mathcal{I}} \Lambda_w h_w \quad (27)$$

In Eq.(27), since

$$\begin{aligned} \alpha[R]_{*,\mathcal{J}_w} h_w = R\eta_w \quad &\text{with} \quad \eta_w = [E_n]_{*,\mathcal{J}_w}(\alpha h_w) \\ [R]_{*,\mathcal{I}} \Lambda_w h_w = R\xi_w \quad &\text{with} \quad \xi_w = [E_n]_{*,\mathcal{I}} \Lambda_w h_w \end{aligned}$$

it follows that

$$p_w = p_0 + R\eta_w - R\xi_w + \xi_w = p_0 + \xi_w + R(\eta_w - \xi_w) \quad (28)$$

Let $x = \eta_w - \xi_w$, by the definition of R in Eq.(19), we have

$$Rx = (E_n - \alpha T_0)^{-1} x = \tfrac{1}{1-\alpha} \mathsf{PPR}(T_0, x, \alpha)$$

Combining this with Eq.(28) produces Eq.(9).

□

A.3 Proof of Theorem 3

Proof. According to Theorem 2, we take the sums over all possible worlds of $\mathcal{G}$ on both sides of Eq.(9), which yields

$$\begin{aligned} \sum_{w \in \mathsf{pw}_u(\mathcal{G})} p_w &= |\mathsf{pw}_u(\mathcal{G})| \cdot (p_0 + \xi) \\ &\quad + \tfrac{1}{1-\alpha} \sum_{w \in \mathsf{pw}_u(\mathcal{G})} \mathsf{PPR}\,(T_0, \eta_w - \xi_w, \alpha) \end{aligned} \quad (29)$$

By the definition of uncertain PPR $p = \frac{1}{|\mathsf{pw}_u(\mathcal{G})|} \sum_w p_w$ and $\frac{1}{|\mathsf{pw}_u(\mathcal{G})|} \sum_w \mathsf{PPR}(T_0, \eta_w - \xi_w, \alpha) = \mathsf{PPR}(T_0, (\frac{1}{|\mathsf{pw}_u(\mathcal{G})|} \sum_w \eta_w) - \xi, \alpha)$ according to Observation 2, it follows from Eq.(29) that

$$p = p_0 + \xi + \tfrac{1}{1-\alpha} \mathsf{PPR}\,(T_0, \zeta, \alpha) \text{ with } \zeta = (\tfrac{1}{|\mathsf{pw}_u(\mathcal{G})|} \textstyle\sum_w \eta_w) - \xi \quad \square$$

References

[1] Konstantin Avrachenkov, Vladimir Dobrynin, Danil Nemirovsky, Kim Son Pham, and Elena Smirnova. 2008. PageRank based clustering of hypertext document collections. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*, Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, and Mun-Kew Leong (Eds.). ACM, 873–874. https://doi.org/10.1145/1390334.1390549

[2] Konstantin Avrachenkov, Nelly Litvak, Danil Nemirovsky, and Natalia Osipova. 2007. Monte Carlo methods in PageRank computation: When one iteration is sufficient. *SIAM J. Numer. Anal.* 45, 2 (2007), 890–904.

[3] Bahman Bahmani, Kaushik Chakrabarti, and Dong Xin. 2011. Fast Personalized PageRank on MapReduce. In *ACM SIGMOD*, Timos K. Sellis, Renée J. Miller, Anastasios Kementsietsidis, and Yannis Velegrakis (Eds.). ACM, 973–984. https://doi.org/10.1145/1989323.1989425

[4] Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. 2010. Fast Incremental and Personalized PageRank. *Proc. VLDB Endow.* 4, 3 (2010), 173–184. https://doi.org/10.14778/1929861.1929864

[5] Pavel Berkhin. 2006. Bookmark-coloring algorithm for Personalized PageRank computing. *Internet Mathematics* 3, 1 (2006), 41–62.

[6] Paolo Boldi and Sebastiano Vigna. 2002. LAW Datasets: The Laboratory for Web Algorithmics Dataset Collection. https://law.di.unimi.it/datasets.php.

[7] Yasuhiro Fujiwara, Makoto Nakatsuji, Hiroaki Shiokawa, Takeshi Mishima, and Makoto Onizuka. 2013. Efficient ad-hoc search for personalized PageRank. In *ACM SIGMOD*, Kenneth A. Ross, Divesh Srivastava, and Dimitris Papadias (Eds.). ACM, 445–456. https://doi.org/10.1145/2463676.2463717

[8] Takayasu Fushimi, Kazumi Saito, Kouzou Ohara, Masahiro Kimura, and Hiroshi Motoda. 2020. Efficient Computing of PageRank Scores on Exact Expected Transition Matrix of Large Uncertain Graph. In *IEEE International Conference on Big Data*, Xintao Wu, Chris Jermaine, Li Xiong, Xiaohua Hu, Olivera Kotevska, Siyuan Lu, Weija Xu, Srinivas Aluru, Chengxiang Zhai, Eyhab Al-Masri, Zhiyuan Chen, and Jeff Saltz (Eds.). IEEE, 916–923. https://doi.org/10.1109/BIGDATA50022.2020.9378076

[9] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.). ACM, 855–864. https://doi.org/10.1145/2939672.2939754

[10] Tao Guo, Xin Cao, Gao Cong, Jiaheng Lu, and Xuemin Lin. 2017. Distributed Algorithms on Exact Personalized PageRank. In *ACM SIGMOD*, Semih Salihoglu, Wenchao Zhou, Rada Chirkova, Jun Yang, and Dan Suciu (Eds.). ACM, 479–494. https://doi.org/10.1145/3035918.3035920

[11] Guanhao Hou, Qintian Guo, Fangyuan Zhang, Sibo Wang, and Zhewei Wei. 2023. Personalized PageRank on Evolving Graphs with an Incremental Index-Update Scheme. *Proc. ACM Manag. Data* 1, 1 (2023), 25:1–25:26. https://doi.org/10.1145/3588705

[12] Glen Jeh and Jennifer Widom. 2002. SimRank: a measure of structural-context similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*. ACM, 538–543. https://doi.org/10.1145/775047.775126

[13] Jung Hyun Kim, Mao-Lin Li, K. Selçuk Candan, and Maria Luisa Sapino. 2017. Personalized PageRank in Uncertain Graphs with Mutually Exclusive Edges. In *ACM SIGIR*, Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White (Eds.). ACM, 525–534. https://doi.org/10.1145/3077136.3080794

[14] N Kishore Kumar and Jan Schneider. 2017. Literature survey on low rank approximation of matrices. *Linear and Multilinear Algebra* 65, 11 (2017), 2212–2244.

[15] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[16] Qin Liu, Zhenguo Li, John C. S. Lui, and Jiefeng Cheng. 2016. PowerWalk: Scalable Personalized PageRank via Random Walks with Vertex-Centric Decomposition. In *CIKM*, Snehasis Mukhopadhyay, ChengXiang Zhai, Elisa Bertino, Fabio Crestani, Javed Mostafa, Jie Tang, Luo Si, Xiaofang Zhou, Yi Chang, Yunyao Li, and Parikshit Sondhi (Eds.). ACM, 195–204. https://doi.org/10.1145/2983323.2983713

[17] Peter Lofgren, Siddhartha Banerjee, and Ashish Goel. 2016. Personalized PageRank Estimation and Search: A Bidirectional Approach. In *ACM WSDM*, Paul N. Bennett, Vanja Josifovski, Jennifer Neville, and Filip Radlinski (Eds.). ACM, 163–172. https://doi.org/10.1145/2835776.2835823

[18] Peter Lofgren, Siddhartha Banerjee, Ashish Goel, and Seshadhri Comandur. 2014. FAST-PPR: Scaling personalized pagerank estimation for large graphs. In *ACM SIGKDD*, Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani (Eds.). ACM, 1436–1445. https://doi.org/10.1145/2623330.2623745

[19] Takanori Maehara, Takuya Akiba, Yoichi Iwata, and Ken-ichi Kawarabayashi. 2014. Computing Personalized PageRank Quickly by Exploiting Graph Structures. *Proc. VLDB Endow.* 7, 12 (2014), 1023–1034. https://doi.org/10.14778/2732977.2732978

[20] Naoto Ohsaka, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Efficient PageRank Tracking in Evolving Networks. In *ACM SIGKDD*, Longbing Cao, Chengqi Zhang, Thorsten Joachims, Geoffrey I. Webb, Dragos D. Margineantu, and Graham Williams (Eds.). ACM, 875–884. https://doi.org/10.1145/2783258.2783297

[21] Lawrence Page, Sergey Brin, Rajeev Motwani, and T Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report.

[22] Kijung Shin, Jinhong Jung, Lee Sael, and U Kang. 2015. BEAR: Block Elimination Approach for Random Walk with Restart on Large Graphs. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, Timos K. Sellis, Susan B. Davidson, and Zachary G. Ives (Eds.). ACM, 1571–1585. https://doi.org/10.1145/2723372.2723716

[23] Zitan Sun, Xin Huang, Jianliang Xu, and Francesco Bonchi. 2021. Efficient Probabilistic Truss Indexing on Uncertain Graphs. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM / IW3C2, 354–366. https://doi.org/10.1145/3442381.3449976

[24] Hanghang Tong, Christos Faloutsos, and Jia-yu Pan. 2006. Fast Random Walk with Restart and Its Applications. In *Sixth International Conference on Data Mining (ICDM'06)*. 613–622. https://doi.org/10.1109/ICDM.2006.70

[25] Sotiris Tsioutsiouliklis, Evaggelia Pitoura, Konstantinos Semertzidis, and Panayiotis Tsaparas. 2022. Link Recommendations for PageRank Fairness. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini (Eds.). ACM, 3541–3551. https://doi.org/10.1145/3485447.3512249

[26] Hanzhi Wang, Zhewei Wei, Junhao Gan, Sibo Wang, and Zengfeng Huang. 2020. Personalized PageRank to a Target Node, Revisited. In *KDD*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 657–667. https://doi.org/10.1145/3394486.3403108

[27] Hanzhi Wang, Zhewei Wei, Junhao Gan, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. 2022. Edge-based Local Push for Personalized PageRank. *PVLDB* 15, 7 (2022), 1376–1389. https://doi.org/10.14778/3523210.3523216

[28] Runhui Wang, Sibo Wang, and Xiaofang Zhou. 2019. Parallelizing approximate single-source personalized PageRank queries on shared memory. *VLDB J.* 28, 6 (2019), 923–940. https://doi.org/10.1007/s00778-019-00576-7

[29] Zhewei Wei, Xiaodong He, Xiaokui Xiao, Sibo Wang, Shuo Shang, and Ji-Rong Wen. 2018. TopPPR: Top-k Personalized PageRank Queries with Precision Guarantees on Large Graphs. In *ACM SIGMOD*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 441–456. https://doi.org/10.1145/3183713.3196920

[30] Zhewei Wei, Ji-Rong Wen, and Mingji Yang. 2024. Approximating Single-Source Personalized PageRank with Absolute Error Guarantees. In *ICDT (LIPIcs)*, Graham Cormode and Michael Shekelyan (Eds.), Vol. 290. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 9:1–9:19. https://doi.org/10.4230/LIPICS.ICDT.2024.9

[31] Mingji Yang, Hanzhi Wang, Zhewei Wei, Sibo Wang, and Ji-Rong Wen. 2024. Efficient Algorithms for Personalized PageRank Computation: A Survey. *IEEE Trans. Knowl. Data Eng.* 36, 9 (2024), 4582–4602. https://doi.org/10.1109/TKDE.2024.3376000

[32] Weiren Yu. 2025. SimEdge: A Scalable Transitivity-Aware Graph-Theoretic Similarity Model for Capturing Edge-to-Edge Relationships. In *WWW '25: The ACM Web Conference 2025, Sydney, Australia, April 28 - May 2, 2025*. ACM.

[33] Weiren Yu and Julie A. McCann. 2016. Random Walk with Restart over Dynamic Graphs. In *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu (Eds.). IEEE Computer Society, 589–598. https://doi.org/10.1109/ICDM.2016.0070

[34] Weiren Yu, Julie A. McCann, Chengyuan Zhang, and Hakan Ferhatosmanoglu. 2022. Scaling High-Quality Pairwise Link-Based Similarity Retrieval on Billion-Edge Graphs. *ACM Trans. Inf. Syst.* 40, 4 (2022), 78:1–78:45. https://doi.org/10.1145/3495209

[35] Weiren Yu, Jian Yang, Maoyin Zhang, and Di Wu. 2022. CoSimHeat: An Effective Heat Kernel Similarity Measure Based on Billion-Scale Network Topology. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*. ACM, 234–245. https://doi.org/10.1145/3485447.3511952

[36] Ye Yuan, Guoren Wang, Lei Chen, and Haixun Wang. 2012. Efficient Subgraph Similarity Search on Large Probabilistic Graph Databases. *Proc. VLDB Endow.* 5, 9 (2012), 800–811. https://doi.org/10.14778/2311906.2311908

[37] Hongyang Zhang, Peter Lofgren, and Ashish Goel. 2016. Approximate Personalized PageRank on Dynamic Graphs. In *ACM SIGKDD*, Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.). ACM, 1315–1324. https://doi.org/10.1145/2939672.2939804