Exploiting the Block Structure of Link Graph for Efficient Similarity Computation

Pei Li[1,2], Yuanzhe Cai[1,2], Hongyan Liu[3], Jun He[1,2], and Xiaoyong Du[1,2]

[1] Key Labs of Data Engineering and Knowledge Engineering, Ministry of Education, China
[2] School of Information, Renmin University of China, Beijing, China
{lp,yzcai,hejun,duyong}@ruc.edu.cn
[3] Department of Management Science and Engineering, Tsinghua University, Beijing, China
hyliu@tsinghua.edu.cn

Abstract. In many real-world domains, link graph is one of the most effective ways to model the relationships between objects. Measuring the similarity of objects in a link graph is studied by many researchers, but an effective and efficient method is still expected. Based on our observation of link graphs from real domains, we find the block structure naturally exists. We propose an algorithm called *BlockSimRank*, which partitions the link graph into blocks, and obtains similarity of each node-pair in the graph efficiently. Our method is based on random walk on two-layer model, with time complexity as low as $O(n^{4/3})$ and less memory need. Experiments show that the accuracy of *BlockSimRank* is acceptable when the time cost is the lowest.

Keywords: Block, link graph, SimRank, object ranking.

1 Introduction

Many applications require executing "find-similar-object" query in their transactions, which implies the computation of similarity between objects. As an example, thinking about clustering a collection of documents, we should compute the similarity scores of each pair documents first. In collaborative filtering, similar items are clustered based on their similarities [6].

In many real-world domains, linkage among objects can be the most useful information for similarity measuring. One obvious example is the WWW. Due to the limited effect of understanding web pages automatically, link analysis is still a popular approach to exploit the web structure. If we treated these object-to-object links as edges, the real-world domain can be best described as a heterogeneous graph. In this paper, we focus our research target on the similarity computation of link graph, which is the abstract model of many real-world domains.

There have been many similarity computing methods [2, 3, 4, 5], which will be introduced in related work. Thereinto, Jeh and Widom [2] propose a notable method called *SimRank* for assessing the similarity that says "two objects are similar if they are related to similar objects". *SimRank* provides a wonderful definition for similarity

T. Theeramunkong et al. (Eds.): PAKDD 2009, LNAI 5476, pp. 389–400, 2009.

on a link graph, but unfortunately, it computes the similarity between every pair of objects, which results in high complexity in both time and space.

The study of real-world datasets indicates that block structure naturally exists among objects. Taking the web graph for example, web pages of host *A* are more likely to cite pages belonging to the same host *A*. Viewing a host as a block, the web graph can be regarded as a block structure, and the similar phenomenon can be found in other domains. As another example, based on the statistics of words frequencies occurring in articles, the density of linkages between collections of articles and words is shown in Figure 1(a) (adapted from Figure 2 in [7]).

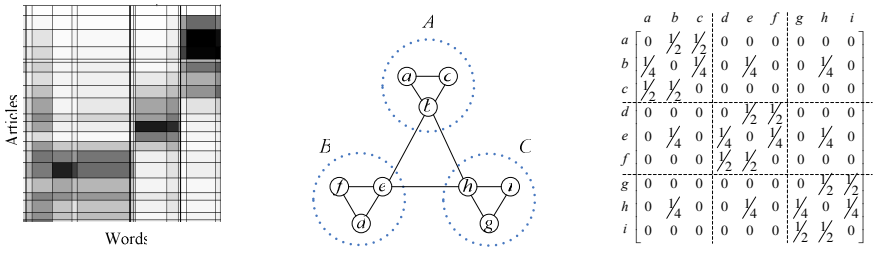

Fig. 1. (a) Density of linkages between articles and words. (b) The graph of a heterogeneous dataset. (c) The corresponding transition matrix.

Taking advantage of the block structure, we propose an algorithm called *BlockSimRank* to effectively measure similarity by following 4 steps: (1) partition the link graph into blocks properly; (2) compute the similarity of each block-pair; (3) for each block, use *SimRank* to obtain local similarity scores; (4) estimate the "inter-block" nodes similarity using block-pair similarity and local similarity.

Comparing with *SimRank*, our method can significantly speed up the identification of similar objects. *SimRank* computes the similarity of each node-pair. However, *BlockSimRank* only computes the similarity between blocks and the similarity between nodes in the same block. The similarity between nodes in different blocks is estimated reasonably. For example, in a link graph shown in Figure 1(b), we can gain its transition matrix shown in Figure 1(c). Because most edges are intra-block links, only the dense area of transition matrix (diagonal area in Figure 1(c)) is worth considering. For a graph having n nodes, supposing this graph has m blocks and let d denote the average number of neighbors, the time complexity of *SimRank* and *BlockSimRank* are $O(kn^2d^2)$ and $O(kd^2(m^2+n^2/m))$ respectively, where k is the number of iterations. When setting m properly, the least time cost of *BlockSimRank* is $O(n^{4/3})$.

Experiments on real datasets are conducted to test the accuracy and efficiency of *BlockSimRank*. By partitioning the link graph, our method can remarkably speed up the similarity computation and win great advantage in efficiency with the accuracy retained, compared with other methods such as *SimRank* and *SimFusion*[4].

The rest of the paper is organized as follows. Section 2 surveys related work. In Section 3, we exploit the block structure of link graph in real domains. Afterwards, Section 4 describes the *BlockSimRank* method and explains its theoretical model. The results of experiments are shown in Section 5 and this study is concluded in Section 6.

2 Related Work

Similarity measuring has been extensively studied in different disciplines for a long time, with multiple methods proposed [2, 3, 4, 5, 9]. Traditional approaches that calculate the similarity of documents using Vector Space Model (VSM) (included in [3]) and its variations can be viewed as a general algorithm, which map documents and queries into a vector space. They differ in how the vectors are constructed and how weights are assigned [4]. These methods are useful to compute the similarity between sets of objects (e.g., a collection of documents), whereas they are not good at using the external information about objects (e.g., the references among documents).

In some situations, linkages among objects can be the only or the most explicit information available. Many datasets are heterogeneous [1], in which there may be multiple object and link types, and are best described as networks or graphs. Naively applying traditional approaches may bring on invalid conclusions.

Some researchers use multiple relationships to compute the similarity between objects based on different theoretical foundations. In the WWW, [5] explains an approach using hyperlinks among pages to calculate the similarity of web objects. To realize the global ranking of objects in a static graph, Jeh and Widom [2] proposed a strategy called *SimRank*, in which the similarity between two objects is updated iteratively by the average similarity between objects linked with them. For example, in Figure 1(b), the similarity between *a* and *d* (defined as $S(a, d)$) is decided by the similarity $S(c, f)$, $S(c, e)$, $S(b, f)$ and $S(b, e)$. The intuitive underlying model of *SimRank* is "random surfer-pairs", a concept derived from *PageRank* algorithm [8]. Unfortunately, *SimRank* takes $O(N^2)$ time and $O(N^2)$ space, making it impractical for large datasets. The authors of [2] also discuss a pruning technology to approximate *SimRank*, but it is a challenge to select the right node pairs in initial stage.

Yin et al. [7] proposed a hierarchical structure called *SimTree* to represent similarities between objects in a compact way. *SimTree* only computes and stores the similarities of sibling nodes and the ratio of each node compared with its parent, thus, it obtains great efficiency compared with *SimRank*. Besides, Xi et al. [4] use a Unified Relationship Matrix (*URM*) to represent a collection of heterogeneous objects and their interrelationships. By iteratively computing over the *URM*, they present a similarity-calculating algorithm called *SimFusion*. *SimFusion* can effectively integrate relationships from multiple sources to measure the similarity, and takes $O(kn^2d)$ time, where d is a constant with respect to n.

In this issue, Sun et al. [9] proposed a correlation degree computation approach for relevance search and anomaly detection, which combines random walk and graph partitioning together to improve scalability. Although the principle of [9] is similar to our work, it is only suitable for bipartite graph, while our method is applicable to arbitrary graph, even if this graph is not a connected graph.

Our method, *BlockSimRank*, is originated from the observation of block structure concealed in the link graph. A similar work has been done by [11], which exploit the block structure of the web for computing *PageRank*. The underlying model of *BlockSimRank* based on random walk [10], a concept that simulates walkers randomly surf the graph along edges; and k-way graph partitioning, a method introduced by [12], which is a high-quality and computationally inexpensive refinement algorithm. Our method can be mainly applied in link-based object ranking.

3 Block Structure of Link Graph in Real Datasets

The taxonomy or hierarchical categories are ubiquitous among data objects in many domains. Usually, two objects of the same type are more similar than objects in different categories, and it makes similar objects gathered into the same block.

More precisely, for two blocks A and B, let $N(A)$ denote the number of links of block A (all "intra-block" and "inter-block" links are included), and $N(A, B)$ denote the number of links between A and B. Obviously, $N(A, B) = N(B, A)$, and $N(A, A)$ represents the number of intra-block links of A. We take

$$R(A,B) = \frac{N(A,B)}{N(A)} \tag{1}$$

to measure the ratio between $N(A, B)$ and $N(A)$. It is easy to know $0 \le R(A,B) \le 1$ and if $R(A, B) = 1$, all links starting from objects in block A point to objects in block B. For Example, in Figure 1(b), the value of $R(A, B)$ is 1/5.

Block Structure of the Citation Graph. Considering a corpus of scientific papers, each paper can be regarded as an identically distributed node, and the references or cited-by relationships correspond to edges between nodes. Our dataset is crawled from ACM Computing Classification System (CCS) [13], which is a subject classification system for Computer Science. Taking Section *F* in ACM CCS for an example, there are 6 main categories in it, and each category includes some subtopics.

Treating each category as a block, We use $R(A, B)$ defined in equation (1) to represent the flow of links from A to B, and color the plot in the grid with black or grey according to the value of $R(A, B)$. The width of a plot corresponds to the number of papers in that category. The block structure of citation graph is shown in Figure 2.

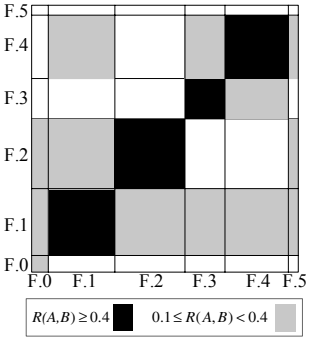

Fig. 2. Block structure of the citation graph of Section *F* in ACM CCS

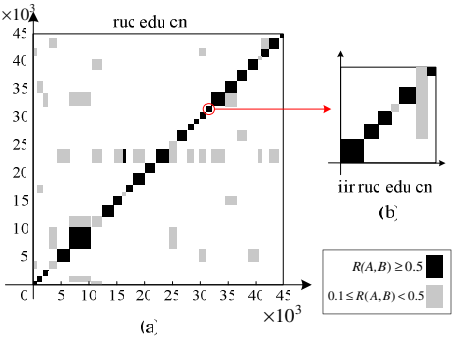

Fig. 3. (a) Block structure of *ruc.edu.cn* at host level. (b) Block structure of *iir.ruc.edu.cn* at directory level.

Block Structure of the Web Graph. The WWW is naturally modeled as a huge graph, with nodes representing pages and edges representing hyperlinks. We do a crawl of the *ruc.edu.cn* in July 2008 to generate a link graph for analysis. This graph contains roughly 45,000 nodes, and more than 0.5 million links with ineffectual

Table 1. Terms for a hierarchical view of URL on the web

Terms	Example
domain	ruc.edu.cn
host	iir.ruc.edu.cn
directory	iir.ruc.edu.cn/people/
page	iir.ruc.edu.cn/people/students.html

hyperlinks (links citing other pages out of the graph) removed. To investigate the structure of the web, we use the terms given in Table 1.

A similar work has been done by Kamvar et al. [11] by using dot-plots to visualize the link matrix at page level. Our work is done at host level (Figure 3 (a)) and directory level (Figure 3 (b)) based on the measurement of links flow among blocks. A view of nested block structure is shown in Figure 3, by filling plots with different color according to $R(A,B)$. The aggregation of black plots on the diagonal line indicates that most links are "intra-block" links.

4 The *BlockSimRank* Method

4.1 Preliminaries

Transition Matrix. Given a undirected graph $G(V, E)$, supposing there is a random surfer standing on node a, he has identical probability to visit each node direct-connected to a on next step, and possibility of *zero* to other nodes. Note that edge weights may be imported to represent varying importance of links, but it's not the key point of our work. We only focus on undirected graphs and it's easy to extend our method to directed graphs.

A transition matrix or stochastic matrix is introduced to describe the transition probability of a Markov Chain [14]. We define transition matrix T as a square matrix each of whose rows consists of nonnegative values, with each row summing to 1. In an undirected graph, T is symmetric. For example, the transition matrix shown in Figure 1 (c) corresponds to the graph in Figure 1 (b). Based on the block feature we discussed in Section 3, we use

$$T = \begin{bmatrix} P_{AA} & P_{AB} & P_{AC} \\ P_{BA} & P_{BB} & P_{BC} \\ P_{CA} & P_{CB} & P_{CC} \end{bmatrix}$$

to generalize it, where P_{IJ} is the probability matrix between all nodes in block I and all nodes in J. Note that only the dense valued areas (e.g. P_{AA}, P_{BB} and P_{CC}) have significant effect on similarity computation. This feature is utilized by our method.

Graph Partitioning. The graph partitioning problem is to divide the vertices of a graph into n roughly equal parts, with the number of edges connecting nodes in different parts minimized. In our method, blocks are detected by graph partitioning. Many algorithms [12, 15, 16] have been developed to find reasonably partitions. We use the multilevel k-way partitioning method proposed by Karypis et al. [12], whose

strategy is performing a *k*-way partitioning on the smaller graph, and then refine it to construct a *k*-way partitioning of the original graph. This algorithm partitions a graph $G = (V, E)$ in $O(|E|)$ time and is implemented in METIS package [18].

4.2 Assessing Similarity between Nodes

Basic *SimRank*. As algorithm *SimRank* [2] is closely related to our work, we describe it here. Given an undirected graph $G = (V, E)$, we denote the *k*-th iterative similarity between nodes V_a and V_b by $S_k(V_a, V_b) \in [0,1]$. If $a=b$ then $S_k(V_a, V_b) = 1$. Otherwise, given transition matrix T and initial similarity matrix Sim_0, we get

$$S_k(V_a, V_b) = c \cdot \sum_{i=1}^{O(T)} \sum_{j=1}^{O(T)} T_{ai} \cdot T_{bj} \cdot S_{k-1}(V_i, V_j) \tag{2}$$

where c is a decay constant (usually set to be 0.8) describing the attenuation of link influences. $O(T)$ is the order of transition matrix T. At the same time, T_{ai} is the element of T on the a-th row and i-th column, and analogously for T_{bj}. A slight notice here is the initialization of $S_k(V_a, V_b)$ when k=0. Since we can't foreknow the similarity between two objects before iteration, it is reasonable to simply define $S_0(V_a, V_b) = 1$ for $a=b$, and $S_0(V_a, V_b) = 0$ for $a \neq b$.

We give the major steps of *SimRank* algorithm as follows.

Algorithm 1. (**Basic *SimRank***)
Input: *transition matrix T, initial similarity matrix Sim_0, tolerance factor t.*
Output: c*onvergent similarity matrix Sim_k.*

```
k ← 1
while ( max(| S_k(V_a,V_b) − S_{k−1}(V_a,V_b) |) > t )
    k ← k + 1
    Sim_{k-1} ← Sim_k
    for each element S_k(V_a, V_b) in Sim_k
        S_k(V_a,V_b) = c · Σ_{i=1}^{O(T)} Σ_{j=1}^{O(T)} T_ai · T_bj · S_{k−1}(V_i,V_j)
    end for
end while
return Sim_k
```

Local Similarity and Global Similarity. Local similarity is the similarity between objects in the same block without considering objects in other blocks. Let $LSim(B)$ denote the local similarity of block B. $LSim(B)$ is the iteration convergence of all node-pairs' similarities in block B, which means given transition matrix P_{BB} (defined in Section 4.1) and initial similarity matrix $LSim_0(B)$ (a identity matrix), we can get $LSim(B)$ recursively by Equation (2). As an example, for block A in Figure 4(a), relationships between A and its nodes can be better visualized in another view shown in Figure 4 (b), which indicates $LSim(A)$ is a 3-by-3 matrix.

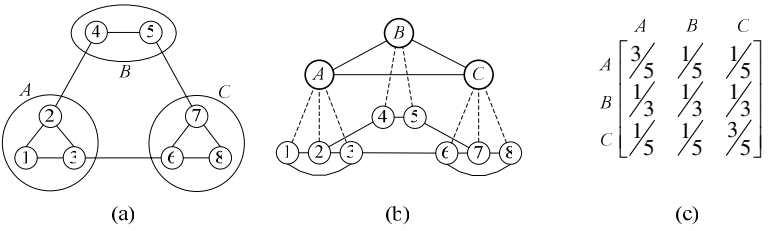

Fig. 4. (a) A view of block structure. (b) Another view of block structure. (c) Transition matrix between blocks.

Definition 1. (**Block Similarity**) Treating each block as an object, we can compute the similarity between blocks as we do between objects by Equation (2). The transition probability between block *A* and *B* is *R*(*A, B*) (defined by Equation (1)), so we can get transition matrix *T*. Figure 4 (c) shows an transition matrix between blocks. Let *BSim*(*A, B*) denote the similarity between block *A* and *B*.

Definition 2. (**Inter-Block Nodes Similarity**) Inter-block nodes similarity is the similarity between nodes in different blocks. Supposing node V_a and V_b are in block *A* and *B* respectively, we define the similarity between V_a and V_b on the *k*-th iteration by

$$S_k(V_a, V_b) = S_k(V_a, A) \cdot BSim(A, B) \cdot S_k(V_b, B) \tag{3}$$

where $S_k(V_x, X)$ represents the similarity between object *x* and the center of block *X*. Usually the center of block *X* is virtual and hard to ascertain, so we take the average similarity between *x* and every object in block *X* to estimate it. That is

$$S_k(V_x, X) = \frac{1}{|X|} \sum_{i=1}^{|X|} S_k(V_x, V_i) \tag{4}$$

where |*X*| is the number of objects in block *X*. We will explain equation (3) in section 4.4 later. Further study indicates that it's not necessary to compute inter-block nodes similarity on every iteration.

Theorem 1. If block similarity and local similarity are convergent, inter-block nodes similarity will be convergent too.

Proof. If local similarity $S_k(V_x, V_i)$ is convergent, $S_k(V_x, X)$ will be convergent too according to Equation (4). Since block similarity *BSim*(*A, B*) is convergent, we can get the Theorem 1 soon from Equation (3).

Definition 3. (**Global Similarity**) Given an undirected graph *G* = (*V, E*), as stated above, for a node-pair V_a and V_b, if they belong to the same block, their similarity $S(V_a, V_b)$ is the local similarity decided by Equation (2), otherwise, it is the inter-block nodes similarity decided by Equation (3). We call the similarity of any node-pair calculated this way global similarity, and use *GSim* to denote it.

The *BlockSimRank* Algorithm. The block structure suggests a fast algorithm. We propose *BlockSimRank* for more effective and efficient similarity computation, and its major steps are summarized as follows:

1. Split the graph into n roughly equal blocks using METIS [18];
2. For the k-th iteration, regarding each block as an object, compute the Block Similarity Matrix using basic *SimRank* algorithm;
3. Compute the Local Similarity Matrix of each block by basic *SimRank* algorithm;
4. If block similarity and local similarity are not convergent, let $k = k+1$ and then jump to step 2; else continue;
5. Estimate the Inter-Block Nodes Similarity using Equation (3);
6. Obtain global similarity matrix *GSim*.

4.3 Complexity Analysis

For simplicity, we assume the link graph $G = (V, E)$ has n objects and can be split into m roughly identical blocks.

Time Complexity. The time consumed by step 1 of *BlockSimRank* Algorithm usually can be ignored, because it takes $O(|E|)$ time and the real graphs are often sparse. The time cost of *BlockSimRank* is mainly composed of block similarity computation (step 2) and local similarity computation (step 3). Noting that the time cost of *SimRank* is $O(kd^2n^2)$, we write the equation for time complexity of *BlockSimRank* as follows:

$$Time(m) = kd^2m^2 + mkd^2(n/m)^2 = kd^2(m^2+n^2/m) \tag{5}$$

k is the number of iteration and d^2 is the average direct-connected neighbor pairs of a block-pair or node-pair (usually a constant with respect to n^2). Calculating the derivative of $Time(m)$ and setting $dTime(m)/dm=0$, we get $m_0 = (2n)^{2/3}/2$, and when $m=m_0$, the time complexity is $O(n^{4/3})$, that is the lowest.

Space Complexity. Since it is not necessary to put the entire global similarity matrix in RAM, *BlockSimRank* takes $O(m^2+n^2/m)$ space to store the block similarity matrix and local similarity matrices. When $m=m_0$, the space complexity is the smallest too.

Our experiment in Section 5 shows that the accuracy is still comparable to the accuracy of *SimRank* when $m=m_0$, so it is our suggestion to set $m=m_0$ when using *BlockSimRank* method. Besides, the computations of local similarity matrices are independent, which indicates the possibility of parallelization.

4.4 Theoretical Model

The block structure implies that a random surfer is more likely to meet another surfer from the same block. The meeting of two surfers from different blocks also needs to consider, but the probability is little and we can estimate it. We develop an intuitive model based on two kinds of random surfers: the surfer traveling among blocks (we call it *BSurfer*) and the surfer traveling among local nodes (we call it *NSurfer*).

Random Walk on Two Layers Model. In Figure 5 (a), all nodes belong to the same block. The similarity $S(a, b)$ is determined by similarities of all its direct-connected

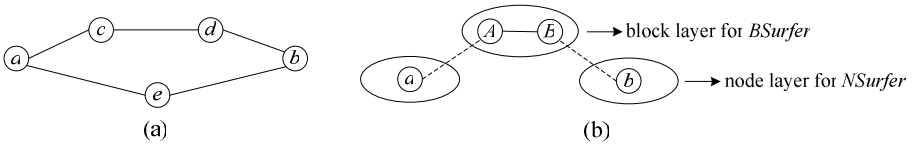

Fig. 5. (a) The graph for *NSurfer*. (b) The meeting of two *NSurfers* in different blocks

neighbor pairs and the probability from (a, b) to these node-pairs. That is to say, $S_k(a, b) = 0.25(S_{k-1}(c, e)+S_{k-1}(c, d)+S_{k-1}(e, e)+S_{k-1}(d, e))$.

If nodes a and b are in different blocks as shown in Figure 5 (b), *NSurfer* at node a and *Nsurfer* at node b can't meet without the help of *BSurfer*. Regarding *BSurfer* as a transmitter or bridge, the similarity $S(a, b)$ is determined by $BSim(A, B)$ and the probability from (a, b) to block-pair (A, B). We use Equation (4) to approximate the probability from object x to its block X. Thus we have the following theorem.

Theorem 2. Global similarity of any node-pair in Graph G can be computed based on the Random Walk on two layers model.

Proof. Given a node-pair (a, b), its similarity is determined by similarities of all its in-neighbor node-pairs (or block-pair) and the probability from (a, b) to these node-pairs (or block-pair).

5 Experimental Evaluation

We described above an approach for efficient similarity computation on link graph. In this section, the accuracy and efficiency of our approach will be tested, compared with the following similarity measuring methods: (1) *SimRank* [2], an approach that iteratively computes the similarity of each node-pair; (2) *SimFusion* [4], an approach that reinforces and propagates the similarity between objects.

All experiments are performed on a PC with a 1.86G Intel Core 2 processor, 2 GB memory, and Windows XP Professional. All algorithms are implemented using Java.

Datasets. A good evaluation of similarity measuring methods is difficult, because the similarity between two objects is hard to ascertain without performing extensively user studies. ACM CCS [13] is a credible subject classification system for Computer Science. It can provide an identification of similar papers by organizing these papers in the same category. Our dataset is crawled from Section *F* in ACM CCS. With the

Table 2. (a) Node number of each category. (b) Edge number between two categories.

Category	Number of nodes
F.1	2738
F.2	4431
F.4	2380

(a)

	F.1	F.2	F.4
F.1	9513	4084	2828
F.2	4084	18843	1046
F.4	2828	1046	8508

(b)

ineffectual papers removed, we get a citation graph using the reference and "cited-by" information. This undirected graph has 9549 nodes and 44822 edges. The detailed node number of each category and edge number between each pair of categories are shown in Table 2.

The Selection of Partition Number *m*. When partitioning a graph into blocks at the beginning of *BlockSimRank* algorithm, the number of partitions should be decided. From Equation (5) we know *m* influences time complexity remarkably. In the meantime, too many partitions imply too many edges between different blocks, which induce the decline of accuracy.

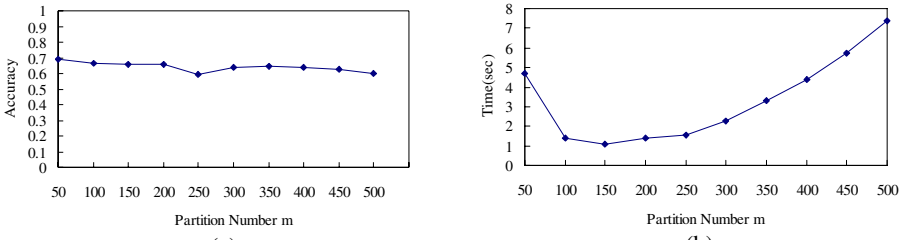

Fig. 6. (a) Accuracy of different partition numbers. (b) Time/iteration.

There is a trade-off between accuracy and efficiency. The accuracy and the time consumed on each iteration of different partition numbers are shown in Figure 6 (a) and (b) respectively. With the increase of partition numbers, the accuracy descends slowly. As we discussed in Section 4.3, if the assumed conditions are satisfied, when $m = m_0 = (2n)^{2/3}/2$, time cost is the lowest. In real situations, we set $m_0 = e(2n)^{2/3}/2$, where e is an adjustment factor and in our ACM dataset, $e \approx 0.4$, making $m_0 \approx 150$.

Experiments show that the accuracy of *BlockSimRank* is acceptable when time cost is the lowest. So we suggest setting $m = m_0$ when using *BlockSimRank* algorithm.

Accuracy. We use PAM [17], a *k*-medoids clustering approach, to cluster papers into groups based on global similarity matrix. In the meantime, ACM CCS provides a credible classification on these papers. Comparing these groups with CCS categories, let *C* denotes the number of correct classified papers, and *N* denotes the total number of papers. We define accuracy as the ratio between *C* and *N*.

In this experiment, we compare *BlockSimRank* with *SimRank* and *SimFusion* on accuracy. We set $m = m_0$ in *BlockSimRank* (now time cost is the lowest). As Table 3 shows, *BlockSimRank* wins slight advantage to other methods, which is because graph partitioning can eliminate noise to some extent.

Table 3. Accuracy of different approaches

Approach	Accuracy
BlockSimRank($m=m_0$)	0.6609
SimRank	0.6419
SimFusion	0.6073

Table 4. Performances of different approaches

Approach	Time/Iteration	Total
BlockSimRank	1.2 sec	× 10
SimRank	13222.7 sec	× 9
SimFusion	5763.2 sec	× 15

Performances. The motivation of *BlockSimRank* is exploiting the block structure to improve the performance of *SimRank*. By partitioning a global matrix into several local matrices, the performance is enhanced markedly. The time complexities on each iteration of *SimRank*, *BlockSimRank* and *SimFusion* are $O(d^2n^2)$, $O(d^2(m^2+n^2/m))$ and $O(dn^2)$ respectively, where d is related to n and explained above.

When we set $m = m_0$, the time cost of *BlockSimRank* is $O(n^{4/3})$, which wins notable advantage to *SimRank*. We list the time consumed on each iteration and the total time before convergence in Table 4. The tolerance factor of convergence is set to be 0.001, which is the maximum difference (explained in next subsection) of similarity scores between adjacent iterations. From Table 4 we can see the improvement of performances from *SimRank* to *BlockSimRank* is huge, which is from $O(n^2)$ to $O(n^{4/3})$, where n is the node number 9549.

Convergence Rate. We measure the convergence rate from two aspects: maximum difference and accuracy. $S_k(i, j)$ denoting a similarity score on the k-th iteration, maximum difference M_k can be described as $max(|S_k(i, j) - S_{k-1}(i, j)|)$. If M_k is less than the tolerance factor of convergence (set to be 0.001), iterative process will finish.

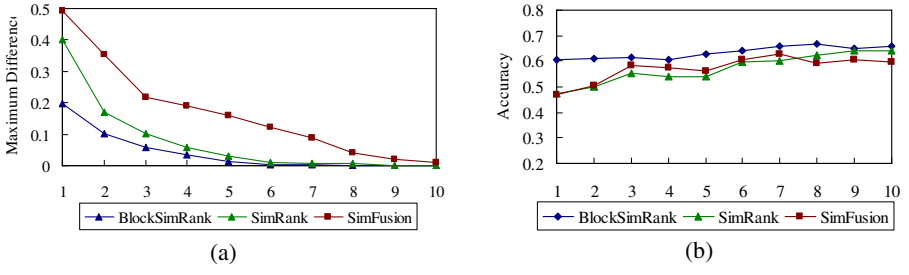

Fig. 7. (a) Maximum differences of each iteration. (b) The accuracy of each iteration.

By analyzing similarity matrices of the first ten iterations of each approach, we get the maximum difference of each iteration shown in Figure 7 (a), and the accuracy of each iteration shown in Figure 7(b). Comparing with other methods, *BlockSimRank* holds advantage at the beginning of iterative process. Considering the first iteration in Figure 7, the maximum difference of *BlockSimRank* is the smallest and the accuracy is the highest. This advantage is attributed to graph partitioning, because it can detect potential groups via partitioning a graph into blocks. The similarity between objects in the same block is usually higher than the one in different blocks, which results in more accurate clustering. Due to this advantage, *BlockSimRank* algorithm has higher convergence rate than *SimFusion*, and is as fast as *SimRank* in most cases.

6 Conclusions

In this paper, we propose a high efficient similarity computation method by exploiting the block structure of link graph. Many datasets involve relationships between objects and can be best described as link graphs. Using the block structure implied in the link graph, we propose an algorithm called *BlockSimRank*, which partitions a graph into

blocks, and iteratively computes the similarity between blocks and the similarity between objects in the same block until similarity matrices are convergent, and then estimate the similarity between objects in different blocks properly. The time cost of our method is $O(n^{4/3})$, whereas *SimRank* takes $O(n^2)$ time. Experimental results show *BlockSimRank* achieves high efficiency and acceptable accuracy in computing similarity of a link graph.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China under Grant No. 70871068, 70621061, 70890083, 60873017, 60573092 and 60496325.

References

1. Getoor, L., Diehl, C.P.: Link mining: A survey. In: SIGKDD 2005 Explorations, vol. 7(2), pp. 3–12 (2005)
2. Jeh, G., Widom, J.: SimRank: A measure of structural-context similarity. In: SIGKDD 2002, pp. 538–543 (2002)
3. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press/Addison-Wesley (1999)
4. Xi, W., Fox, E.A., Fan, W., Zhang, B., Chen, Z., Yan, J., Zhuang, D.: SimFusion: measuring similarity using unified relationship matrix. In: SIGIR 2005, pp. 130–137 (2005)
5. Dean, J., Henzinger, M.R.: Finding Related Pages in the World Wide Web. In: WWW 1999, pp. 1467–1479 (1999)
6. Shardanand, U., Maes, P.: Social information filtering: Algorithms for automating "word of mouth". In: Proceedings of the Conference on Human Factors in Computing Systems, Denver, Colorado (1995)
7. Yin, X., Han, J.: Yu. P.S.: Linkclus: Efficient clustering via heterogeneous semantic links. In: VLDB 2006, pp. 427–438 (2006)
8. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: Bringing order to the Web. Technical report, Stanford University Database Group (1998)
9. Sun, J., Qu, H., Chakrabarti, D., Faloutsos, C.: Relevance search and anomaly detection in bipartite graphs. SIGKDD Explorations 7(2), 48–55 (2005)
10. Lovasz, L.: Random walks on graphs: a survey. Combinatorics, Paul Erdos is Eighty, vol. 2, Keszthely (Hungary), pp. 1–46 (1993)
11. Kamvar, S.D., Haveliwala, T.H., Manning, C.D., Golub, G.H.: Exploiting the Block Structure of the Web for Computing PageRank. Technical Report, Stanford University, Stanford, CA (2003)
12. Karypis, G., Kumar, V.: Multilevel k-way partitioning scheme for irregular graphs. Journal of Parallel and Distributed Computing 48(1), 96–129 (1998), `http://www.cs.umn.edu/~karypis`
13. ACM Computing Classification System, `http://portal.acm.org/ccs.cfm`
14. Kallenberg, O.: Foundations of Modern Probability. Springer, New York (1997)
15. Meila, M., Shi, J.: Learning Segmentation by Random Walks. Advances in Neural Information Processing Systems (2001)
16. Fischer, I., Poland, J.: Amplifying the block matrix structure for spectral clustering. In: Proceedings of the 14th Annual Machine Learning Conference of Belgium and the Netherlands, pp. 21–28 (2005)
17. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: an Introduction to Cluster Analysis. John Wiley & Sons, Chichester (1990)
18. Karypis, G., Kumar, V.: METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System. Technical Report, Department of Computer Science, University of Minnesota (1995)