Effective and Scalable Manifold Ranking-Based Image Retrieval with Output Bound

DANDAN LIN, Tencent Inc.
VICTOR JUNQIU WEI, The Hong Kong Polytechnic University
RAYMOND CHI-WING WONG, The Hong Kong University of Science and Technology

Image retrieval keeps attracting a lot of attention from both academic and industry over past years due to its variety of useful applications. Due to the rapid growth of deep learning approaches, more better feature vectors of images could be discovered for improving image retrieval. However, most (if not all) existing deep learning approaches consider the similarity between two images *locally* without considering the similarity among a *group* of similar images *globally*, and thus could not return accurate results. In this article, we study the image retrieval with manifold ranking (MR) which considers both the local similarity and the global similarity, which could give more accurate results. However, existing best-known algorithms have one of the following issues: (1) they require to build a bulky index, (2) some of them do not have any theoretical bound on the output, and (3) some of them are time-consuming. Motivated by this, we propose two algorithms, namely *Monte Carlo-based MR* (*MCMR*) and *MCMR+*, for image retrieval, which do not have the above issues. We are the first one to propose an index-free manifold ranking image retrieval with the output theoretical bound. More importantly, our algorithms give the first best-known time complexity result of $O(n \log n)$ where n is the total number of images in the database compared with the existing best-known result of $O(n^2)$ in the literature of computing the exact top-k results with quality guarantee. Lastly, our experimental result shows that *MCMR+* outperforms existing algorithms by up to four orders of magnitude in terms of query time.

CCS Concepts: • **Information systems** → **Similarity measures**; • **Mathematics of computing** → **Graph algorithms;**

Additional Key Words and Phrases: Similarity measures, manifold ranking, efficient algorithms, image retrieval

ACM Reference format:
Dandan Lin, Victor Junqiu Wei, and Raymond Chi-Wing Wong. 2023. Effective and Scalable Manifold Ranking-Based Image Retrieval with Output Bound. *ACM Trans. Knowl. Discov. Data.* 17, 5, Article 61 (April 2023), 31 pages.
https://doi.org/10.1145/3565574

For Dandan Lin, work was done while she was at The Hong Kong University of Science and Technology.

The research of Raymond Chi-Wing Wong is supported by PRP/026/21FX. The research of Victor Junqiu Wei is supported by PolyU internal fund (1-BD47) under the research project (P0039657) of the Hong Kong Polytechnic University.

Authors' addresses: D. Lin, Tencent Inc., Shenzhen 518054, China; email: danieslin@tencent.com; V. J. Wei (corresponding author), The Hong Kong Polytechnic University, Hung Hom, Kowloon 999077, Hong Kong, China; email: junqiu.wei@polyu.edu.hk; R. C.-W. Wong, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon 999077, Hong Kong, China; email: raywong@cse.ust.hk.

1556-4681/2023/04-ART61 $15.00
https://doi.org/10.1145/3565574

1 INTRODUCTION

Image retrieval [5] keeps attracting a lot of attention from both academic and industry over past years due to its variety of useful applications. In academic, before the growth of deep learning studies, most researchers [5, 6, 67] studied how to "engineer" good image features (manually) such that any two given images should have a high "pre-defined" similarity measure if they look similar to human. Recently, due to the growth of deep learning approaches, researchers focused on how to use deep learning models [64] like **convolutional neural networks** (**CNNs**) to capture or find the "embedded" features to get rid of (manual) feature engineering. Besides, it is found [14, 40, 44, 49, 64] that the embedded features capture a lot of important and good ingredients in the image, resulting a good performance of some tasks (e.g., similarity search) in image retrieval. In industry, giant technology companies in the world have large research teams for image retrieval due to their attractive applications. One example is Taobao, the biggest mobile e-commerce platform in China, with its famous application of "similar item search" which returns a list of items which are similar to the photo of an item that a customer would like to buy [63]. Another example is Google, the world leading search company hosted in US, with its image search engine function which returns a list of images similar to an image uploaded by users.

1.1 Feature Extraction and Similarity Search

Specifically, image retrieval involves two phases [6, 13, 14, 22, 40, 41, 49, 51, 64], namely the feature extraction and the similarity search. The feature extraction is to learn a representation of each image as a *feature vector* which is a d-dimensional vector. The similarity search is to find a list of images which are "similar" to the given image.

Feature extraction is a very fundamental and important phase for image retrieval. With good feature extraction, the similarity search (in the second phase) could be done more accurately and more effectively. Due to the successful development of deep learning approaches, the features found by these approaches could capture image ingredients well [14, 40, 41, 44, 64]. These deep learning approaches employ the CNN frameworks to find the features. Some representative approaches are the basic CNN approaches [14, 40, 44] and the advanced CNN approach called *MAC* [40, 41] which exploits a feature of focusing essential parts of images and could be regarded as the state-of-the-art in the literature. In [64], it was found that the features could help to improve the accuracy of the similarity search by up to 51.3%.

Although feature extraction is well-studied among deep learning approaches, since most studies [14, 40, 41] directly adopt a "traditional" $\mathcal{L}^p$-norm-based measure (e.g., the Euclidean Distance and Inner Product) of evaluating the similarity of two images, the performance of similarity search is not that good. It is because this traditional measure only captures the similarity between two images *locally* based on their d-dimensional representation vectors (termed *local similarity*) without considering the similarity among a *group* of similar images *globally* (termed *global similarity*) [65]. Note that the d-dimensional representation vectors of all images can form a d-dimensional feature space. However, two similar images are not always adjacent in the $\mathcal{L}^p$-space, and hence, the similarity score between two images by the $\mathcal{L}^p$-norm-based measures might be incorrect [8]. For example, for two images with same color but different shape, the Euclidean distance of their representation vectors might be very small and hard to distinguish clearly. However, by considering the global similarity that exploits more information revealed by a group of similar images, the error inside above local similarity could be narrowed. For example, if the query image is blue triangle, we could utilize more similar images in the image dataset that contains the triangle shape and aggregate the information revealed by these triangle images, it is of high probability that the "similarity distance" of the query image and an image of blue circle could be enlarged. In this case, the image of blue circle can be excluded from the similar images results of the query one. Thus, both local similarity and global similarity are important to be included in a similarity measurement.

1.2 Manifold Ranking: A Better Similarity Search Approach

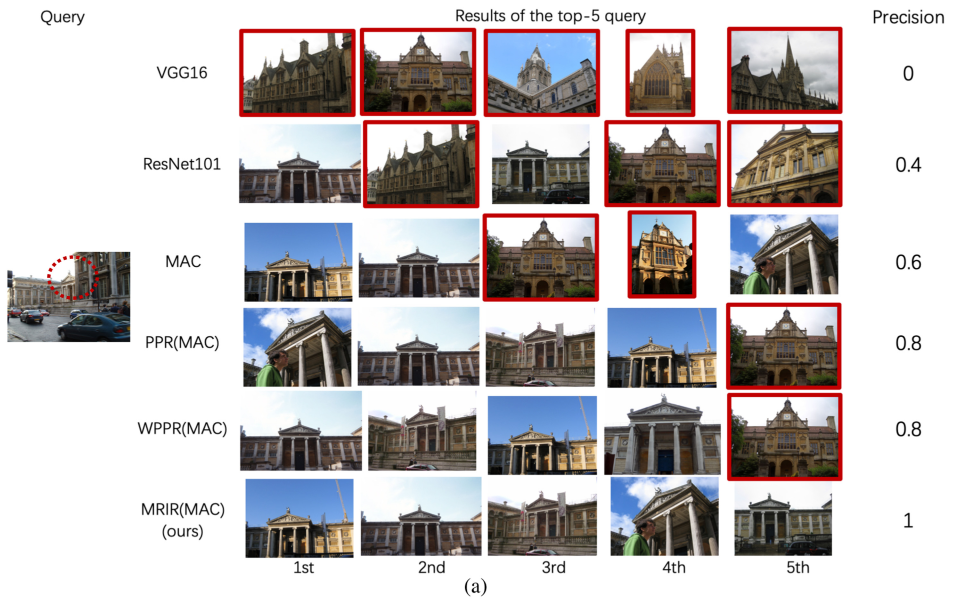
However, **Manifold ranking** (**MR**) [13, 65, 66], one similarity measure for similarity search, is found to be an effective measure of capturing the similarity both locally and globally. Due to this diverse ability, in recent years, MR is applied for not only image retrieval but also other problems such as *person re-identification* [3, 34], *document similarity search* [50], *identifying quantitative chemical relationship* [42], *saliency detection* [19, 38, 47, 55, 60], and *object co-segmentation* [39].

Next, we would like to give two case studies showing how MR has better performance compared with some best-known models. We included three deep learning approaches (all of which use the $\mathcal{L}^p$-norm-based measure for similarity search) for comparison, namely *VGG16* [44], *ResNet101* [14], and MAC [40, 41], where *MAC* is considered as the state-of-the-art in the literature. We also included two popular similarity approaches for comparison, namely ***Personalized PageRank*** (**PPR**) [59] and ***Weighted Personalized PageRank*** (**WPPR**). We call the MR framework as the **manifold ranking image retrieval** (***MRIR***). Note that since PPR, WPPR, and MRIR require the feature space for computation, we adopt the feature space learnt by the state-of-the-art deep learning approach, *MAC*, for this purpose. We denote these three models with *PPR(MAC)*, *WPPR(MAC),* and *MRIR(MAC)* where the feature space is denoted in the bracket of the notation of the model names. The implementation details of the case studies could be found in Section 6.9.

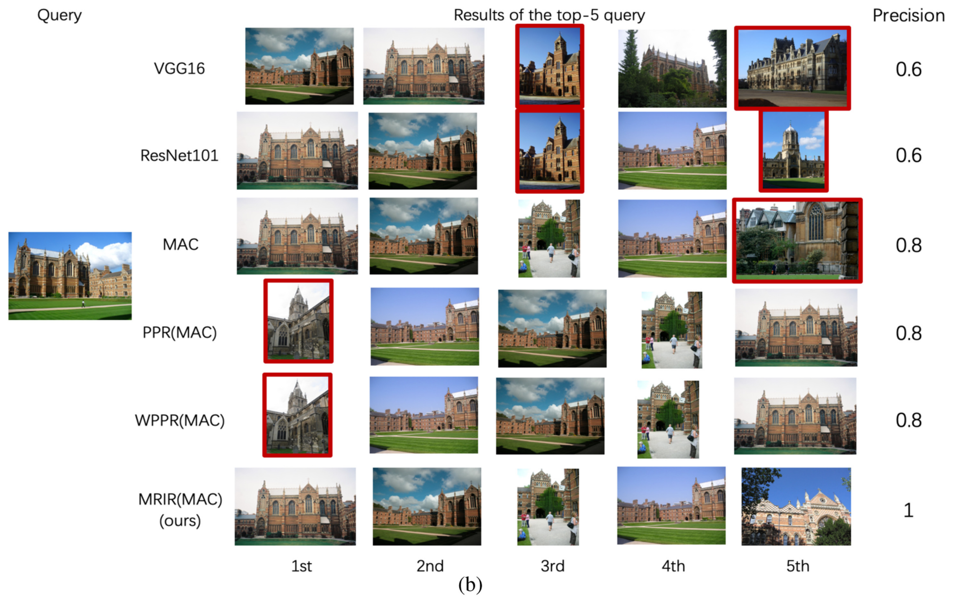
Figure 1 shows the results of two case studies on a benchmark dataset *Roxford5k* which contains a list of images of buildings in Oxford. In Figure 1(a), the query image is given on the left-hand side where the object in the red dotted circle denotes the "special" part/characteristics of the building which should be found in the "correct" output. On the right-hand side, we could see the result of the top-5 query returned by each mentioned model. For each model, the result has two parts: (1) the first part contains the five images with an ordering from the most similar image (on the left side) to the least similar image (on the right side) in the output of this model, and (2) the second part is the precision of this model based on these five images. Besides, in the first part, if the image in the result of each model is incorrect (*i.e.*, the building in the image is different from that in query image), it is enclosed with a red border boundary. In this case study, we found that only *MRIR(MAC)* could obtain 100% precision but all others returned incorrect images. In Figure 1(b), similar results are obtained for another query image. It is because the similarity measurements the baselines used are not good. Specifically, for the VGG16, ResNet101, and MAC methods, they used the traditional $\mathcal{L}^p$-based similarity measure (i.e., Inner Product), which computes the similarity between two images based on their representation vectors (termed as local similarity) and cannot reveal the global similarity by exploiting more information in the whole image datasets. Although both the PPR(MAC) and WPPR(MAC) methods could combine both local and global similarity, they still suffer from a limitation that they consider only the graph topological information, and so, the global similarity they used is not that accurate as our method. However, our method *MRIR* considers not only the local similarity but also the global similarity which combines the graph topology with a weighted information computed by a heat kernel on the feature space revealed by the whole image dataset. Thus, compared with these baselines, MRIR is more effective to improve the accuracy of image retrieval task.

1.3 Our Proposed Methods: *MCMR* and *MCMR+*

Given a querying image, MR exploits both the local similarity and the global similarity to compute a score called the *MR* score for each image. In a top-k query, the images with the greatest MR scores are returned as an output.



(a)



(b)

Fig. 1. Top-5 Query Results on Dataset *Roxford5k* Returned by VGG16 [44], ResNet101 [14], MAC [41], PPR(MAC) [59], WPPR(MAC), and MRIR(MAC) (ours).

Definition 1 (Top-k MR Search). Given an image database, a query image q and a constant k, the top-k MR search finds the top-k images with the highest MR scores w.r.t q.

As shown above, MR is good for accurate image retrieval. We consider the following three requirements for evaluating a method M for MR:

— *Index-free:* M has no index for computing MR scores.

Table 1. Summary of Existing Methods for MR

Method	*Mogul-E* [8] (the fastest exact algorithm)	*Mogul* [8] (the fastest approximate algorithm)	*MCMR* and *MCMR+* (Our Methods)
Index-free?	No	No	Yes
Output Bound?	Yes	No	Yes
Efficiency?	No	Yes	Yes

— *Output bound:* The MR scores of images returned by M should have a theoretical quality guarantee.
— *Efficiency:* M is computationally cheap to return the MR scores.

For the first index-free requirement, unfortunately, most (if not all) existing algorithms about computing MR scores require to build a bulky index, hindering the flexibility of any image database update. For example, in the Taobao platform involving at least 2 billion items (or images) [54], the existing algorithms are not scalable in this large dataset due to the bulky index. For the second output bound requirement, all existing approximate algorithms compute only the approximate MR scores *without any theoretical guarantee*. Then, to answer the top-k MR search problem, all of them return the top-k nodes with the largest approximate MR scores based on their approximate computation. Thus, the top-k nodes they returned are very likely to be inaccurate. Besides, they cannot provide the theoretical analysis to guarantee the accuracy of the returned top-k results. Although existing exact algorithms could return answers with theoretical guarantee (*i.e.*, the exact MR scores), they suffer from the bulky index size problem. For the third efficiency requirement, all exact algorithms are time-consuming. Although existing approximate algorithms could return answers in a short time, they still have no theoretical guarantee on the output. Therefore, none of existing algorithms can satisfy the three requirements simultaneously.

To tackle these issues, we propose two approaches termed *MCMR* and *MCMR+*, both of which satisfy all these requirements simultaneously. Table 1 summarizes existing algorithms and our proposed algorithms in terms of above three requirements.

The following shows our contributions.

— We propose two algorithms, namely *MCMR* and *MCMR+*, both of which adopt the random walk sampling strategy *without pre-computing any index*. Besides, we provide a non-trivial theoretical analysis on the proposed unbiased weighted random walk sampling strategy in terms of correctness. Based on *MCMR*, *MCMR+* is further enriched with a refinement step to prune the nodes that are not in the top-k results, leading to higher efficiency.
— To the best of our knowledge, we are the first to propose index-free algorithms for computing MR scores. All existing algorithms require to build an index for efficient computation.
— The time complexities of *MCMR* and *MCMR+* are $O(n \log n)$ where n is the total number of images in the database. This is the first best-known time complexity result in the literature of returning the exact top-k results with quality guarantee. The existing best-known time complexity in this literature [8] is $O(n^2)$.
— We conducted comprehensive experiments on four real-world image datasets. The experimental results show that *MCMR+* outperforms existing algorithms *by up to four orders of magnitudes* in terms of query time.

This manuscript is a journal extension to our previous conference paper [28]. We summarize the main differences from our conference version below. (1) Although *MCMR* in [28] has shown its ability to balance the tradeoff between efficiency and accuracy, it wastes time to estimate the MR scores of a large number of *unnecessary* nodes who are not in the top-k set. In Section 5, we

Table 2. The List of Symbols

Symbol	Description
$G(V, E)$	The k-NN graph G with nodes set V and edges set E
n, m	The number of nodes and edges in the graph, respectively
$d(u)$	The degree of node u in the k-NN graph
$\mathbf{A}$	The adjacency matrix of graph G
$\mathbf{C}$	The diagonal matrix of $\mathbf{A}$ where $C_{ii} = \sum_{j=1}^{n} A_{ij}$ for $i \in [1, n]$ and $C_{ik} = 0$ for each $i \neq k$
$\mathbf{W}$	The symmetrically normalized matrix of $\mathbf{A}$ such that $\mathbf{W} = \mathbf{C}^{-1/2}\mathbf{A}\mathbf{C}^{-1/2}$
$\mathbf{D}$	The diagonal matrix of $\mathbf{W}$ where $D_{ii} = \sum_{j=1}^{n} W_{ij}$ for $i \in [1, n]$ and $D_{ik} = 0$ for each $i \neq k$
q	The query node
α	The constant parameter of MR
k	The number of answered nodes in the top-k search
$\mathbf{q}$	The $n \times 1$ query vector of zeros except that $q(q) = 1$
$\mathbf{x}_q^*$	The exact MR scores vector w.r.t the query node q
$x_q^*(v)$	The exact MR score of node v w.r.t q
$\pi(q, v)$	The estimated MR score of v w.r.t q
v_k	The node whose exact MR score w.r.t q is the kth largest
n_r	The number of weighted random walks
gap_k	The difference between the kth and $(k+1)$-th largest exact MR scores w.r.t q such that $gap_k = x_q^*(v_k) - x_q^*(v_{k+1})$
$r^f(q, v)$	The residue of node v w.r.t q in *Local Search*
$\pi^f(q, v)$	The reserve of node v w.r.t q in *Local Search*
r_{max}^f	The residue threshold used in *Local Search*
r_{sum}^f	The sum of residues of all nodes w.r.t q such that $r_{sum}^f = \sum_{v \in V} r^f(q, v)$
$\beta(q, v)$	The confidence bound of node v w.r.t q

propose a more efficient algorithm called *MCMR+*, which is enriched with a refinement step to prune iteratively the nodes that are not in the top-k results, leading to higher efficiency. Although our newly developed algorithm *MCMR+* in this article adopts *MCMR* as an inherent component, the newly developed algorithm *MCMR+* is *non-trivial* by returning the exact top-k results with accuracy guarantee in the refinement step and the whole algorithm could not be found in [28]. (2) In Section 5.5, we prove that the prune technique designed in *MCMR+* can correctly compute the lower and upper bounds of exact MR scores of all nodes with high probability. This theoretical analysis shows the correctness of the newly developed algorithm *MCMR+*. Besides, in Section 5.6, we provide the time complexity analysis of *MCMR+*, which takes $O(n \log n)$ time cost. Both theoretical analysis could not be found in [28]. Besides, our experimental results demonstrate that *MCMR+* further achieves better practical performance than *MCMR* in [28] *by up to two orders of magnitude* in terms of query time. Furthermore, in Section 6.4, we examined the effectiveness of *MCMR+* compared with *MCMR* to see how *MCMR+* reduces the query time cost for achieving the exact top-k results. (3) In this article, we also provide comprehensive experiments on evaluating the performance of our proposed methods *MCMR* and *MCMR+* in terms of parameter sensitivity of proposed methods (see Section 6.5), preprocessing time cost of proposed methods when the dataset is dynamically updated (see Section 6.6) and query time cost for out-of-sample queries in datasets (see Section 6.7). However, these experiments cannot be found in [28].

The remainder of this article is organized as follows. We give the preliminaries of MR in Section 2. Next, Section 3 reviews the related work. Then, Sections 4 and 5 present the details of the *MCMR* and *MCMR+* algorithms, respectively. Section 6.9 presents the case study. In Section 6, we show the results of our experiments. Finally, Section 7 gives our conclusion.

2 PRELIMINARIES

In this section, we introduce the background of MRIR. Table 2 lists the symbols used in this article.

In MRIR, an image database is modelled as a k-NN graph where each node represents an image [13]. Let $G(V, E)$ denote an undirected k-NN graph where V and E are the set of nodes and edges, respectively. Given two nodes u and v, $e(u, v) \in E$ exists only when (i) u is one of the

k-nearest neighbours of v or (ii) v is one of the k-nearest neighbours of u, according to [8, 13]. Thus, the k-NN graph is undirected. Note that the k-NN graph is regarded as an off-the-shelf component in MRIR since it is easy to construct and dynamically maintained [7]. Our exact k-NN graph construction method is discussed in Section 6.8.

MRIR can be formulated as follows: Given a query node $q \in V$, it computes the MR scores of all nodes in the graph w.r.t node q. Let $\mathbf{A} \subset \mathcal{R}^{n\times n}$ be the adjacency matrix of the graph G. Normally, the edge weight can be defined by the heat kernel [8]: $A_{ij} = \exp\{-\mathbf{d}^2(v_i, v_j)/2\sigma^2\}$ if there is an edge linking node v_i with v_j; otherwise, $A_{ij} = 0$, where function $\mathbf{d}(v_i, v_j)$ is the Euclidean distance between v_i and v_j. Node v_i and node v_j are defined in the $\mathcal{L}^p$-space (which are obtained by the feature extraction methods like deep learning approaches), and σ is a hyper-parameter. Note that the sum of each row/column in $\mathbf{A}$ can be smaller or larger than 1. Let $\mathbf{q}$ be an $n \times 1$ column vector of zeros except that the qth entry is set to 1, *i.e.*, $q(q) = 1$. Let $\mathbf{x}_q^*$ denote an $n \times 1$ column exact MR scores vector w.r.t node q where $x_q^*(v)$ denotes the exact MR score of node v w.r.t node q. Thus, the MR score $x_q^*(v)$ measures how similar node v is to node q.

Intuitively, MR can be understood from the perspective of information spreading from the query node in the k-NN graph. Initially, the query node q owns a fixed number of information to be propagated along the edges in the graph. Then, each node in the graph constantly receives the information from its neighbours until the total information held by each node remains unchanged, reaching a stationary state (*i.e.*, convergence). After the spreading process, the "similarity" score of each node $v \in V$ w.r.t node q can be represented as the final information held by v, which is exactly the MR score of node v w.r.t node q. The larger the MR score of node v is, the more similar node node v is to node q. Next, we introduce how to compute MR scores by the information spreading process. Firstly, it symmetrically normalizes the adjacency matrix $\mathbf{A}$ and constructs a new matrix $\mathbf{W}$ such that $\mathbf{W} = \mathbf{C}^{-1/2}\mathbf{A}\mathbf{C}^{-1/2}$ where $\mathbf{C}$ is the diagonal matrix of $\mathbf{A}$ such that $C_{ii} = \sum_{j=1}^{n} A_{ij}$ for each $i \in [1, n]$ while other entries are zeros. To be distinguished from $\mathbf{A}$, $\mathbf{W}$ is called the symmetrically normalized matrix. This normalization is necessary for the convergence of information spreading later. Next, MR scores are computed by iteratively using the following equation until the convergence is reached:

$$\mathbf{x}^{(t+1)} = \alpha \mathbf{W}\mathbf{x}^{(t)} + (1-\alpha)\mathbf{q}, \tag{1}$$

where $\mathbf{x}^{(t)}$ is the MR score vector obtained in the tth iteration and $\mathbf{x}^{(0)} = \mathbf{0}$. In each iteration, each node receives the information from its neighbours (the first term), and also retains its initial information (the second term) [65, 66]. The parameter α specifies the relative amount of the information from its neighbours and itself. This iterative process is denoted as *Power*. When it converges, the exact MR scores $\mathbf{x}_q^*$ of all nodes w.r.t node q are obtained. Besides, it has been shown in [8, 13, 65] that after convergence, the vector $\mathbf{x}_q^*$ holds the following equation:

$$\mathbf{x}_q^* = (1-\alpha)(\mathbf{I} - \alpha\mathbf{W})^{-1}\mathbf{q}. \tag{2}$$

This equation indicates that the exact MR scores $\mathbf{x}_q^*$ can be obtained by computing the inverse of a matrix. As such, a straightforward solution to solve the top-k MRIR search is to firstly compute the MR scores of all nodes w.r.t node q by Equation (2), and then returns the top-k nodes by sorting the MR scores in decreasing order. However, this solution takes $O(n^{2.373})$ time since it takes $O(n^{2.373})$ time to compute the inverse of a matrix, which is expensive when the graphs (*i.e.*, image databases) are large. Thus, a fast solution is essential.

3 RELATED WORK

Manifold Ranking-based Image Retrieval (MRIR). Recently, *FMR* [15], *EMR* [62], *Mogul* [8], and *Mogul-E* [8] were proposed to improve the efficiency of computing MR scores. However, all of them cannot satisfy the three requirements simultaneously mentioned in Section 1. Among them, *Mogul-E* is *exact* method which compute the exact MR scores, and others are *approximate* methods which compute approximate MR scores. All of them can be classified as *matrix-based* methods since their basic idea is to compute the MR scores by solving Equation (2). Specifically, they exploit existing matrix decomposition techniques to construct several pre-computed indices in the preprocessing phase for accelerating the computation of Equation (2) in the query phase. In the preprocessing phase, *FMR* partitions the graph into several blocks and decomposes the matrix by using SVD [15]. *EMR* computes an anchor graph by selecting a set of anchor nodes and decomposes the adjacency matrix by using Woodbury formula [62]. *Mogul* and *Mogul-E* compute and store the decomposed matrices by using *Incomplete Cholesky Factorization* and *Modified Cholesky Factorization*, respectively. Since all the matrix decomposition techniques they used except *Modified Cholesky Factorization* are approximate, the MR scores computed by them except *Mogul-E* are with errors. None of these approximate methods satisfies the requirement of "output bound". Furthermore, all existing algorithms need to construct several special-purpose indexes in the prepossessing phase, and thus, they are not index-free. *Mogul-E* requires $O(n^2)$ preprocessing time and $O(n^2)$ space cost which will introduce a huge overhead. For example, in a dataset with one billion images whose graph size is 300 GB, the index size of *Mogul-E* is expected to be over 6 TB, which cannot be handled by most of servers with their limited memory size. *Mogul* requires $O(n)$ preprocessing time and $O(n)$ space overhead cost. Though *Mogul* has a better prepossessing time and a better space cost than *Mogul-E*, it still needs to build the index. Furthermore, the query time complexities of *Mogul* and *Mogul-E* are $O(n)$ and $O(n^2)$, respectively. Although the time complexity of *Mogul* is small, it does not return accurate results due to its inability of returning an output with an output bound. Besides, *Mogul-E* is very time-consuming due to its quadratic time complexity. For example, in our experimental result, *Mogul-E* could run more than four days in our real *Flick* with only 503,509 images.

Following the previous work [8, 15, 62], there are several possible index-based methods [1, 11, 12], which were originally designed for efficiently computing the inverse of a matrix by decomposing the matrix into sub-matrices where the number of non-zero elements in the matrix are reduced. The basic matrix decomposition methods include LU decomposition [1], QR decomposition [12], Singular Vector decomposition [11] and so on. However, none of them have been applied for solving our MRIR problem yet. Besides, most of them do not outperform the state-of-the-art *Mogul* in terms of efficiency since the number of non-zero elements in their decomposed matrix is more than that in *Mogul*. Thus, we excluded them for comparison in experiments.

Top-*k* Similarity Search. MR is a similarity search method to compute the similarity score of a node w.r.t all other nodes in the graph. In graph-based similarity search area, there exist other two classical methods, *i.e.*, SimRank [18, 20, 25, 31, 53, 58] and PPR [16, 29, 30, 37, 43, 52, 56, 57, 59], both of which consider the local similarity and the global similarity in the graph simultaneously. However, different from MR, these two similarity search methods have different definitions of how to compute the similarity score between two nodes in the graph. Specifically, Simrank is based on two intuitive statements: (1) two nodes are similar if they are referenced by similar nodes in the graph, and (2) a node is most similar to itself [58]. PPR focuses on the relative significance of a target node t w.r.t a source node s in a graph. The PPR value of node t w.r.t node s represents the probability that a random walk from s terminates at t on the graph which is usually unweighted, and thus, the value reveals the original graph topology between node s and node t. Both Simrank and PPR haven't been applied to image retrieval tasks yet. However, a comprehensive study about graph-based similarity measurements has empirically verified that PPR has better performance than Simrank as a similarity measurement on some real-world applications [27]. Besides, from our case studies in Section 6.9, MR outperforms PPR in image retrieval task by achieving higher precision of top-k results. It is because MR considers not only the graph topology between two nodes but also the symmetrically normalized weights on the edges obtained by heat kernel, while PPR considers only the graph topology.

For the top-k similarity search problem, it highly depends on which one similarity measurements are used. For different similarity measurements, the top-k search methods are different since different pruning strategies are exploited. In the area of MR, only *Mogul* [8] and *Mogul-E* [8] were proposed to solve the top-k MR search. Both of them computes the lower and upper bound of MR score of each candidate node that is likely in the top-k results. Based on the bounds computed in each iteration, they prune the nodes that must be not in the top-k results and narrow the size of candidate set. However, as mentioned above, both methods cannot satisfy the three requirements simultaneously. However, our proposed methods, namely *MCMR* and *MCMR+* can satisfy the three requirements simultaneously since we design a non-trivial unbiased weighted random walk sampling strategy for computing MR scores and theoretically proved that our method could answer the top-k MR search problem with guarantees. Moreover, comprehensive experiments verified that our proposed methods are more efficient than *Mogul* and *Mogul-E*.

Message Passing Neural Networks. Due to rapid development of deep learning-based approaches, the idea of *information diffusion* (which is the major idea of MR) has also been adopted to several existing **graph neural networks** (**GNNs**) since it helps to learn the feature representations by considering the graph structures [10, 21, 24, 61]. For comprehensiveness, we discuss these studies below. In 2017, ***Message Passing Neural Network*** (**MPNN**)-based algorithms [10] were proposed to learn the representation of a node in a given graph. Specifically, for each node v in the graph, *MPNN* aggregates the message of all neighbours of node v to node v itself, and then updates the hidden representation of each node v. Although *MPNN*-based algorithms for semi-supervised classification on graphs have recently achieved a great success, these methods only consider passing the messages in a few iterations, which is not enough for learning the feature vectors of nodes based on the graph structure. To solve this issue, Klicpera et al. [21] proposed the **Approximate Personalized Propagation of Neural Predictions** (***APPNP***) by connecting ***Graph Convolutional Network*** (**GCN**) with PPR to learn the high-order information of any node in the graph. However, *APPNP* propagates the information along the edges in the graph by treating each neighbour of a node equally, which is always not true in reality. In Section 6.10, our experimental results show that our proposed method *MCMR+* can achieve higher precision than *APPNP* **by up to 2.9%** for the node classification task. It is because *MCMR+* can treat each neighbour of a node *discriminatively* by assigning a "discriminative" weight to each neighbour in the graph.

4 FIRST ALGORITHM: MCMR

To address the deficiencies of existing algorithms about computing MR scores, we propose an algorithm called the ***Monte Carlo-based Manifold Ranking*** (***MCMR***) algorithm, which is efficient *without index* and can return the *true* top-k nodes with accuracy guarantee. In other words, it satisfies all the three requirements simultaneously. The idea behind *MCMR* is to utilize Random Walk Sampling on the graph (*i.e.*, one of the *Graph Sampling approaches* [26]).

It is well known that graph sampling is a promising paradigm to address the computational challenge of graph analysis tasks since it generates representative samples of the graphs without traversing the whole graph [26]. Among various graph sampling approaches, Random Walk Sampling is the mainstream one due to its scalability and simplicity to implementation. The general idea of Simple Random Walk Sampling strategy is as follows. A single random walk starts at a given node, then repeatedly jumps to another node by choosing from the current node's neighbours *uniformly at random*. After many steps, the probability of a node being visited tends to reach a stationary probability distribution [26].

In addition, we observed that the MR score of a node v w.r.t the query node q (*i.e.*, $x_q^*(v)$) can be regarded as a stationary probability of v being visited by a walk starting from q multiplied by a coefficient (which is decided by the symmetrically normalized matrix $\mathbf{W}$ and the parameter α). It is because $x_q^*(v)$ is obtained by repeatedly receiving the information from its neighbours. Thus, Random Walk Sampling can be a possible solution for our problem.

Challenge. To perform unbiased general Random Walk Sampling on the graph, the whole process can be divided into two phases: (1) *the simulation phase* (which simulates enough number of samples, *i.e.*, random walks), and (2) *the estimation phase* (which performs an *unbiased estimation* on simulated samples). However, it is *non-trivial* and *challenging* to apply Random Walk Sampling to our problem since the scheme of MR is based on the symmetrically normalized matrix $\mathbf{W}$, which is not a stochastic matrix. In particular, the sum of entries in each row in $\mathbf{W}$ is not exactly 1. It means that we cannot simulate the random walks as the general Random Walk Sampling strategy does. Thus, the new simulation and estimation phases are needed.

4.1 Overview

The major idea behind *MCMR* is to simulate *weighted random walks* instead of the simple random walks. Given a query node q, *MCMR* consists of two phases: the *simulation* phase and the *estimation* phase. In particular, in the simulation phase, it simulate a specified number of *weighted random walks* from q. In the estimation phase, it estimates the MR score of each node in the graph w.r.t q based on the simulated weighted random walks.

4.2 The Simulation Phase

Before going into the details, we firstly formally define the weighted random walk in Definition 2.

Definition 2 (Weighted Random Walk). Given a parameter α, and a query node q, a *weighted random walk* is simulated as follows: it starts from node q; then, at each step, it chooses one of the following two options: (i) terminate with $(1-\alpha)$ probability; (ii) with α probability, moves to one neighbour of the current node according to our *transition policy* (to be introduced later).

Note that the expected length of the weighted random walk is $O(\frac{1}{1-\alpha})$ since it terminates with $1-\alpha$ probability. Next, we introduce the *transition policy*. Suppose that the walk is currently at node v. The idea behind the policy is to choose a neighbour of v with *a weighted probability* instead of uniformly at random. Definition 3 gives the details of the transition policy.

Definition 3 (Transition Policy). At each step, the weighted random walk jumps to a neighbour u of the current node v with probability $\frac{W_{v,u}}{D_{vv}}$ where $D_{vv} = \sum_{u\in\mathcal{N}(v)} W_{v,u}$ and $\mathcal{N}(v)$ is the set of neighbours of v.

Here, we say that D_{vv} is the (v,v)-th entry of the diagonal matrix $\mathbf{D}$ of matrix $\mathbf{W}$ such that $D_{ii} = \sum_{j=1}^{n} W_{ij}$ for $i \in [1,n]$ while other entries are zeros. In this way, it is ensured that the sum of the probabilities that node v chooses one of its neighbours as the next node is equal to 1. Besides, we construct an *Alias structure* [48] on the matrix $\mathbf{W}$ so that one neighbour u can be sampled according to probability $\frac{W_{v,u}}{D_{vv}}$ in constant $O(1)$ time. Note that selecting a neighbour u of the current node v with constant time complexity is very important since it ensures that the time complexity of simulating a weighted random walk is bounded by the length of this walk.

Alias structure. The Alias structure [23, 26] is an optimal data structure for the weighted sampling problem. To keep this article self-contained, we present the alias structure below. Algorithm 1 illustrates the pseudocodes of constructing the alias structure for a node v, which takes as input the set $\mathcal{N}(v)$ of neighbours of v and the symmetrically normalized matrix $\mathbf{W}$. Its major idea is to

ALGORITHM 1: Build Alias Structure

```
Input: a node $v$, the set $\mathcal{N}(v)$, the degree $d(v)$ of node $v$, matrix $\mathbf{W}$
Output: The switch probability $p(u)$ and the alias $a(u)$ for each $u \in \mathcal{N}(v)$
1: $p(u) \leftarrow \frac{W_{v,u}}{D_{vv}}$ for each $u \in \mathcal{N}(v)$;
2: $a(u) \leftarrow u$ for each $u \in \mathcal{N}(v)$;
3: Let $\mathcal{B}^l = \{u \in \mathcal{N}(v) | p(u) > \frac{1}{d(v)}\}$ and $\mathcal{B}^s = \{u \in \mathcal{N}(v) | p(u) < \frac{1}{d(v)}\}$;
4: while $\mathcal{B}^l$ is not empty do
5:     Pick any node $u_i \in \mathcal{B}^s$ and $u_j \in \mathcal{B}^l$;
6:     $a(u_i) \leftarrow u_j$;
7:     $p(u_j \leftarrow p(u_j) - (\frac{1}{d(v)} - p(u_i))$;
8:     Remove $u_i$ from $\mathcal{B}^s$;
9:     if $p(u_j) \leq \frac{1}{d(v)}$ then
10:        Remove $u_j$ from $\mathcal{B}^l$;
11:        if $p(u_j) < \frac{1}{d(v)}$ then
12:            Add $u_j$ into $\mathcal{B}^s$;
13: Return $p(u)$ and $a(u)$ for each $u \in \mathcal{N}(v)$
```

compute a *switch probability* $p(u)$ and an *alias* $a(u)$ for each node $u \in \mathcal{N}(v)$. As such, a neighbour u can be selected from $\mathcal{N}(v)$ with constant time complexity by exploiting $p(u)$ and $a(u)$.

Initially, for each node $u \in \mathcal{N}(v)$, its switch probability $p(u)$ is set as $\frac{W_{v,u}}{D_{vv}}$, and its alias $a(u)$ is set as u (Lines 1 and 2). Denote $\mathcal{B}^l$ (resp. $\mathcal{B}^s$) as the set of nodes u whose $p(u) > \frac{1}{d(v)}$ (resp. $p(u) < \frac{1}{d(v)}$) (Line 3). Then, we iteratively modify the switch probability $p(u)$ so that each switch probability could be $\frac{1}{d(v)}$ (Lines 4–12). Specifically, in each iteration, we pick a node u_i from $\mathcal{B}^s$ and a node u_j from $\mathcal{B}^l$, respectively (Line 5). After that, we update the alias of u_i as $a(u_i) = u_j$, and decrease the switch probability of u_j by $(\frac{1}{d(v)} - p(u_i))$, referred as $p(u_j) = p(u_j) - (\frac{1}{d(v)} - p(u_i))$ (Lines 6 and 7). As u_i is processed, we remove u_i from $\mathcal{B}^s$ since the switch probability of u_i has been "filled" to be $\frac{1}{d(v)}$ (which is unnecessary to be updated) (Line 8). Besides, if $p(u_j) \leq \frac{1}{d(v)}$, node u_j should be removed from $\mathcal{B}^l$, and furthermore, if $p(u_j) < \frac{1}{d(v)}$, we add u_j into $\mathcal{B}^s$ since the switch probability of u_j should be filled in to be $\frac{1}{d(v)}$ (Lines 9–12). The process continues until the set $\mathcal{B}^l$ is empty. Finally, we have $p(u)$ and $a(u)$ for each node $u \in \mathcal{N}(v)$ as the results. Since each node in $\mathcal{N}(v)$ is inspected only once, the time complexity of this construction algorithm is $O(d(v))$. We construct such alias structure for each node $v \in V$ so that we can sample a neighbour of any node v in $O(1)$ time (to be introduced later). Thus, the total time complexity of this construction algorithm for a graph is $O(n)$ since the number of edges in a k-NN graph is $O(n)$.

To select a node u from $\mathcal{N}(v)$ for a node $v \in V$ with the computed alias structure, we first uniformly sample a node u' with probability $\frac{1}{d(v)}$. Then, a probability threshold $z \in [0, 1]$ is generated by random If $z \leq p(u')$, then we have u' as the final sampled node; otherwise, we have the alias of u' (*i.e.*, $a(u')$) as the final sampled node. It is easy to see that selecting a sampled node takes $O(1)$ time.

4.3 The Estimation Phase

Now, we present how to estimate the MR score of each node $v \in V$ w.r.t the query node q in the estimation phase.

One limitation of using the transition policy defined in Definition 3 in the simulation phase is that the information from a node v to a node u could not be captured *appropriately*. Specifically, during the information spreading process defined by Equation (1), at each iteration, the

ALGORITHM 2: MCMR

```
Input: Graph $G = (V, E)$, query node $q$, matrices $\mathbf{W}$ and $\mathbf{D}$, constant parameter $\alpha$, number of random walks $n_r$
Output: Estimated MR score $\pi(q, v)$ for each $v \in V$
1: $\pi(q, v) \leftarrow 0$,$S(q, v) \leftarrow 0$ for each $v \in V$;
2: for $i$ from 1 to $n_r$ do
3:     $S_i \leftarrow 1$; $v \leftarrow q$;
4:     while $rand() > 1 - \alpha$ do
5:         Pick one neighbor $u$ of $v$ with the probability $\frac{W_{v,u}}{D_{vv}}$;
6:         $S_i \leftarrow S_i \cdot D_{vv}$; $v \leftarrow u$;
7:     $S(q, v) \leftarrow S(q, v) + S_i$;
8: $\pi(q, v) \leftarrow S(q, v)/n_r$ for each $v \in V$;
```

information that v transfers to its neighbour u should be $W_{v,u}$, instead of $\frac{W_{v,u}}{D_{vv}}$ in our transition policy. The basic idea of our estimation phase is to compute u's information *appropriately* at each step of a weighted random walk by using some derivations involving $W_{v,u}$ (instead of $\frac{W_{v,u}}{D_{vv}}$). In addition, we regard the MR score of a node v w.r.t q as the "expected" amount of information that node v obtains from node q (via the weighted random walks). Let $\pi(q, v)$ be the estimated MR score of a node $v \in V$ obtained by *MCMR*. Let n_r denote the number of weighted random walks simulated. Suppose that the ith weighted random walk $\{X_l\}_{0<l\leq L}$ of length L starts from the query node q. First, we define the *scalar* of the ith walk for each $v \in V$, denoted as $S_i(v)$, to be the amount of the information that the ith walk should transfer to node v as follows:

$$S_i(v) = \begin{cases} \prod_{l=1}^{L-1} D_{X_l X_l}, \text{if the } i\text{-th walk ends at } v, \\ 0, \text{otherwise,} \end{cases} \tag{3}$$

where $i \in \{1, 2, \dots, n_r\}$ and X_l is the lth node in this walk. Next, we define the total scalars of all walks for each $v \in V$, denoted by $S(q, v)$, to be $\sum_{i=1}^{n_r} S_i(v)$. Finally, the estimated MR score $\pi(q, v)$ of node v is computed as follows: $\pi(q, v) = \frac{S(q,v)}{n_r}$. The correctness of *MCMR* is proved in Theorem 1.

4.4 Implementation Details of MCMR

Algorithm 2 gives the pseudocode of our *MCMR* algorithm. It firstly initializes the estimated MR score $\pi(q, v)$ and the sum of scalars $S(q, v)$ to 0 for each node $v \in V$ (Line 1). Then, it generates n_r random walks as follows: for the ith walk, it initializes the scalar of this walk S_i as 1 (Line 3); at each step, it uniformly generates a random value; if the value is greater than $1 - \alpha$, it picks up one neighbour u of current node v with probability $\frac{W_{v,u}}{D_{vv}}$ and multiplies S_i by D_{vv} (Lines 4–6); otherwise, the walk terminates at v, it adds the scalar of this walk S_i to $S(q, v)$ (Line 7). After the simulation phase finishes, the estimated MR score $\pi(q, v)$ is computed as $\frac{S(q,v)}{n_r}$ for each node $v \in V$ (Line 8).

4.5 Theoretical Analysis

In this section, we first prove that the estimated MR scores obtained from *MCMR* are unbiased as shown in Theorem 1 and then give a time complexity analysis of *MCMR*.

Theorem 1. *Let $\pi(q, v)$ be the estimated MR score by Algorithm 2. We have $E[\pi(q, v)] = x_q^*(v)$.*

Proof. Firstly, we have $E[\pi(q, v)] = E[\frac{S(q,v)}{n_r}] = \frac{E[\sum_{i=1}^{n_r} S_i(v)]}{n_r} = E[S_i(v)]$. Next, we prove that $E[S_i(v)]$ is the expected amount of the information of a weighted random walk from q propagating to node v by considering all possible walks from q, which is exactly the MR score $x_q^*(v)$.

Now, consider the ith random walk $\{X_l\}_{0<l\leq L}$ of length L that starts from the query node q, *i.e.*, $X_1 = q$, and terminates at node v, *i.e.*, $X_L = v$. As defined, the scalar of this walk (*i.e.*, $S_i(v)$) is the amount of the information that this walk should transfer to v. That is, $S_i(v) = \prod_{l=1}^{L-1} D_{X_l X_l}$, where X_l is the lth node in this walk. To distinguish different values of length L, we denote $S_i^L(v)$ to be equal to $S_i(v)$. Next, let $Pr_i^L(v)$ denote the probability of the ith random walk of length L that terminates at v. From the simulation phase of *MCMR*, at the lth step of this walk where $0 < l < L$, node X_l moves to one of its neighbours X_{l+1} with $\alpha \cdot \frac{W_{X_l,X_{l+1}}}{D_{X_l X_l}}$ probability. At the Lth step, it terminates with $1-\alpha$ probability and ends at node v. Therefore, we can compute the probability that this walk terminates at node v as follows: $Pr_i^L(v) = (1-\alpha)\alpha^{L-1}\prod_{l=1}^{L-1}\frac{W_{X_l,X_{l+1}}}{D_{X_l X_l}}$. If $L = 1$, it indicates that this walk terminates at the source node at the first step. By considering all possible walks with different length L where $L \in \{1, 2, \ldots, \infty\}$, the expected amount of the information that this walk should transfer to node v, *i.e.*, $E[S_i(v)]$, is computed as follows:

$$\begin{aligned} E[S_i(v)] &= \sum_{L=1}^{\infty} Pr_i^L(v) \cdot S_i^L(v) \\ &= (1-\alpha)\sum_{L=1}^{\infty}\left(\alpha^{L-1}\prod_{l=1}^{L-1}\left(\frac{W_{X_l,X_{l+1}}}{D_{X_l X_l}} \cdot D_{X_l X_l}\right)\right) \\ &= (1-\alpha)\sum_{L=1}^{\infty}\left(\alpha^{L-1}\prod_{l=1}^{L-1} W_{X_l,X_{l+1}}\right). \end{aligned}$$

Now, we would like to show that for this walk, the value of $\prod_{l=1}^{L-1} W_{X_l,X_{l+1}}$ can be computed by using the multiplication of the matrix $\mathbf{W}$ and the vector $\mathbf{q}$. According to the graph theory [45], the Lth power of matrix $\mathbf{W}$ is a matrix where each entry (u, v) indicates the amount of the information that node u propagates to node v by a path of length $(L+1)$. So, the result of $\mathbf{W}^L\mathbf{q}$ is a vector where the vth entry is the amount of the information that node q propagates to node v by a path of length $(L+1)$. Since the value of $\prod_{l=1}^{L-1} W_{X_l,X_{l+1}}$ is the amount of the information that the query node q (*i.e.*, the first node) propagates to node v (*i.e.*, the last node of this walk) by a path of length L, it is easy to get that $\prod_{l=1}^{L-1} W_{X_l X_{l+1}} = [\mathbf{W}^{L-1}\mathbf{q}](v)$, where $[\cdot](v)$ stands for the vth entry of this vector in square bracket. Thus, the following equation can be obtained:

$$\begin{aligned} E[S_i(v)] &= (1-\alpha)\sum_{L=1}^{\infty}[\alpha^{L-1}\mathbf{W}^{L-1}\mathbf{q}](v) \\ &= (1-\alpha)\left[\left\{\sum_{L=0}^{\infty}(\alpha\mathbf{W})^L\right\}\mathbf{q}\right](v). \end{aligned}$$

As shown in [37], [9] and [65], if $\lim_{l\to\infty}(\alpha\mathbf{W})^L = \mathbf{0}$,

$$(\mathbf{I}-\alpha\mathbf{W})^{-1} = \mathbf{I} + \alpha\mathbf{W} + \alpha^2\mathbf{W}^2 + \cdots = \sum_{L=0}^{\infty}(\alpha\mathbf{W})^L. \tag{4}$$

By letting $(\alpha\mathbf{W})^0 = \mathbf{I}$. Thus, we have the following equations:

$$E[S_i(v)] = (1-\alpha)[(\mathbf{I}-\alpha\mathbf{W})^{-1}\mathbf{q}](v) = x_q^*(v).$$

The proof completes. □

Time complexity. We give the time complexity of *MCMR* for solving the top-k search. The straightforward way is to firstly compute the estimated MR scores $\pi(q, v)$ of each node v w.r.t q by using Algorithm 2, and then returns the top-k nodes with the highest estimated scores. By using *Bernstein Inequality* [35], it is easy to prove that if generating $\frac{10\ln(1/p_{fail})}{3\cdot\sqrt{2k}\cdot(gap_k)^2}$ random walks, Algorithm 2 returns the exact top-k set with high probability, as shown in Theorem 2. This theorem indicates that *MCMR* satisfies the output-bound requirement.

Theorem 2. *Given a graph $G(V,E)$, a query node q, a failure probability p_{fail}, and the number of random walks $n_r = \frac{10\ln(1/p_{fail})}{3\cdot\sqrt{2k}\cdot(gap_k)^2}$, Algorithm returns the top-k set V_k such that for any node u in V_k, if $x_q^*(u) - x_q^*(v_{k+1}) \geq gap_k$, with probability at least $1 - p_{fail}$, the following holds:*

$$\pi(q,u) - \pi(q,v_{k+1}) > 0,$$

where v_{k+1} is node whose exact MR score is the $(k+1)$-th largest and $gap_k = x_q^*(v_k) - x_q^*(v_{k+1})$.

Proof. For the ith walk simulated by Algorithm 2, let Z_i be the following random variable: $Z_i = S_i(u)$, if this walk ends at u, $Z_i = -S_i(v_{k+1})$, if this walk ends at node v_{k+1}, and $Z_i = 0$ otherwise. Then from Theorem 1, we can know that $E[Z_i] = E[S_i(u)] - E[S_i(v_{k+1})] = x_q^*(u) - x_q^*(v_{k+1}) \geq gap_k > 0$. Estimating MR scores from n_r random walks, the event of interchanging u and v_{k+1} in the rankings in the final top-k set is equivalent to taking n_r independent Z_i variables and having $\sum_{i=0}^{n_r} Z_i < 0$. The probability of this event can be upper bounded by using Bernstein's inequality [35] and the fact that $Var(Z_i) = x_q^*(u)+x_q^*(v_{k+1})-(x_q^*(u)-x_q^*(v_{k+1}))^2 \leq x_q^*(u)+x_q^*(v_{k+1})$. The derivation is as follows:

$$Pr\{\pi(q,u) - \pi(q,v_{k+1}) < 0\} = Pr\left\{\frac{1}{n_r}\sum_{i=0}^{n_r} Z_i < 0\right\}$$

$$\leq \exp\left(-n_r \frac{(E[Z_i])^2}{2Var(Z_i) + 4/3E[Z_i]}\right) \leq \exp\left(-n_r \frac{\left(x_q^*(u) - x_q^*(v_{k+1})\right)^2}{2\left(x_q^*(u) + x_q^*(v_{k+1})\right) + 4/3\left(x_q^*(u) - x_q^*(v_{k+1})\right)}\right)$$

$$\leq \exp\left(-n_r \frac{\left(x_q^*(u) - x_q^*(v_{k+1})\right)^2}{10/3x_q^*(u) + 2/3x_q^*(v_{k+1})}\right).$$

In addition, for any $v \in V$, $x_q^*(v) < \sqrt{C_{vv}}$ (see Equation (6) in [66]). Recall that matrix $\mathbf{C}$ is the diagonal matrix of matrix $\mathbf{A}$. Let C_{max} denote the largest value in matrix $\mathbf{C}$. In the following, we would like to prove that $C_{max} < 2k$. From the definition of MRIR, each entry in matrix $\mathbf{A}$ is smaller than 1. Besides, for each node in the undirected k-NN graph, it has at most $2k$ neighbours. Thus, $C_{max} < 2k$ and $x_q^*(v) < \sqrt{2k}$ for any node $v \in V$.

So, we can drive as follows:

$$\begin{aligned}
&Pr\{\pi(q,u) - \pi(q,v_{k+1}) < 0\}\\
&\leq \exp\left(-0.3\cdot\sqrt{2k}\cdot n_r\left(x_q^*(u) - x_q^*(v_{k+1})\right)^2\right)\\
&= \exp\left(-0.3\cdot\sqrt{2k}\cdot\frac{10\ln(1/p_{fail})}{3\cdot\sqrt{2k}\cdot(gap_k)^2}\left(x_q^*(u) - x_q^*(v_{k+1})\right)^2\right)\\
&\leq \exp\left(-0.3\cdot\sqrt{2k}\cdot\frac{10\ln(1/p_{fail})}{3\cdot\sqrt{2k}\cdot(gap_k)^2}(gap_k)^2\right) \leq p_{fail}.
\end{aligned}$$

The proof completes. □

Since the expected length of a single walk is $\frac{1}{1-\alpha}$, we have the time complexity as shown in Theorem 3.

Theorem 3. *Let* $n_r \geq \frac{10\ln(1/p_{fail})}{3\cdot\sqrt{2k}\cdot(gap_k)^2}$ *and* $p_{fail} = \frac{1}{n}$, MCMR *returns the top-k set in* $O(\frac{\ln(n)}{(1-\alpha)\cdot\sqrt{k}\cdot(gap_k)^2})$ *time with* $1-\frac{1}{n}$ *probability.*

Since gap_k is not pre-known, in our experiments, we set a parameter c for tune the number of walks to be simulated, *i.e.*, $n_r = c\cdot\frac{10\ln(1/p_{fail})}{3}$. Obviously, the more random walks are simulated, the longer query time of *MCMR* and the higher the precision of top-k results. In other words, *MCMR* can balance the tradeoff between efficiency and accuracy. Besides, if Chernoff Inequality (see Lemma 1) is applied, it is easy to show that the time complexity of *MCMR* is $O(n\log n)$ which is independent of gap_k, as shown in Theorem 4.

Lemma 1 (Chernoff Inequality [32, 35]). *Let* $Y_1,\ldots,Y_i$ *be the i.i.d random variables such that* $Y_i\in[0,1]$. *Let* $Y=\sum_{i=1}^{n}Y_i$. *The following inequality holds:*

$$Pr\{|Y-E[Y]|\leq\epsilon\cdot E[Y]\}\leq 2\exp\left(-\frac{\epsilon^2\cdot E[Y]}{3}\right), \tag{5}$$

where ϵ *is a real value such that* $\epsilon\in(0,1)$.

Theorem 4. *Given a graph* $G(V,E)$, *a query node* q, *a failure probability* p_{fail}, *and the number of random walks* $n_r=\frac{3\cdot n\cdot\log(2n)}{\epsilon^2}$, *Algorithm returns the top-k set* V_k *such that for any node* u *in* V_k, *if* $x_q^*(v)>\frac{1}{n}$, *then the following holds:*

$$(1-\epsilon)\cdot x_q^*(v)\geq\pi(q,v)\leq(1+\epsilon)\cdot x_q^*(v),$$

with probability at least $1-\frac{1}{n}$, *where* ϵ *is a user-specified error such that* $\epsilon\in(0,1)$.

Proof. As shown in Algorithm 2, we will average over $n_r=\frac{3\cdot n\cdot\log(2n)}{\epsilon^2}$ weighted random walks. We denote v as the last node of the ith walk. Let $Y_i=S_i(v)$. Besides, from Algorithm 2, we can know that the estimated MR score $\pi(q,v)$ of node v is equal to $\frac{1}{n_r}\cdot\sum_{i=1}^{n_r}Y_i$. Now, we start our proofs based on these definitions. Firstly, from Theorem 5, we have that $E[\pi(q,v)]=E[Y_i]=x_q^*(v)>\frac{1}{n}$ where the last inequality is the given condition. Next, we define $Y=\sum_{i=1}^{n_r}Y_i$. Since $E[Y_i]>\frac{1}{n}$, we have $E[Y]>\frac{n_r}{n}$. Now, we show concentration of the estimated MR score by applying Lemma 1:

$$\begin{aligned}&Pr\{|\pi(q,v)-x_q^*(v)|\geq\epsilon\cdot x_q^*(v)\}\\&=Pr\{|\bar{Y}-E[Y_i]|\leq\epsilon\cdot E[Y_i]\}=Pr\{|Y-E[Y]|\leq\epsilon\cdot E[Y]\}\\&<2\exp\left(-\frac{\epsilon^2\cdot E[Y]}{3}\right)<2\exp\left(-\frac{\epsilon^2\cdot n_r}{3\cdot n}\right)=2\exp\left(-\frac{\epsilon^2\cdot 3\cdot n\cdot\log(2n)}{3\cdot n\cdot\epsilon^2}\right)=\frac{1}{n}.\end{aligned}$$

Thus, the proof completes. □

Based on Theorem 4, we can derive that the time complexity of *MCMR* is $O(n\log n)$ since the user-specified error ϵ is generally set to be 0.5. Thus, *MCMR* satisfies the efficiency requirement.

In summary, as elaborated above, *MCMR* satisfies all three requirements.

5 EFFICIENT TOP-K SEARCH ALGORITHM: MCMR+

Although *MCMR* has shown its ability to balance the tradeoff between efficiency and accuracy, it wastes time to estimate the MR scores of a large number of unnecessary nodes who are not in the top-k set. To address this deficiency, we propose a more efficient algorithm called *MCMR+*, which combines *Local Search* (to be introduced later) and *MCMR* in a non-trivial way.

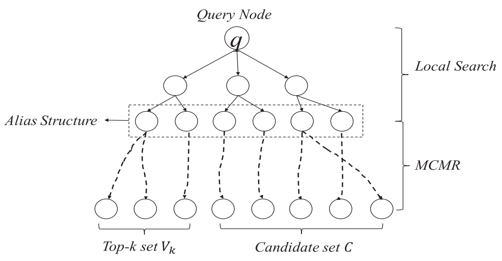


Fig. 2. Illustration of MCMR+ algorithm.

5.1 Overview

MCMR+ is an algorithm by utilizing the incremental refinement for efficiently finding the top-k nodes. Specifically, *MCMR+* contains a number of iterations, and in each iteration, it has two phases: the *local search* phase and the *prune* phase, as illustrated in Figure 2. In particular, the local search phase performs *Local Search* from the query node without traversing the whole graph so that only some "special" nodes need to simulate the weighted random walks (*i.e.*, the nodes in the dotted rectangle in Figure 2). Then, the prune phase simulates a number of weighted random walks from these special nodes by using *MCMR*, and then estimates the lower and upper bounds to prune the nodes who are definitely not in the top-k set, called *the non-top-k nodes*, to avoid unnecessary computations. After each iteration, all nodes in the graph would be categorized into three sets: (i) the top-k set, denoted as V_k, which contains the nodes whose MR scores are definitely the top-k largest; (ii) the non-top-k set, which contains the non-top-k nodes which are definitely not in the top-k set; (ii) the candidate set, denoted as C, which contains the nodes which are probably in the top-k set. As a whole, *MCMR+* gets a finer top-k set after each iteration until the exact top-k set is found.

5.2 Local Search

Before going into the details of *MCMR+*, we present the *Local Search* algorithm in this section. As a whole, the *Local Search* algorithm can be regarded as an efficient variant of *Power*. Firstly, we show that *Power* can be interpreted from a slightly different perspective. Specifically, for any node $v \in V$, the MR score of v w.r.t q consists of two parts: (1) the amount of the information reaching v as the result of the propagation starting from the query q along the edges in the graph; and (2) the amount of the information to be retained (only affecting the case where $v = q$). Thus, we can write that

$$x_q^*(v) = \sum_{u \in \mathcal{N}(q)} \alpha \cdot W_{q,u} \cdot x_u^*(v) + \begin{cases} (1-\alpha), & \text{if } v = q, \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

where the first term indicates that the amount of the information reaching v from the query q is equal to the amount of the information received by each neighbour u of q from the query q (*i.e.*, $\alpha \cdot W_{q,u}$) multiplied by the amount of the information that v can receive from u (*i.e.*, $x_u^*(v)$). Similarly, the MR score of v w.r.t each u, can be further represented as follows:

$$x_u^*(v) = \sum_{w \in \mathcal{N}(u)} \alpha \cdot W_{u,w} \cdot x_w^*(v) + \begin{cases} (1-\alpha) & \text{if } v = u \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

By recursively applying this, at the end, $x_q^*(v)$ can be represented by the sum of $x_u^*(v)$ for each $u \in V$ with different coefficients and a constant related to the query q, as shown in the following equation:

$$x_q^*(v) = \sum_{u\in V} r^f(q,u)x_u^*(v) + \pi^f(q,v), \tag{8}$$

where $r^f(q,u)$ is the coefficient part for $x_u^*(v)$, denoted by *residue*, and $\pi^f(q,v)$ is the constant related to q, denoted by *reserve*. Besides, the reserve $\pi^f(q,v)$ can be regarded as the amount of the information that has been propagated to node v from q, while the residue $r^f(q,v)$ as the amount of the information that has been propagated to u from q but has not been transferred to v from u. Equation (8) is the main intuition of *Local Search* (see Algorithm 3). The algorithm maintains $\pi^f(q,v)$ and $r^f(q,v)$ for each node $v \in V$. Initially, it starts with $\pi^f(q,v) = 0$ and $r^f(q,v) = 0$ for all v, while $r^f(q,q) = 1$ (Line 1–2). Then, it performs a series of *push operations* by adjusting the reserves and residues based on Equation (6). The algorithm recursively conducts a *push operation* on the node v whose residue exceeds a residue threshold $r^f_{max} \cdot d(v)$ where $d(v)$ is the degree of node v. The algorithm terminates when no such node v could be found. Finally, it returns the reserve and residue of each node $v \in V$ w.r.t the query node q.

Time complexity. Now, we show that the time cost of *Local Search* in Lemma 2.

Lemma 2. *Let r^f_{max} be the residue threshold. The total cost of Local Search is $O(\frac{1}{(1-\alpha)\cdot r^f_{max}})$.*

Proof. The time cost of *Local Search* is bounded by the total number of push operations. Firstly, we prove that the total number of push operations conducted at any node v can be bounded by $\frac{x_q^*(v)}{r^f_{max}}$. For each push operation at a node v, the algorithm transfers $(1-\alpha)$ portion from v's residue to v' reserve, which is at least $(1-\alpha)\cdot r^f_{max}\cdot d(v)$ since $r^f(q,v)$ must be at least $r^f_{max}\cdot d(v)$. Thus, it follows that the total number of push operations on v is bounded by $\frac{x_q^*(v)}{(1-\alpha)\cdot r^f_{max}\cdot d(v)}$ since the total reserve of v is exactly the MR score of v w.r.t q. Note that each push operation visits all neighbours of v, and thus, the cost of all push operations on node v is bounded by $\frac{x_q^*(v)\cdot d(v)}{(1-\alpha)\cdot r^f_{max}\cdot d(v)}$. Summing up over all $v \in V$ and we have the total time cost is at most $\sum_{v\in V}\frac{x_q^*(v)\cdot d(v)}{(1-\alpha)\cdot r^f_{max}\cdot d(v)}$. Since the matrix $\mathbf{W}$ used by MR is symmetrically normalized such that $\sum_{v\in V} x_q^*(v)$ is O(1) [66], the total time cost is $O(\frac{1}{(1-\alpha)\cdot r^f_{max}})$. □

5.3 MCMR+

As stated in Section 4, *MCMR* estimates the MR scores of all nodes w.r.t a query node q by simulating a number of random walks, which is very expensive since the number of weighted random walks simulated is large to provide a required accuracy guarantee. To reduce the number of random walks required for guaranteeing the accuracy of results, *MCMR+* uses a new approach to estimate the MR scores by incorporating the *Local Search* algorithm and *MCMR* in a non-trivial way.

The idea behind *MCMR+* is to apply Equation (8) since this equation demonstrates the relationship between $x_q^*(v)$ and $x_u^*(v)$ of node v w.r.t each node u in the graph. However, it is time-consuming to compute the exact MR score $x_u^*(v)$ of node v w.r.t each node u in the graph. To speed up the computation, *MCMR+* estimates a rough approximation $\pi^o(u,v)$ of MR score $x_u^*(v)$ for each $u \in V$ by exploiting *MCMR*. Thus, given a query node q, for any node $v \in V$, *MCMR+* computes the estimated MR score $\pi(q,v)$ of node v w.r.t node q by applying the following equation:

$$\pi(q,v) = \pi^f(q,v) + \sum_{u\in V} r^f(q,u)\cdot\pi^o(u,v). \tag{9}$$

By exploiting *Local Search*, *MCMR+* needs to simulate the weighted random walks from only the nodes whose residue is non-zero, instead of all nodes in the graph, leading to high efficiency.

ALGORITHM 3: Local Search

```
Input: Graph $G = (V, E)$, query node $q$, matrix $\mathbf{W}$, constant parameter $\alpha$, threshold $\pi^f_{max}$
Output: Residue $r^f(q, v)$ and reserve $\pi^f(q, v)$ for each $v \in V$
1: $r^f(q, q) \leftarrow 1$ and $r^f(q, v) \leftarrow 0$ for all $v \neq q$;
2: $\pi^f(q, v) \leftarrow 0$ for all $v$;
3: while $\exists v \in V$ such that $r^f(q, v) \geq r^f_{max} \cdot d(v)$ do
4:     for each $u \in \mathcal{N}(v)$ do
5:         $r^f(q, u) \leftarrow r^f(q, u) + \alpha \cdot W_{v,u} \cdot r^f(q, v)$;
6:     $\pi^f(q, v) \leftarrow \pi^f(q, v) + (1 - \alpha) \cdot r^f(q, v)$;
```

Besides, to obtain the top-k nodes with the largest MR scores w.r.t a query node, a straightforward solution is similar to *MCMR* which estimates the MR scores of all nodes and after that, finds out the top-k nodes, which is very time-consuming since it takes huge amount of time cost for the unnecessary computations of the nodes which are not in the top-k nodes. To avoid this deficiency, *MCMR+* adopts an iterative refinement strategy to compute the estimated MR scores, and the lower/upper bounds of the exact MR score of each node w.r.t the query node q, based on which the *non-top-k* nodes are pruned iteratively.

As stated in Section 5.1, *MCMR+* contains two phases: the *Local Search* phase and the *Prune* phase. Algorithm 4 gives the pseudocode of *MCMR+*. Given a k-NN graph $G(V, E)$, a query node q, the matrix $\mathbf{W}$, the constant parameters α and k, *MCMR+* finds the exact top-k node set of q with high probability. Initially, it sets the top-k set $V_k = \emptyset$ and the candidate set $C = V$ (Line 1). Next, it iteratively starts the *Local Search* phase by invoking Algorithm 3 and the *Prune* phase by invoking the *Prune* algorithm (to be introduced later) to refine C and to construct V_k. It first sets the initial number n_r of weighted random walks to be simulated as $\log n$ (which is small so that the random walks can be simulated quickly) and set the initial residue threshold r^f_{max} as $\frac{1}{m}$ (which is large so that *Local Search* can terminate quickly) where m is the number of edges in the graph, n_r and r^f_{max} are the parameters used in *Prune* phase and the *Local Search* phase, respectively (Line 2). In each iteration, it firstly performs *Local Search* with r^f_{max} which returns $\pi^f(q, v)$ and $r^f(q, v)$ for each node $v \in V$ (Line 4). Then, it calls the *Prune* algorithm to perform Random Walk Sampling (by using *MCMR*) and update candidate set C and the top-k set V_k (Line 5). If the number of nodes in V_k is still smaller than k, it halves r^f_{max} and doubles the number of random walks n_r so that the estimated MR scores of candidate nodes could be more "accurate" (Line 6).

5.4 Prune Algorithm

The *Prune* algorithm updates the candidate set C and top-k set V_k by estimating the lower and upper bounds of each node in C by applying the *empirical Bernstein Inequality*, as shown in the following lemma.

Lemma 3 (Empirical Bernstein Inequality [2]). *Let $X_1, \ldots, X_{n_r}$ be real-valued i.i.d random variables, such that $X_i \in [0, r]$, $E[X_i]=\mu$ and $Var[X_i]=\sigma^2$. Let $\bar{X}_{n_r} = \frac{1}{n_r}\sum_{i=1}^{n_r} X_i$ denote the empirical mean, and $\bar{\sigma}^2=\frac{1}{n_r}\sum_{i=1}^{n_r}(X_i - \bar{X}_{n_r})^2$. With probability $1 - p_f$, we have*

$$|\bar{X}_{n_r} - \mu| \leq \sqrt{\frac{2\bar{\sigma}^2 \log(3/p_f)}{n_r}} + \frac{3r\log(3/p_f)}{n_r} = \beta.$$

Based on Lemma 3, the lower and upper bounds for each $v \in V$ can be estimated. Let $\pi(q, v)$ be the estimator of $x^*_q(v)$ by simulating n_r walks, and $\bar{\sigma}^2(q, v)$ be the empirical variance. We define

ALGORITHM 4: MCMR+

```
Input: Graph $G = (V, E)$, query node $q$, matrix $\mathbf{W}$, constant parameters $\alpha$ and $k$
Output: $V_k$, the set of the exact top-$k$ nodes
1: $V_k \leftarrow \emptyset, C \leftarrow V$;
2: $n_r \leftarrow \log n, r^f_{max} \leftarrow \frac{1}{m}$;
3: while $|V_k| < k$ do
4:     $[r^f, \pi^f] \leftarrow$Local Search$(r^f_{max})$;
5:     $[V_k, C] \leftarrow$Prune$(r^f, \pi^f, V_k, C)$;
6:     $n_r \leftarrow 2n_r, r^f_{max} \leftarrow \frac{r^f_{max}}{2}$;
7: Return $V_k$;
```

$$\beta(q,v) = \sqrt{\frac{2\bar{\sigma}^2(q,v)\log(3n)}{n_r}} + \frac{3r^f_{sum}\log(3n)}{n_r}. \tag{10}$$

where $r^f_{sum} = \sum_{v \in V} r^f(q,v)$. Next, we define $\pi(q,v) - \beta(q,v)$ and $\pi(q,v) + \beta(q,v)$ as the lower and upper bounds of node v, respectively.

Algorithm 5 gives the pseudocode of *Prune* algorithm. It starts by performing n_r random walks. For each random walk, it samples a node u according to the probability $r^f(q,u)/r^f_{sum}$ in $O(1)$ time (Line 2), and performs the weighted random walks $(1-\alpha)$ from u (Line 4–7). If the random walk terminates at node v, it updates the estimator $\pi(q,v)$ and empirical variance $\bar{\sigma}^2(q,v)$ (Line 8 and 9). After all random walks are processed, it computes the final estimator $\pi(q,v)$ (Line 11) and $\beta(q,v)$ for each $v \in C$ (Line 12). Finally, it updates the top-k node set V_k and the candidate set C as follows. For each node $v \in C$, it counts the number of nodes $v' \in C$ with the upper bounds $\pi(q,v') + \beta(q,v')$ that are equal to or smaller than the lower bound $\pi(q,v) - \beta(q,v)$ of node v (Line 13). If this number exceeds $|C| + |V_k| - k$, it implies that with high probability, there are more than $n-k$ nodes whose optimal MR scores are lower than that of v, and thus, v is a true top-k node. In this case, we move v from C to V_k (Line 14). Besides, it counts the number of nodes v' in C with lower bounds $\pi(q,v') - \beta(q,v')$ that are higher or equal to the upper bound $\pi(q,v) + \beta(q,v)$ of node v (Line 15). If this number exceeds $k - |V_k|$, it implies that with high probability, there are more than k nodes whose optimal MR scores are higher than that of v, and thus, v is unable to be a top-v node. In this case, v is removed from C (Line 16).

5.5 Correctness of MCMR+

In this section, we show the correctness *MCMR+*. Based on the Bernstein Inequality [2] (*i.e.*, Lemma 3), we prove that *MCMR+* computes all lower and upper bounds correctly with high probability.

Lemma 4. *With probability* $1 - 1/n$, *at any iteration of* MCMR+, *we have* $x^*_q(v) \in [\pi(q,v) - \beta(q,v), \pi(q,v) + \beta(q,v)]$ *for any* $v \in V$.

It is easy to understand this lemma since we set the failure probability p_f in Lemma 3 to be $\frac{1}{n}$ and use the fact that each independent random variable $X_i \in [0, r^f_{sum}]$. From Lemma 4, we can derive Theorem 5 which indicates that *MCMR+* satisfies the output-bound requirement.

Theorem 5. *Given a query node q and parameter k,* MCMR+ *returns the exact top-k nodes set with probability* $1 - 1/n$.

Proof. We prove the lemma by induction. Assume that at the end of the $(i-1)$-th iteration, all nodes in V_k are top-k nodes, and all nodes in $V \backslash (C \cup V_k)$ are the non-top-k nodes. At the ith iteration, recall that *MCMR+* moves a node v from C to V_k if the number of $v' \in C$ such that

ALGORITHM 5: Prune

```
Input: Graph $G = (V, E)$, query node $q$, constant parameters $\alpha$ and $k$, forward reserves $\pi^f$, Alias structure
    of forward residues $r^f$, candidate set $C$, top-$k$ set $V_k$, number of random samples $n_r$
Output: Updated candidate set $C$ and top-$k$ set $V_k$
 1: for $j$ from 1 to $n_r$ do
 2:   Sample a node $u$ from Alias structure $r^f$ with probability $r^f(q,u)/r^f_{sum}$ and set $v \leftarrow u$;
 3:   $S_j \leftarrow 1$; //scalar of the $j$-th walk
 4:   while $rand() > 1 - \alpha$ do
 5:     $S_j \leftarrow S_j \cdot D_{vv}$;
 6:     Uniformly pick $w$ with probability $W_{v,w}/D_{vv}$;
 7:     $v \leftarrow w$;
 8:   $\pi(q,v) \leftarrow \frac{j-1}{j}\pi(q,v) + \frac{1}{j}r^f_{sum} \cdot S_j$;
 9:   $\bar{\sigma}^2(q,v) \leftarrow \frac{j-1}{j}\bar{\sigma}^2(q,v) + \frac{1}{j}(r^f_{sum} \cdot S_j)^2$
10: for each node $v \in C$ do
11:   $\pi(q,v) \leftarrow \pi(q,v) + \pi^f(q,v)$;
12:   compute $\beta(q,v)$ by Equation (10);
13:   if Number of $v' \in C$ such that $\pi(q,v') + \beta(q,v') < \pi(q,v) - \beta(q,v)$ exceeds $|C| + |V_k| - k$ then
14:     $V_k \leftarrow |V|_k \cap \{v\}, C \leftarrow C - \{v\}$;
15:   else if Number of $v' \in C$ such that $\pi(q,v') - \beta(q,v') > \pi(q,v) + \beta(q,v)$ exceeds $k - |V_k|$ then
16:     $C \leftarrow C - \{v\}$;
```

$\pi(q,v') + \beta(q,v') \leq \pi(q,v) - \beta(q,v)$ exceeds $|C| + |V_k| - k$. By Lemma 4.2, all lower and upper bounds are correctly computed by *MCMR+*, and thus, the number v such that $\pi(q,v') \leq \pi(q,v)$ exceeds $|C|+|V_k|-k$. This proves that there are less than k nodes in V_k and C with MR scores larger than $\pi(q,v)$. Since by the induction hypothesis, nodes outside C and V_k are non-top-k nodes, it follows that the top-k nodes are either in C or V_k, and thus, v is also a top-k node. Similarly, we can prove that if *MCMR+* removes a node v from C, then there are at least k nodes with MR scores larger or equal to $\pi(q,v)$ and thus v is a non-top-k node. Hence, the induction holds and the theorem follows. □

5.6 Time Complexity Analysis

Each iteration of *MCMR+* takes $O(\frac{1}{(1-\alpha)\cdot r^f_{max}})$ time for the local search phase and $O(n_1 \cdot n_r)$ time for the pruning phase where n_1 is the number of nodes that can simulate random walks. Recall that at the ith iteration of *MCMR+*, we have $r^f_{max} = \frac{1}{2^i m}$ and $n_r = 2^i \log n$. Besides, in each iteration, we can reuse the results obtained in the previous iteration such as $r^f(q,v)$ for each $v \in V$. So, the total time cost of *MCMR+* is $O(2^T \cdot m + 2^{T+1} \cdot n_1 \log n)$ where T is the number of iterations. Since the number of edges in the k-NN graph is $O(n)$ and n_1 is at most n, the time complexity of *MCMR+* is $O(2^{T+1}(n \log n))$. Our experiments show that T is always small and can be omitted since our lower and upper bounds are tight. Thus, the time complexity of *MCMR+* can be simplified as $O(n \log n)$, satisfying the efficiency requirement.

In summary, as elaborated above, *MCMR+* satisfies all the three requirements.

6 EXPERIMENTS

In this section, we performed experiments to evaluate the effectiveness and efficiency of *MCMR* and *MCMR+*, all of which were conducted on a machine with Intel(R) Xeon(R) E5-2650 @ 2.2GHz CPU and 500GB memory. We implemented our algorithms in C++.

Table 3. Datasets (K = 10^3)

Dataset	Name	k-NN	n	m	T (*Power*)
COIL	C5	5	7.2K	58.5K	50
	C10	10		111.1K	
	C15	15		164.9K	
	C20	20		216.7K	
Pub-Fig	P5	5	57.2K	500.2K	55
	P10	10		936.3K	
	P15	15		1.4M	
	P20	20		1.8M	
NUS-WIDE	N5	5	269.7K	2.2M	1000
	N10	10		4.3M	
	N15	15		6.4M	
	N20	20		8.5M	
Flickr	F5	5	503.5K	4.0M	1000
	F10	10		7.9M	
	F15	15		11.7M	
	F20	20		15.3M	

6.1 Experimental Setting

Methods. We compared our two approaches with three algorithms: *Mogul* [8], *Mogul-E* [8], and *Power* [66]. Note that we did not include other approximate algorithms, *i.e.*, *EMR* and *FMR*, since *Mogul* outperforms them in terms of both efficiency and accuracy as shown in [8]. Since the source codes of *Mogul* and *Mogul-E* are not publicly available, we implemented them strictly following the pseudocode in [8]. All of our source codes are publicly published. We set $\alpha = 0.99$, following previous work [13, 65, 66]. For *MCMR*, we set n_r as $10^3 \cdot \frac{10 \log n}{3}$ for each dataset.

Datasets. We used the four images datasets following [8]:

(1) COIL [36]: It contains 7,190 images taken in Columbia University. We use SIFT algorithm [33] to obtain the feature vector of each image.
(2) Pub-Fig [23]: It has 57,244 images of 200 famous persons, each of which has 73 attributes [23]. Each attribute itself is a semantic description of images relevant to human faces; they were automatically detected by pre-trained classifiers.
(3) NUS-WIDE [46]: It consists of 269,648 images, each of which has 144 attributes.[23]. This dataset was collected by randomly downloading photographs from Flickr through its public API. The dataset was created by a research group of National University of Singapore.
(4) Flickr [17]: It has 503,510 images. We use SIFT algorithm [33] to obtain the feature vector of each image.

We constructed the k-NN graph for each dataset. Table 3 shows the statistics of each dataset. For each dataset, in order to obtain *quickly* the k-NN graph for a given value k in the query phase, we *pre-constructed* a 500-NN graph for each dataset which contains 500 nearest neighbours for each image (or node) in the dataset. Here, we chose 500 as the number of nearest neighbours used in the constructed graph since the top-k search in real-world applications generally assumes that $k \leq 500$ [8, 13, 56, 57, 59]. More details about constructing k-NN graph online could be found in Section 6.8. Besides, following the previous work in [8], we used k from the set of $\{5, 10, 15, 20\}$.

Accuracy metric. For each dataset, we randomly selected 50 query nodes. The ground truth for each dataset is obtained by using *Power* which stops when the absolute error $|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}|$ dropped below 10^{-10} (see Table 3 for the number of iterations T required). Note that we cannot directly compute the matrix inverse to obtain the ground truth since it runs out of memory for the large datasets. We evaluated the accuracy of the top-k search results obtained by each method by using the classic metrics: Precision P@$k = \frac{|V_k \cap V_k'|}{k}$, where V_k is the ground-truth top-k set and $V_k' = \{v_1', \ldots, v_k'\}$ is the approximate top-k set returned by the method to be evaluated.

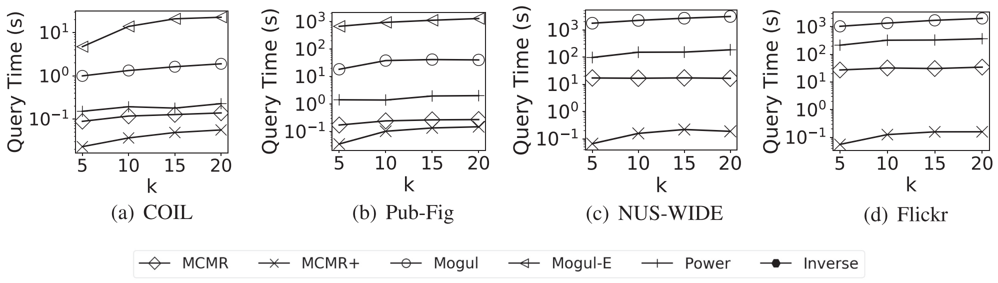


Fig. 3. Query time (in seconds) of the top-k query on different datasets.

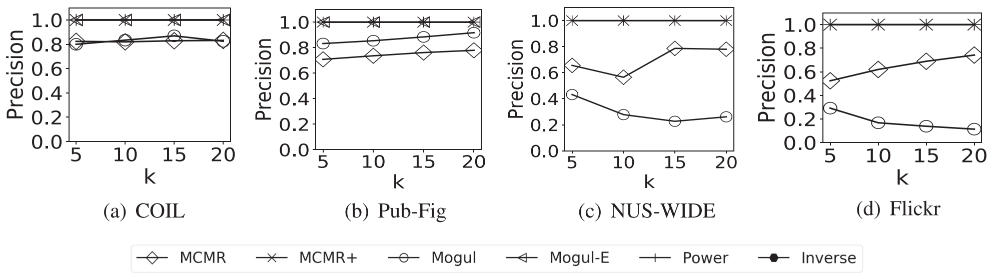


Fig. 4. Precision of the top-k query on different datasets.

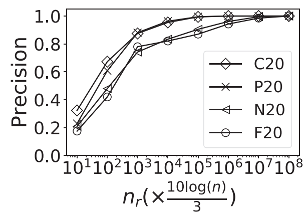
6.2 Efficiency of MCMR and MCMR+

In this section, we evaluated the efficiency of our proposed methods in the query phase. For each dataset, the average query time of each method is plotted in Figure 3. Note that we did not include *Mogul-E* in Figure 3(c) and Figure 3(d) because the query time of *Mogul-E* on both datasets is very large (*i.e.*, more than four days). From Figure 3, we can see that *MCMR+* and *MCMR* are faster than *Mogul* and *Mogul-E* by 1 to 4 orders of magnitude, satisfying the efficiency requirement. It is because that *Mogul* and *Mogul-E* need to compute the approximate MR scores for almost all the nodes in the graph while ours do not. *MCMR* is slightly slower than *MCMR+* because *MCMR+* can prune the non-top-k nodes to avoid unnecessary computations.

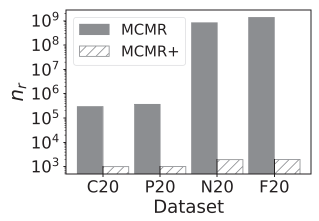
6.3 Precision of MCMR and MCMR+

In this section, we evaluated the quality of the top-k results obtained by each method. Over 50 queries, the average precision of each method is shown in Figure 4. The results show that *MCMR+* can return the exact top-k results on all datasets, satisfying the output-bound requirement. Although *Mogul-E* and *Power* can return the exact top-k results, their query time is slower than *MCMR+* by up to four orders of magnitude. Besides, *MCMR+* dominates the fastest approximate method *Mogul* on all datasets as well. Besides, on the largest dataset Flickr, the precision of the state-of-the-art *Mogul* is below 0.3. It is because *Mogul* directly sets some entries in the factorized matrices to zero for achieving high efficiency, making the estimated MR scores be approximate with high error. The result quality of *Mogul* for top-k search deteriorate when the dataset becomes larger since the number of "bad" entries in the factorized matrices increases accordingly.

In addition, as an approximate method, the precision of *MCMR* outperforms the existing best-known approximate method *Mogul* on most of datasets. Although *Mogul* can achieve higher precision on Pub-Fig than *MCMR*, its performance is not stable, for example, *Mogul* achieves with lower



(a) Precision of *MCMR* varying the number of walks n_r



(b) The number of walks n_r to achieve precision 1

Fig. 5. Fair comparison of *MCMR* and *MCMR+* for the top-k query (where k is set as 20).

precision on some larger datasets, e.g., NUS-WIDE and Flickr, than *MCMR*. Finally, the results also show that *MCMR+* has higher precision than *MCMR* on all datasets. It is because the number of weighted random walks simulated by *MCMR* is not enough.

6.4 Fair Comparison of MCMR and MCMR+

In this section, we examined the performance of *MCMR* in terms of precision for the top-k query by varying the number of weighted random walks n_r. Besides, we examined the effectiveness of *MCMR+* compared with *MCMR* to see how *MCMR+* reduces the number of random walks required for achieving precision 1 for the top-k query. We included four datasets for comparison and set k to be 20 for each dataset. Thus, we have four graphs, namely C20, P20, N20, and F20, where C20 denotes the k-NN graph of dataset COIL when k is 20, and P20, N20, and F20 have similar meanings. Furthermore, for each dataset, we chose 50 query nodes uniformly at random and reported the average precision. The results are shown in Figure 5. Specifically, Figure 5(a) shows the precision of *MCMR* for top-k query by varying n_r. In this figure, the legend shows four graphs. The results show that the precision of *MCMR* increases when n_r increases on all datasets. In particular, the precision of *MCMR* achieves 1 when n_r is large enough. For example, for datasets COIL and Pub-Fig (*i.e.*, C20 and P20, respectively), *MCMR* returns the 100% correct top-k nodes when $n_r \geq 10^5 \cdot \frac{10\log(n)}{3}$, while for datasets NUS-WIDE and Flickr (*i.e.*, N20 and F20, respectively), *MCMR* returns the true top-k results when $n_r \geq 10^7 \cdot \frac{10\log(n)}{3}$, which is consistent with Theorem 2.

Figure 5(b) shows the number of weighted random walks used by *MCMR* and *MCMR+* to achieve precision 1 for the top-k query on all datasets. Specifically, *MCMR+* generates fewer random walks than *MCMR* by up to six orders of magnitude. It is because *MCMR+* reduces the number of random walks by utilizing *Local Search* and the lower/upper bounds to avoid unnecessary computations on the nodes which are not in the top-k results.

6.5 Parameter Sensitivity

In this section, we evaluated the effect of the number of weighted random walks simulated by *MCMR*. We varies the number n_r of walks for the top-k search by tuning the coefficient value of n_r from $\{10^1, 10^2, 10^3, 10^4, 10^5, 10^6\}$. The results of dataset COIL are illustrated in Figure 6. We can see that the precision increases when n_r increases, which is consistent with Theorem 2. Besides, when n_r is not large enough (e.g., 10^1), the precision of top-k search decreases when k increases. It is because the value of gap_k decreases when k increases. From Theorem 2, the smaller the value of gap_k, the more random walks required to guarantee the quality of top-k results. Thus, for the same number of random walks, the larger the value of k, the smaller the value of gap_k, and so the lower the precision of top-k search. Besides, when n_r approaches 10^6, the precision of *MCMR* reaches 1 for all top-k queries.

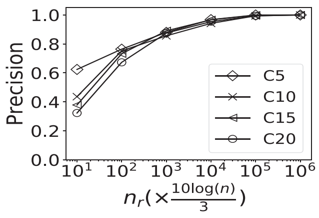


Fig. 6. Precision of *MCMR* on COIL dataset varying the number of random walks n_r.

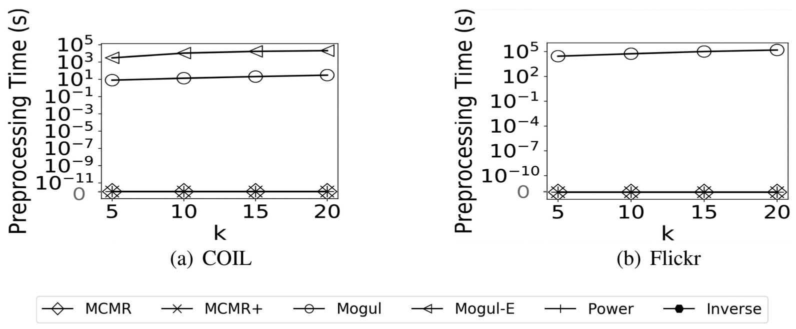


Fig. 7. Preprocessing time (in seconds) of each method on two datasets. Mogul-E is not plotted in Flickr since it ran over four days.

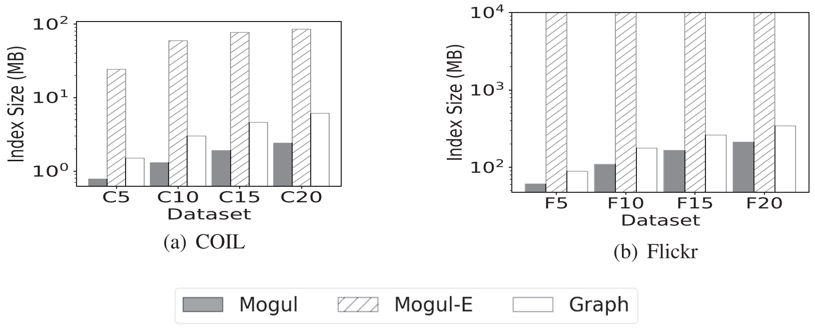
6.6 Preprocessing Cost for Dynamic Updating

In the following, we would like to conduct experiments on dynamic datasets to study the performance of our proposed algorithms.

For datasets COIL and Flickr, we removed 50 randomly selected nodes as well as their linking edges from the graph to measure the preprocessing cost of dynamic updates: (1) the preprocessing time and (2) the index size. The reconstruction time of k-NN graph for all methods are the same and so not reported here (more details about dynamic k-NN graph construction could be found in Section 6.8). The preprocessing time of each method is plotted in Figure 7. Since our proposed methods *MCMR+* and *MCMR* satisfy the index-free requirement, it requires zero time cost in the preprocessing phase. The same case for *Power*. However, without any mechanism supporting dynamic updates for *Mogul* and *Mogul-E*, we have to rebuild the indices from scratch with highly expensive preprocessing cost. For example, *Mogul* takes nearly 100,000 seconds to rebuild the index for Flickr, and the rebuilding must be carried out repeatedly with updates, which is completely time-consuming in the dynamic settings. However, our proposed methods can be easily adapted to any dynamic update without extra cost. Besides, the index size of *Mogul* and *Mogul-E* are illustrated in Figure 8. To present how large of their indices, the graph size is plotted as well, denoted by Graph. The results show that the index size of *Mogul-E* is larger than the graph size by up to 20 times, which is not practical for real-world applications.

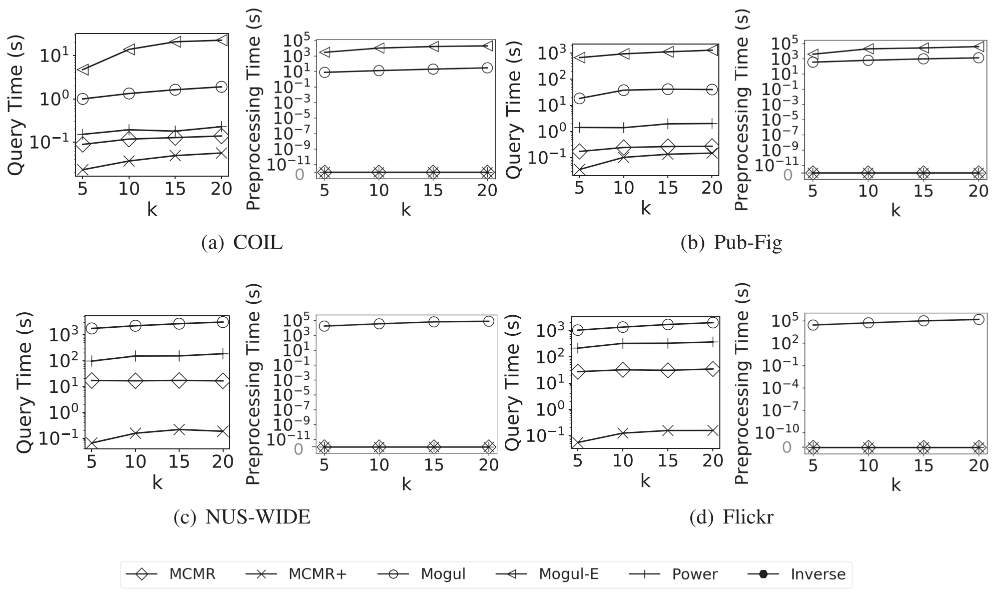
6.7 Query Time for Out-of-Sample Queries

In the previous sections, we assumed that the query node is in the database by following the previous work [8, 13]. However, in real applications, the user can select a query node outside the database (the out-of-sample query). Our approach for the out-of-sample query follows the previous work [13]. By following the previous work [8, 13], our approach is to first project the query node



(a) COIL (b) Flickr

Fig. 8. Index size of each method on each dataset. Mogul-E is not plotted in Flickr since it ran over four days.



(a) COIL (b) Pub-Fig

(c) NUS-WIDE (d) Flickr

Fig. 9. Query time and preprocessing time (in seconds) for out-of-sample query on four datasets.

in the out-of-sample query to a node in the feature space. Next, it connects the query with its k-nearest neighbours in the image database, and calculates the symmetrically normalized matrix $\mathbf{W}$. Thirdly, we add one row and one column to the symmetrically normalized matrix $\mathbf{W}$, with each element equal to the corresponding edge weight. More details about constructing a new k-NN graph for a newly inserted image could be found in Section 6.8. All the other operations are performed similarly using this enlarged matrix $\mathbf{W}$. Figure 9 shows the query time of each method for an out-of-sample query on four datasets. For *Mogul* and *Mogul-E*, their indices need to be rebuilt from scratch for the out-of-sample query. The results show that our proposed method can work well and solve the out-of-sample query by taking less query time than the existing algorithms.

6.8 Exact k-NN Graph Construction

In this section, we present how to build a k-NN graph for a given value of k in the query phase. Recall that existing work about MR [8, 13, 15, 47, 55] requires exact k-NN graph for computing

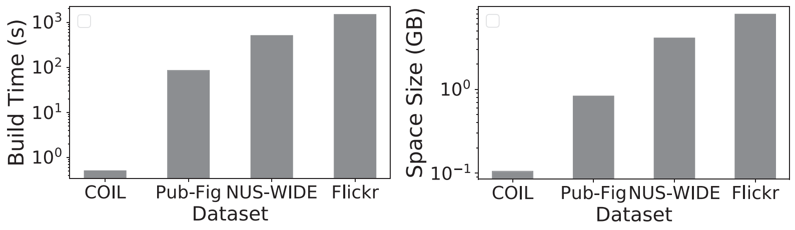


Fig. 10. Build time and the space size of a large 500-NN graph on four datasets.

the MR scores w.r.t a given query. The (naive) greedy method for constructing a k-NN graph for a dataset containing n images (*i.e.*, nodes) takes $O(n^2)$ query time by computing distance of each pair of images (*i.e.*, nodes) in the dataset, which is very time-consuming for large image datasets. Recall that by providing the k-NN graph in advance, our proposed algorithms could return the top-k nodes with the highest MR score w.r.t a query node in only $O(n \log n)$ query time. Thus, to efficiently obtain the k-NN graph in the query phase, we include our construction method below.

Based on the learnt feature vectors of images in the dataset, our construction method has two phases: (1) the preprocessing phase, and (2) the query phase (for computing MR scores w.r.t a query node). In the preprocessing phase, it constructs and stores a large 500-NN graph which contains 500 nearest neighbours of each image in each dataset by exploiting CoverTree [4] which is a widely used structures for high-dimensional nearest neighbours search. Here, we choose 500 as the number of nearest neighbours used in the constructed graph since the top-k search in real-world applications generally assumes that $k \leq 500$ [56, 57, 59]. Thus, our construction method creates a cover tree for a dataset in $O(c^6 n \log n)$ time where $c \in (1.3, 3)$ is a user parameter for splitting [3] and also creates the 500-NN graph in $O(c^{12} n \log n)$ time since it takes $O(c^{12} \log n)$ time to find the nearest neighbours for a node in the graph using CoverTree [4] and there are n nodes required. In addition, storing 500-NN graph and the cover tree takes only $O(n)$ space size. Figure 10 shows the build time and the space size of our construction method on four datasets.

In the query phase, there are two cases: (1) the query image is in the dataset, and (2) the query image is out-of-sample. For the first case, we could directly use the large 500-NN graph with the given value of k and so it does not require any time for building a k-NN graph. For the second case, for a query image q which is out-of-sample, it is required to find its k nearest neighbours and its reverse nearest neighbours (which contain q in their k-NN set). Following previous work [8, 13], the out-of-sample query image will not be put into the image dataset for the top-k image retrieval search for other query images. Thus, it is no need to update the 500-NN graphs and the cover tree in the indexing since this query image is just for one use and is no need to be inserted. Motivated by this, in the query phase, we directly find out its k nearest neighbours and reverse nearest neighbours by taking $O(n)$ time, which is smaller than the query time of our proposed methods (excluding the k-NN graph construction time). Thus, the total time complexities of our proposed methods are still $O(n \log n)$. Besides, in Section 6.7, we can see that in the query phase, our proposed method *MCMR+* runs faster than all existing algorithms.

Besides, in our case study of image retrieval, we show the breakdown of time cost of our proposed method in terms of the preprocessing phase and the query phase. The results are shown in Table 6. From the results, we can see that the build time for constructing the 500-NN graph for dataset *Roxford5k* is much smaller than the training time for the deep learning model *MAC* by up to five orders of magnitude. Thus, the preprocessing time for constructing the 500-NN graph is acceptable in the real-world applications.

Table 4. Details of Each Method Used in Case Study

Method	VGG16	ResNet101	MAC	PPR (MAC)	WPPR (MAC)	MRIR (MAC)
Feature extraction model	VGG16	ResNet101	MAC	MAC	MAC	MAC
Similarity measure	Inner Product	Inner Product	Inner Product	PPR	WPPR	MRIR
Reference	[44]	[14]	[41]	[41] [59]	Adaption	ours

6.9 Case Study of Image Retrieval

In this section, we evaluated the superiority of *MRIR* over the existing similarity measures in terms of effectiveness by a case study.

Methods. For this case study, six methods were compared and the details of each method are shown in Table 4. Specifically, for the top-k search, each method contains two phases: (i) feature extraction and (ii) similarity measure. For most of existing methods, they apply *Inner Product* as similarity measure to return the top-k results after the feature extraction, e.g., **VGG16** [44] (a traditional CNN network with 16 layers), **ResNet101** [14] (a widely used powerful CNN network with 101 layers), and **MAC** [41] (the state-of-the-art CNN network for the image retrieval tasks). Besides, **MRIR(MAC)** is our methods which returns the top-k results by using our proposed approach *MCMR+* (since MCMR+ can empirically retrieve the exact top-k results with the highest exact MR scores). To better illustrate the effectiveness of MRIR, we include PPR and WPPR as alternative similarity measures and implement two adaptive methods, namely, **PPR(MAC)** and **WPPR(MAC)**, respectively. PPR is a popular graph-based similarity measure used for link prediction and friend-recommendation in social media. For WPPR, its transition probability for random walk sampling is the same as MRIR(MAC) for fair comparison. The latter three methods extract the feature vectors by MAC.

Datasets. Following the state-of-the-art of image retrieval [40, 41], we used *Roxford5k* for evaluating, which contains 5,063 images of different buildings in Oxford.

Implementation issues. We conducted the experiments for six methods by taking complete fair principles. Specifically, during the process of feature extraction, we used the same whitening and multi-scale representation as in [41], and used **generalized-mean** (**GeM**) pooling method. Moreover, since *PPR* and *WPPR* are graph-based similarity measures, we use the same k-NN graph as *MRIR* to ensure fair comparison.

Accuracy metric. In this study, our focus is : *Whether or not the top-k results retrieved by* MRIR *really have the same objects as the query?* We denote $\mathcal{P}@k$ as the precision, which measures the percentage of how many images who have the same object as the query are returned as the top-k results. Besides, to show the performance of each method on a dataset where the number of images with the same object are not distributed evenly, we included *Mean Average Precision* as another metric, denoted by $MAP@k$. Specifically, $MAP@k$ for a set of queries is the mean of the *average precision score* for a query image q, denoted by $AvgP(q)$, such that

$$MAP@k = \frac{\sum_{i=1}^{Q} AvgP(q_i)}{Q},$$

where q_i is the ith query image, Q is the number of query images, and $AvgP(q_i) = \frac{\sum_{i=1}^{k}(\mathcal{P}@i)\cdot true(i)}{k}$ where $\mathcal{P}@i$ is the precision for the top-i queries as defined above and $true(i)$ is an indicator function where it is equal to 1 if the ith image returned by a method has the same object as the query image, 0 otherwise.

Results. The results in terms of both $\mathcal{P}@k$ and $MAP@k$ are shown in Table 5. We can see that *MRIR(MAC)* has the best performance in terms of both $\mathcal{P}@k$ and $MAP@k$. Specifically, by using the same feature extraction method, *MRIR(MAC)* has higher $\mathcal{P}@k$ than *MAC*, *PPR(MAC),* and *WPPR(MAC)* by up to 12%, indicating that *MRIR* is a more effective similarity measure than

Table 5. Performance of Each Method on Dataset *Roxford5k* in Case Study

Method	VGG16	ResNet101	MAC	PPR (MAC)	WPPR (MAC)	MRIR (MAC)
$\mathcal{P}@k$	0.750	0.791	0.852	0.894	0.907	**0.93**
$MAP@k$	0.621	0.637	0.7883	0.7964	0.854	**0.897**

Table 6. Time Cost of MRIR(MAC) in Case Study

Dataset	Preprocessing Phase			Query Phase
Roxford5k	MAC model training time (s)	Time for obtaining feature vectors (s)	Time for building 500-NN graph (s)	Query time (s)
	28545.31	5418.16	0.241	0.02128

Table 7. Datasets for Node Classification

Dataset	Number of nodes	Number of edges	Number of nodes for training
CiteSeer	2,110	3,668	1,900
Pubmed	19,717	44,324	1,900
MS-Academic	18,333	81,894	6,000

Table 8. Results of *MCMR+* and *APPNP* for Node Classification

	CiteSeer	*Pubmed*	*MS-Academic*
APPNP	53.7%	75.4%	86.9%
MCMR+	**54.5%**	**78.3%**	**87.5%**

Inner Product, *PPR* and *WPPR*. The reasons are presented in Section 1. Besides, both *VGG16* and *ResNet101* have lower $\mathcal{P}@k$ than *MRIR(MAC)* due to two reasons: (1) the feature extraction methods they used are not superior to *MAC* and (2) the similarity measure they used only focuses on the locally nearest neighbours in the feature space. Similar results could be found in terms of $MAP@k$ where *MRIR(MAC)* has the highest $MAP@k$ among all methods, indicating that *MRIR(MAC)* is more effective to return the top-k results with correct orderings. To sum up, *MRIR* can further improve the quality of results with powerful feature extraction methods.

6.10 MCMR+ for Node Classification

This section evaluated the performance of *MCMR+* and *APPNP* [21] for the node classification task on a graph since both of them adopt the idea of information diffusion. Following previous work [21], we used three datasets for evaluation, namely *CiteSeer*, *Pubmed*, and *MS-Academic*, as shown in Table 7. All datasets except *MS-Academic* are citation graphs and use a bag-of-words representation of the papers' abstracts as features, and *MS-Academic* is a co-authorship graph where edges denote the co-authorships. Given a graph and a set of nodes which are with class labels, node classification aims to predict the class label of each unlabelled node in the graph. Besides, each dataset is split into two sets of nodes: (i) the training set for training the model, and (ii) the test set for evaluating the final performance of the trained model. Table 7 shows the number of training nodes for each dataset. To adapt *MCMR+* for node classification, we firstly use the same GCN structure as *APPNP* to learn the hidden representations of nodes in the graph, and then propagate the learned hidden representations of nodes by using *MCMR+* (instead of PPR in *APPNP*). For fair comparison, the value of propagation probability α is set to 0.01, and for the setting of other parameters, we strictly follow [21]. In addition, we use the metric *precision* to evaluate the performance of each method, *i.e.*, the ratio of the number of nodes with correct label over all testing nodes.

Table 8 shows the results of *MCMR+* and *APPNP*. The results show that *MCMR+* achieves higher precision than *APPNP* on all datasets. Specifically, on datasets *Pubmed* and *Citeseer*, the precision of *MCMR+* is higher than *APPNP* by up to 2.9%. It is because *MCMR+* pays more attention on the

k-nearest neighbours of each node and adopts a "discriminative" weight on each edge for propagating the information but *APPNP* considers each neighbour of a node equally.

7 CONCLUSION

To efficiently address the top-k MRIR search, this article proposes two novel approaches by exploiting the basic idea of Random Walk Sampling. We theoretically proved the correctness of our new weighted Random Walk Sampling strategy used in *MCMR* and *MCMR+* as well as their time complexity for answer the top-k MRIR search. Comprehensive experiments show that our proposed method *MCMR+* dominates the state-of-the-arts in terms of query time, accuracy, and space cost.

REFERENCES

[1] S. Abbasbandy, R. Ezzati, and A. Jafarian. 2006. LU decomposition method for solving fuzzy system of linear equations. *Applied Mathematics and Computation* 172, 1 (2006), 633–643.

[2] J. Y. Audibert, R. Munos, and C. Szepesvári. 2007. Tuning bandit algorithms in stochastic environments. In *Proceedings of the International Conference on Algorithmic Learning Theory.*

[3] S. Bai, X. Bai, and Q. Tian. 2017. Scalable person re-identification on supervised smoothed manifold. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

[4] A. Beygelzimer, S. Kakade, and J. Langford. 2006. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning.*

[5] R. Datta, D. Joshi, J. Li, and J. Z. Wang. 2008. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys* 40, 2 (2008), 1–60.

[6] R. Datta, J. Li, and J. Z. Wang. 2005. Content-based image retrieval: Approaches and trends of the new age. In *Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval.*

[7] W. Dong, C. Moses, and K. Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th International Conference on World Wide Web.*

[8] Y. Fujiwara, G. Irie, S. Kuroyama, and M. Onizuka. 2014. Scaling manifold ranking based image retrieval. *Proceedings of the VLDB Endowment* 8, 4 (2014), 341–352.

[9] Y. Fujiwara, M. Nakatsuji, H. Shiokawa, T. Mishima, and M. Onizuka. 2013. Efficient ad-hoc search for personalized PageRank. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data.* ACM.

[10] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70.* JMLR. org.

[11] G. H. Golub and C. Reinsch. 1971. Singular value decomposition and least squares solutions. In *Linear Algebra.* F. L. Bauer (ed.), Springer, Berlin, 134–151.

[12] C. R. Goodall. 1993. 13 computation using the QR decomposition. (1993).

[13] J. He, M. Li, H. J. Zhang, and H. Tong. 2004. Manifold-ranking based image retrieval. In *Proceedings of the 12th Annual ACM International Conference on Multimedia.*

[14] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

[15] R. He, Y. Zhu, and W. Zhan. 2009. Fast manifold-ranking for content-based image retrieval. In *Proceedings of the 2009 ISECS International Colloquium on Computing, Communication, Control, and Management.*

[16] Guanhao Hou, Xingguang Chen, Sibo Wang, and Zhewei Wei. 2021. Massively parallel algorithms for personalized PageRank. *Proceedings of the VLDB Endowment* 14, 9 (2021), 1668–1680.

[17] M. J. Huiskes and M. S. Lew. 2008. The MIR flickr retrieval evaluation. In *Proceedings of the 1st ACM International Conference on Multimedia Information Retrieval.* ACM.

[18] Glen Jeh and Jennifer Widom. 2002. Simrank: A measure of structural-context similarity. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 538–543.

[19] M. Jian, L. Wu, X. Zhang, and Y. He. 2018. Manifold ranking-based kernel propagation for saliency estimation. In *Proceedings of the 2018 4th International Conference on Control, Automation and Robotics.* IEEE.

[20] M. Jiang, A. W.-C. Fu, and R. C.-W. Wong. 2017. READS: A random walk approach for efficient and accurate dynamic SimRank. *Proceedings of the VLDB Endowment* 10, 9 (2017), 937–948.

[21] J. Klicpera, A. Bojchevski, and S. Günnemann. 2019. Predict then propagate: Graph neural networks meet personalized PageRank.In *Proceedings of the International Conference on Learning Representations.*

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems.*

[23] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. 2009. Attribute and simile classifiers for face verification. In *Proceedings of the 2009 IEEE 12th International Conference on Computer Vision*. IEEE.

[24] J. Lanchantin, A. Sekhon, and Y. Qi. 2019. Neural message passing for multi-label classification. arXiv:1904.08049. Retrieved from https://arxiv.org/abs/1904.08049.

[25] P. Lee, L. V. S. Lakshmanan, and J. X. Yu. 2012. On top-k structural similarity search. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*. IEEE, 774–785.

[26] Y. Li, Z. Wu, S. Lin, H. Xie, Min Lv, Y. Xu, and J. C. S. Lui. 2019. Walking with perception: Efficient random walk sampling via common neighbor awareness. In *Proceedings of the 2019 IEEE 35th International Conference on Data Engineering*.

[27] D. Liben-Nowell and J. Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology* 58, 7 (2007), 1019–1031.

[28] D. Lin, V. J. Wei, and R. C.-W. Wong. 2019. First index-free manifold ranking-based image retrieval with output bound. In *Proceedings of the 2019 IEEE International Conference on Data Mining*. IEEE, 1216–1221.

[29] D. Lin, R. C.-W. Wong, M. Xie, and V. J. Wei. 2020. Index-free approach with theoretical guarantee for efficient random walk with restart query. In *Proceedings of the 2020 IEEE 36th International Conference on Data Engineering*. IEEE, 913–924.

[30] W. Lin. 2019. Distributed algorithms for fully personalized PageRank on large graphs. In *Proceedings of the World Wide Web Conference*. 1084–1094.

[31] Y. Liu, B. Zheng, X. He, Z. Wei, X. Xiao, K. Zheng, and J. Lu. 2017. Probesim: Scalable single-source and top-k simrank computations on dynamic graphs. *Proceedings of the VLDB Endowment* 11, 1 (2017), 14–26.

[32] P. Lofgren, S. Banerjee, and A. Goel. 2016. Personalized PageRank estimation and search: A bidirectional approach. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*. ACM.

[33] D. G. Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60, 2 (2004), 91–110.

[34] C. C. Loy, C. Liu, and S. Gong. 2013. Person re-identification by manifold ranking. In *Proceedings of the 2013 IEEE International Conference on Image Processing*.

[35] R. Motwani and P. Raghavan. 2010. Chap. Randomized algorithms.

[36] S. Nayar, S. Nene, and H. Murase. 1996. *Columbia Object Image Library (COIL 100)*. Technical Report CUCS-006-96. Department of Comp. Science, Columbia University.

[37] L. Page, S. Brin, R. Motwani, and T. Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report. Stanford InfoLab.

[38] W. Qi, M. M. Cheng, A. Borji, H. Lu, and L. F. Bai. 2015. SaliencyRank: Two-stage manifold ranking for salient object detection. *Computational Visual Media* 1, 4 (2015), 309–320.

[39] R. Quan, J. Han, D. Zhang, and F. Nie. 2016. Object co-segmentation via graph optimized-flexible manifold ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[40] F. Radenović, G. Tolias, and O. Chum. 2016. CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples. In *Proceedings of the European Conference on Computer Vision*. Springer.

[41] F. Radenović, G. Tolias, and O. Chum. 2018. Fine-tuning CNN image retrieval with no human annotation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 7 (2018), 1655–1668.

[42] L. Shen, D. Cao, Q. Xu, X. Huang, N. Xiao, and Y. Liang. 2016. A novel local manifold-ranking based K-NN for modeling the regression between bioactivity and molecular descriptors. *Chemometrics and Intelligent Laboratory Systems* 151 (2016), 71–77.

[43] K. Shin, J. Jung, S. Lee, 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*. ACM.

[44] K. Simonyan and A. Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556. Retrieved from https://arxiv.org/abs/1409.1556.

[45] S. Skiena. 1991. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Addison-Wesley Longman Publishing Co., Inc.

[46] J. Song, Y. Yang, Y. Yang, Z. Huang, and H. T. Shen. 2013. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM.

[47] D. Tao, J. Cheng, M. Song, and X. Lin. 2016. Manifold ranking-based matrix factorization for saliency detection. *IEEE Transactions on Neural Networks and Learning Systems* 27, 6 (2016), 1122–1134.

[48] A. J. Walker. 1974. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters* 8, 10 (1974), 127–128.

[49] J. Wan, D. Wang, S. Chu Hong Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li. 2014. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22nd ACM International Conference on Multimedia*. ACM.

[50] X. Wan, J. Yang, and J. Xiao. 2008. Towards a unified approach to document similarity search using manifold-ranking of blocks. *Information Processing & Management* 44, 3 (2008), 1032–1048.

[51] H. Wang, Y. Cai, Y. Zhang, H. Pan, W. Lv, and H. Han. 2015. Deep learning for image retrieval: What works and what doesn't. In *Proceedings of the 2015 IEEE International Conference on Data Mining Workshop*. IEEE.

[52] H. Wang, M. He, Z. Wei, S. Wang, Y. Yuan, X. Du, and J.-R. Wen. 2021. Approximate graph propagation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1686–1696.

[53] H. Wang, Z. Wei, Y. Liu, Y. Yuan, X. Du, and J.-R. Wen. 2021. ExactSim: Benchmarking single-source SimRank algorithms with high-precision ground truths. *The VLDB Journal* 30, 6 (2021), 989–1015.

[54] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D. L. Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM.

[55] Q. Wang, J. Lin, and Y. Yuan. 2016. Salient band selection for hyperspectral image classification via manifold ranking. *IEEE Transactions on Neural Networks and Learning Systems* 27, 6 (2016), 1279–1289.

[56] S. Wang, Y. Tang, X. Xiao, Y. Yang, and Z. Li. 2016. HubPPR: Effective indexing for approximate personalized PageRank. *Proceedings of the VLDB Endowment* 10, 3 (2016), 205–216.

[57] S. Wang, R. Yang, X. Xiao, Z. Wei, and Y. Yang. 2017. FORA: Simple and effective approximate single-source personalized PageRank. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.

[58] Z. Wei, X. He, X. Xiao, S. Wang, Y. Liu, X. Du, and J.-R. Wen. 2019. PRSim: Sublinear time simrank computation on large power-law graphs. In *Proceedings of the 2019 International Conference on Management of Data*. 1042–1059.

[59] Z. Wei, X. He, X. Xiao, S. Wang, S. Shang, and J. R. Wen. 2018. TopPPR: Top-k personalized PageRank queries with precision guarantees on large graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM.

[60] J. Wu, Y. He, X. Guo, Y. Zhang, and N. Zhao. 2017. Heterogeneous manifold ranking for image retrieval. *IEEE Access* 5 (2017), 16871–16884.

[61] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. 2019. A comprehensive survey on graph neural networks. arXiv:1901.00596. Retrieved from https://arxiv.org/abs/1901.00596.

[62] B. Xu, J. Bu, C. Chen, D. Cai, X. He, W. Liu, and J. Luo. 2011. Efficient manifold ranking for image retrieval. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

[63] Y. Zhang, P. Pan, Y. Zheng, K. Zhao, Y. Zhang, X. Ren, and R. Jin. 2018. Visual search at Alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM.

[64] L. Zheng, Y. Yang, and Q. Tian. 2018. SIFT meets CNN: A decade survey of instance retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 5 (2018), 1224–1244.

[65] D. Zhou, O. Bousquet, Thomas N. Lal, J. Weston, and B. Schölkopf. 2004. Learning with local and global consistency. In *Proceedings of the International Conference on Neural Information Processing Systems*.

[66] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. 2004. Ranking on data manifolds. In *Proceedings of the International Conference on Neural Information Processing Systems*.

[67] W. Zhou, H. Li, and Q. Tian. 2017. Recent advance in content-based image retrieval: A literature survey. arXiv:1706.06064. Retrieved from https://arxiv.org/abs/1706.06064.

Received 16 September 2019; revised 19 August 2022; accepted 18 September 2022