# Accuracy Estimate and Optimization Techniques for SimRank Computation

Dmitry Lizorkin
Institute for System Programming
of the Russian Academy of Sciences
lizorkin@ispras.ru

Pavel Velikhov
Institute for System Programming
of the Russian Academy of Sciences
velikhov@ispras.ru

Maxim Grinev
Institute for System Programming
of the Russian Academy of Sciences
maxim@grinev.net

Denis Turdakov
Institute for System Programming
of the Russian Academy of Sciences
turdakov@ispras.ru

## ABSTRACT

The measure of similarity between objects is a very useful tool in many areas of computer science, including information retrieval. SimRank is a simple and intuitive measure of this kind, based on graph-theoretic model. SimRank is typically computed iteratively, in the spirit of PageRank. However, existing work on SimRank lacks accuracy estimation of iterative computation and has discouraging time complexity.

In this paper we present a technique to estimate the accuracy of computing SimRank iteratively. This technique provides a way to find out the number of iterations required to achieve a desired accuracy when computing SimRank. We also present optimization techniques that improve the computational complexity of the iterative algorithm from $O(n^4)$ to $O(n^3)$ in the worst case. We also introduce a threshold sieving heuristic and its accuracy estimation that further improves the efficiency of the method.

As a practical illustration of our techniques we computed SimRank scores on a subset of English Wikipedia corpus, consisting of the complete set of articles and category links.

## 1. INTRODUCTION

The requirement for measuring similarity between objects arises in many fields of computer science; examples include recommender systems, “related pages” queries of web search engines, document classification and clustering. Large amounts and fast growth of information require machine aid to humans for finding, classifying and analyzing requested information. Such a range of challenges includes automatically detecting objects similar to a given object and ranking them in accordance with their similarity scores. While humans make judgment on object similarity intuitively, based on their previous experience, the task of systematically computing object similarity by a machine remains nontrivial. For practical applicability, an effective similarity measure should both reflect human intuition on objects similarity and provide reasonable computational complexity.

For the existing similarity measures, two broad categories can be outlined: (i) content- or text-based similarity measures that treat each object as a bag of items or as a vector of word weights, and (ii) link-based ones that consider object-to-object relations expressed in terms of links. In the recent research [12], the extensive evaluation of different similarity measures was performed, and link-based measures produced systematically better correlation with human judgments compared to text-based measures. It is worth mentioning that the success of the Google search engine began with its ability to rank search results in accordance with human expectations; the latter feature was essentially based on a purely link-based ranking algorithm called PageRank [1]. From this perspective, it is reasonable to assume that an effective similarity measure would have a comparable impact for computer science techniques as PageRank ranking algorithm had for web search.

Considering the outlined state of the art, the similarity measure *SimRank* [7] can be considered as one of the promising ones, due to the following reasons. SimRank is a link-based similarity measure, and builds on the approach of previously existing link-based measures. SimRank is based on both a clear human intuition and a solid theoretical background. Similarly to PageRank, SimRank is defined recursively with respect to “random surfer” model and is computed iteratively. Unlike the similarity measures that require human-built hierarchies, SimRank is applicable to any domain with object-to-object relationships, including the Web.

Nevertheless, existing work on SimRank lacks two important issues. Firstly, although SimRank iterative similarity scores are known to converge [7], a real-life computation naturally involves performing a finite number of iterations.

PVLDB '08, August 23-28, 2008, Auckland, New Zealand
Copyright 2008 VLDB Endowment, ACM 978-1-60558-305-1/08/08

However, a potential difference between SimRank iterative similarity scores and theoretical ones remains an open question. The symmetric question is finding out the precise number of iterations sufficient to guarantee a desired accuracy.

Secondly, optimization issue of SimRank computation is not the primary focus of the original SimRank proposal [7]. To the best of our knowledge, there exists the only research work concerning SimRank optimization [3]. That work is initially oriented on SimRank *probabilistic* computation, details are considered in the Related Work section. As for SimRank iterative computation, optimization has not been addressed in scientific literature yet, and the time complexity of the straightforward SimRank computation is an obstacle for using SimRank on practical data corpora.

This paper presents a solution to both issues, both mathematically proven and practically approved by experimental results. In summary, the main contributions of this paper are the following:

- A precise accuracy estimate is presented for SimRank scores computed iteratively, with respect to the theoretical ones. This allows one to find out the number of iterations required for achieving the desired accuracy.

- Optimization techniques for SimRank computation are presented that improve SimRank computational complexity from $O(n^4)$ to $O(n^3)$. A threshold sieving heuristic is introduced and its accuracy estimation is given that further improves the efficiency of the method.

- SimRank computational viability for relatively large object corpora in the presence of the suggested optimization techniques is verified experimentally by computing SimRank similarity scores for a subset of English Wikipedia corpus, consisting of the complete set of articles and category links.

The rest of the paper is organized as follows. In the next section, SimRank overview is given and the necessary notations and formulae are introduced. In Sect. 3, accuracy estimate for the SimRank iterative computation model is established. In Sect. 4, SimRank optimization techniques are suggested and summarized in the general algorithm; the computational complexity of the proposed algorithm is given. Sect. 5 gives the overview of the related work. Experimental results are presented in Sect. 6. Future work is discussed in Sect. 7.

## 2. SIMRANK OVERVIEW

In this section, SimRank overview is given, and notations, formulae and SimRank properties necessary for further discussion are provided. The material presented in this section recalls Jeh's and Widom's work [7].

SimRank approach is focused on "object-to-object relationships found in many domains of interest" [7]. From the relationships perspective, a domain is assumed to be modeled as a (logical) graph, with nodes representing objects and edges (links) representing relationships.

The *basic intuition* behind SimRank approach is: "two objects are similar if they are referenced by similar objects" [7]. Note that the given intuition is recursive by nature. As the base case, any object is considered maximally similar to itself, i.e. having a similarity score of 1 assigned.

Before presenting the mathematical formula that reifies the basic SimRank intuition, several notations are introduced. Given a graph $G(V, E)$ consisting of a set of nodes $V$ and a set of links $E$, the following two mappings are further assumed defined for each node $v$ in the graph:

- $I(v)$ denotes all in-neighbours of node $v$, i.e. all nodes that have a link to $v$:
$$I(v) = \{u \in V \mid (u, v) \in E\} \ .$$

- $O(v)$ denotes all out-neighbours of $v$, i.e. all nodes the node $v$ has a link to:
$$O(v) = \{w \in V \mid (v, w) \in E\} \ .$$

Notations $|I(v)|$ and $|O(v)|$ denote the number of nodes in $I(v)$ and $O(v)$ respectively. Individual member of $I(v)$ and $O(v)$ are referred to as $I_i(v)$, $1 \le i \le |I(v)|$ and $O_i(v)$, $1 \le i \le |O(v)|$; a particular order of members when associated with indices is not important for further discussion.

With the similarity score between objects $a$ and $b$ denoted by $s(a, b) \in [0, 1]$, the basic SimRank intuition is then written as follows:

$$\begin{aligned} s(a,a) &= 1 \ , \\ s(a,b) &= \frac{C}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b)) \ , \end{aligned} \qquad (1)$$

with a constant $C$ being the decay factor, $0 < C < 1$. For preventing division by zero in the general formula (1) in case of $I(a)$ or $I(b)$ being an empty set, $s(a, b)$ is specially defined as zero for $I(a) = \emptyset$ or $I(b) = \emptyset$.

We will further refer to $s(*, *)$ as SimRank *theoretical* similarity function, and refer to its value $s(a, b)$ as theoretical similarity score between nodes $a$ and $b$.

A solution to the SimRank equations (1) is reached by iteration to a fixed-point. For each iteration $k$, an *iterative* similarity function $R_k(*, *)$ is introduced, with $R_k(a, b)$ denoting the iterative similarity score between $a$ and $b$ on iteration $k$. The iterative computation is started with $R_0(*, *)$ defined as

$$R_0(a,b) = \begin{cases} 1 \ , & \text{if } a = b \ , \\ 0 \ , & \text{if } a \ne b \ . \end{cases} \qquad (2)$$

On the $(k + 1)$-th iteration, $R_{k+1}(*, *)$ is defined in special cases as

$$\begin{aligned} R_{k+1}(a,b) &= 1 \ , \ \text{if } a = b \ , \\ R_{k+1}(a,b) &= 0 \ , \ \text{if } I(a) = \emptyset \text{ or } I(b) = \emptyset \ , \end{aligned}$$

and is computed from $R_k(*, *)$ in the general case as follows:

$$R_{k+1}(a,b) = \frac{C}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_k(I_i(a), I_j(b)) \ . \qquad (3)$$

The following SimRank properties stated in [7] are worth noting for the purposes of our further discussion:

1. A solution $s(*, *)$ to SimRank equations (1) always exists and is unique, and $s(*, *) \in [0, 1]$.

2. For each $k$, $R_k(*, *)$ is a lower bound on the theoretical SimRank function $s(*, *)$, i.e. $R_k(a, b) \le s(a, b)$.

3. Iterative functions $R_k(*,*)$ converge to SimRank theoretical function $s(*,*)$, i.e. $\lim_{k\to\infty} R_k(a,b) = s(a,b)$.

We will further refer to these properties by their corresponding item numbers within the above list.

Symbol $K$ further denotes the total number of iterations performed; $n$ denotes the number of nodes in a graph.

## 3. ITERATIVE SIMILARITY ACCURACY ESTIMATE

Although Jeh and Widom proved iterative similarity convergence [7], SimRank practical computation naturally implies performing a finite number of iterations. From this perspective, no quantitative estimates were given for a potential difference between SimRank iterative similarity scores and theoretical ones. In this section, we fill in this gap and estimate the accuracy of computing SimRank iteratively.

The following proposition establishes the accuracy estimate for iterative similarity function $R_k(*,*)$ obtained after $k$ iterations with respect to theoretical similarity function $s(*,*)$.

PROPOSITION 1. *The difference between SimRank theoretical and iterative similarity scores decreases exponentially in the number of iterations and uniformly for every pair of nodes. Precisely, for every iteration number $k = 0,1,2,\ldots$ and for every two nodes $a, b$, the following estimate holds:*

$$s(a,b) - R_k(a,b) \le C^{k+1} \ . \tag{4}$$

In conjunction with SimRank Property 2 listed above, the proposition gives the following estimate:

$$0 \le s(a,b) - R_k(a,b) \le C^{k+1} \ .$$

PROOF. If $a = b$ then $s(a,a) = R_k(a,a) = 1$ by definition for every $k = 0,1,2,\ldots$, the left-hand side of (4) is zero, and thus (4) obviously holds.

Similarly, if $I(a) = \emptyset$ or $I(b) = \emptyset$ then by definition $s(a,b) = R_k(a,b) = 0$, and the left-hand side of (4) is zero as well.

For the general case of $a \ne b$, $I(a) \ne \emptyset$ and $I(b) \ne \emptyset$, the proof is organized by mathematical induction.

**Induction Basis** Let us prove that (4) holds for $k = 0$, i.e. that for every two nodes $a$, $b$:

$$s(a,b) - R_0(a,b) \le C \ . \tag{5}$$

Since $a \ne b$, $I(a) \ne \emptyset$ and $I(b) \ne \emptyset$, then $R_0(a,b) = 0$ by definition, $s(a,b)$ is defined by the general recursive equation (1), and consequently

$$\begin{aligned} s(a,b) - R_0(a,b) = s(a,b) = \\ = \frac{C}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} \underbrace{s(I_i(a), I_j(b))}_{\le 1} \le \\ \le \frac{C}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} 1 = C \ , \end{aligned}$$

which proves (5).

**Inductive step** Provided that (4) holds for a given $k$ for all node pairs, let us prove that (4) holds for $(k+1)$ as well:

$$\begin{aligned} & s(a,b) - R_{k+1}(a,b) = \\ &= \frac{C}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b)) - \\ & - \frac{C}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_k(I_i(a), I_j(b)) = \\ &= \frac{C}{|I(a)|\,|I(b)|} \times \\ & \times \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} \underbrace{\{s(I_i(a), I_j(b)) - R_k(I_i(a), I_j(b))\}}_{\le C^{k+1} \text{ by inductive hypothesis}} \le \\ &\le \frac{C \cdot |I(a)|\,|I(b)| \cdot C^{k+1}}{|I(a)|\,|I(b)|} = C \cdot C^{k+1} = C^{(k+1)+1} \ . \end{aligned}$$

The latter finally proves (4). □

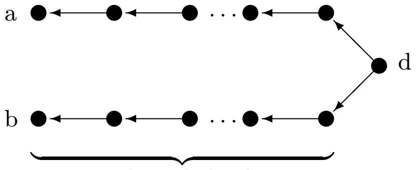

**Figure 1: Illustration of the upper bound stated in Proposition 1 reached:** $R_k(a,b) = 0$, $R_{k+1}(a,b) = C^{k+1}$

As an accompanying result, it follows from the above proposition that $R_k(*,*)$ converges to $s(*,*)$ uniformly.

NOTE 1. *The upper bound stated in Proposition 1 is precise.*

*Indeed, let us consider an arbitrary $k = 0,1,2,\ldots$ and a pair of nodes $a$, $b$ in a graph presented in Fig. 1 for the chosen $k$. In this figure, each of the nodes $a$ and $b$ has an incoming directed path of length $(k+1)$ that starts from some common node $d$. In such a graph configuration, it can easily be seen that $R_k(a,b) = 0$, whereas $R_{k+m}(a,b) = C^{k+1}$, $m = 1,2,3,\ldots$, which gives $s(a,b) = C^{k+1}$. Subtracting $R_k(a,b)$ from $s(a,b)$, we obtain*

$$s(a,b) - R_k(a,b) = C^{k+1} \ ,$$

*which gives the precise upper bound stated in Proposition 1.*

For PageRank iterative computation [13], a condition on whether the current iteration can be considered the last one has to be checked on every iteration: the iterative loop is terminated when the $L_1$ norm between PageRank iterative vectors for two consecutive iterations becomes no greater than the constant $\epsilon$:

$$\|R_{k+1} - R_k\|_1 \le \epsilon \ . \tag{6}$$

On the contrary, Proposition 1 presented in this paper for SimRank allows finding out the precise number of iterations required for achieving a desired accuracy *a-priori*. This observation provides additional freedom for optimizing SimRank computation, the subject of our discussion in the next section.

Proposition 1 shows that the number of iterations $K$ required for achieving a desired accuracy depends on neither the number of nodes in an input graph nor on any other graph characteristics like the degree of nodes. Jeh and Widom observed $K$ independence from an input graph experimentally [7]; Proposition 1 now gives the theoretical foundation for this observation. The observation has an important implication for SimRank computational complexity, as the latter naturally depends on the number of iterations. In accordance with Proposition 1, the number of iterations can now be considered constant with respect to different input graphs.

It is worth noting that Jeh and Widom [7] suggested choosing the decay factor value $C = 0.8$ and the total number of iterations $K = 5$, which in accordance with Proposition 1 stated above imply a relatively large potential difference between theoretical and computed similarity scores:

$$0.8^{5+1} = 0.8^6 > 0.26 \ ,$$

that could be unacceptable for many domains recalling that similarity scores fall into the [0..1] segment. For guaranteeing more accurate computation results, it can be advised using either a smaller decay factor or more iterations. The recommendation for using a smaller decay factor is approved from the viewpoint of similarity scores quality by the experimental results obtained in [3].

Depending on the needs of a particular application, Proposition 1 provides a precise mechanism for finding out (i) either the accuracy value for the given decay factor $C$ and the number of iterations $k$, or (ii) vice versa, the number of iterations required for achieving a desired accuracy. A more detailed discussion on the subject is given in Subsect. 4.3 below.

Proposition 1 provides an estimate for *absolute* difference between theoretical and iterative similarity scores; for ranking purposes, the relative order of nodes with respect to their similarity to a given node is generally more important than absolute scores. From this perspective, the following ranking accuracy estimate can be established.

PROPOSITION 2. *The minimum difference between theoretical similarity scores that allows correctly ranking two nodes with respect to a third node in accordance with their pairwise iterative similarity scores decreases exponentially in the number of iterations. Precisely, if*

$$s(a,b) > s(a,d) + C^{k+1} \ , \tag{7}$$

*then it necessarily follows that*

$$R_k(a,b) > R_k(a,d) \ . \tag{8}$$

PROOF. Let us estimate the difference:

$$\begin{aligned} & R_k(a,b) - R_k(a,d) > \{\text{using (7)}\} > \\ > \ & R_k(a,b) - R_k(a,d) - s(a,b) + s(a,d) + C^{k+1} = \\ = \ & \underbrace{s(a,d) - R_k(a,d)}_{\geq 0} - \underbrace{(s(a,b) - R_k(a,b))}_{\leq C^{k+1} \text{ using (4)}} + C^{k+1} \geq \\ \geq \ & 0 - C^{k+1} + C^{k+1} = 0 \ , \end{aligned}$$

which gives (8). □

The proposition states that the iterative SimRank computation appropriately grabs exponentially smaller differences in theoretical similarity scores.

Propositions 1 and 2 are important not only on their own, establishing an a-priori correlation between the number of iterations $k$ and iterative similarity scores accuracy, but also as a theoretical basis for one of the optimization techniques suggested in the next section.

# 4. OPTIMIZATION TECHNIQUES

In this section, optimization techniques for SimRank computation are suggested.

The optimization techniques cover three consecutive aspects in SimRank computation. Firstly, a technique for selecting essential node pairs is presented, aimed at skipping node pairs that do not require similarity scores computing for a given iteration. Secondly, the notion of partial sums is suggested for reducing the number of access operations to the iterative similarity function and for facilitating efficient clustering. Thirdly, threshold-sieved similarities are introduced for speeding up subsequent iterations.

Each of the three outlined aspects is covered in its own subsection accordingly. In the fourth subsection, the suggested optimization techniques are integrated into a general SimRank optimized computation algorithm.

## 4.1 Selecting Essential Node Pairs

In this subsection, techniques for selecting essential node pairs are suggested for not computing iterative similarity scores for the remaining node pairs while preserving the correct semantics of SimRank iterative similarity functions.

*Definition 1.* Let *essential paired nodes* for a given node $a$ over a similarity function $R_k(*, *)$ denote all nodes from the following set of nodes[1]:

$$\begin{aligned} & \text{Essential}_{R_k}(a) = \\ & = \{b \mid \exists u \in I(a), \exists v : R_k(u,v) \neq 0, \ \ b \in O(v)\} \ . \end{aligned} \tag{9}$$

The reason for referring to nodes from a thus defined set as "essential paired nodes" for a given node becomes clear from the following proposition.

PROPOSITION 3. *For a given node $a$, $R_{k+1}(a,b)$ is zero for every node $b$ such that*

$$b \notin \{a\} \cup \mathit{Essential}_{R_k}(a) \ . \tag{10}$$

PROOF (BY CONTRADICTION). Suppose that $R_{k+1}(a,b)$ is non-zero for some node $b$ from (10). Since it follows from (10) that $b \neq a$, and since $I(a) \neq \emptyset$ and $I(b) \neq \emptyset$ in accordance with our supposition $R_{k+1}(a,b) \neq 0$, then $R_{k+1}(a,b)$ is computed by general iterative formula (3). As all items in (3) are non-negative, it follows from $R_{k+1}(a,b) \neq 0$ that there exists at least a single pair of indexes $(i_0, j_0)$ such that $R_k(I_{i_0}(a), I_{j_0}(b)) \neq 0$. Let us denote $I_{i_0}(a) = u$, $I_{j_0}(b) = v$.

By their notation, $u \in I(a)$, $v \in I(b)$. By symmetry, it follows from $v \in I(b)$ that $b \in O(v)$. Thus $b \in \text{Essential}_{R_k}(a)$, since the chosen nodes $u$ and $v$ are the ones that satisfy (9). The latter result leads to a contradiction with the initial terms of the proposition. □

Definition 1 and Proposition 3 provide an algorithm for considering only essential node pairs and thus skipping iterative scores computation for the remaining ones. From the

[1] Recall that notation $O(v)$ denotes the set of all out-neighbour nodes of node $v$, as mentioned in Sect. 2.

computational viewpoint, the set of essential paired nodes $\text{Essential}_{R_k}(a)$ for a given node $a$ can be obtained by first constructing a temporary set of nodes $\text{Temp}_{R_k}(a)$ that consists of all nodes having non-zero similarity scores with some $I(a)$ member:

$$\text{Temp}_{R_k}(a) = \{v \mid \exists u \in I(a) : \;\; R_k(u, v) \neq 0\} \;\; . \tag{11}$$

It can easily be verified that the set of essential paired nodes for $a$ can then be obtained by taking all out-neighbours for every node in $\text{Temp}_{R_k}(a)$, i.e.:

$$\text{Essential}_{R_k}(a) = \{b \mid \exists v \in \text{Temp}_{R_k}(a) : b \in O(v)\} \;\; . \tag{12}$$

Computational complexity for both (11) and (12) for a given node $a$ is quadratic in the number of nodes in a graph, intermediate memory consumption is linear. As a single SimRank iteration generally involves calculating (11) and (12) for *every* node $a$ in the graph, their computational complexity is $O(n^3)$ for a single iteration. Memory consumption remains linear, since $\text{Temp}_{R_k}(a)$ can be freed after $\text{Essential}_{R_k}(a)$ is constructed, and $\text{Essential}_{R_k}(a)$ can be freed after essential paired nodes for node $a$ are processed. Further discussion in this section shows that computational complexity is no more than $O(n^3)$ for subsequent processing within a SimRank iteration as well.

Essential node pairs provide a better selectivity when the iterative similarity function $R_k(*, *)$ has a relatively small fraction of non-zero values with respect to zero ones. We will refer to such a similarity function as a *sparse* one. A technique for keeping an intermediate iterative similarity function sparse is suggested in Subsect. 4.3.

Iterative similarity scores computation can be skipped not only for node pairs with a-priori zero scores, but also for the ones that are not required for a subsequent iterative computation.

PROPOSITION 4. *If a node $u$ in a graph has no outgoing links, then $R_{k+1}(*, *)$ does not depend on $R_k(u, *)$.*

PROOF (BY CONTRADICTION). Suppose that there exists a pair of nodes $a, b \in V$ such that $R_{k+1}(a, b)$ depends on $R_k(u, *)$. Note that $a \neq b$, $I(a) \neq \emptyset$ and $I(b) \neq \emptyset$, since in a otherwise case $R_{k+1}(a, b)$ is constant by definition and thus does not depend on $R_k(u, *)$.

It follows from $a \neq b$, $I(a) \neq \emptyset$ and $I(b) \neq \emptyset$ that $R_{k+1}(a, b)$ is calculated by the general iterative formula (3). Since $R_k(*, *)$ is presented in the right-hand side of (3) in the form of $R_k(I_i(a), I_j(b))$, the only way for $R_{k+1}(a, b)$ to depend on $R_k(u, *)$ is to have $u = I_{i_0}(a)$ for some index $i_0$, i.e. to have $u \in I(a)$. But $u \in I(a)$ implies by symmetry that $a \in O(u)$. The latter contradicts the proposition terms on $O(u) = \emptyset$. □

COROLLARY 1. *If a node $u$ in a graph has no outgoing links, then it is sufficient to calculate $R_k(u, *)$ on just the last iteration without violating the semantics of SimRank iterative computation.*

An a-priori knowledge of the precise number of iterations provided by Proposition 1 plays the crucial role for the practical applicability of Corollary 1. If the number of iterations were unknown a-priori, all non-zero similarity scores would have had to be computed anyway for the reason of finding out when to terminate the iterative computation—recall (6) in PageRank approach.

From the computational viewpoint, checking the applicability of Corollary 1 for a given node $u$ can be performed in constant time and requires no additional memory, and thus constitutes a practical pruning mechanism.

While Proposition 3 is focused on pruning node pairs with zero similarity from consideration, Proposition 4 and Corollary 1 cover a different pruning aspect. Indeed, even if a node $u$ has no outgoing links, it can still have many incoming links and thus have a non-zero similarity with many other nodes in a graph. Moreover, calculating all these similarity scores can involve a large computational effort. However, if the node $u$ has no outgoing links, then this computational effort can be saved on intermediate iterations and performed for the last iteration only. We will show the practical importance of this proposition when considering the experimental results.

## 4.2 Partial Sums

After essential node pairs for a given node are selected, optimization technique presented in this subsection allows reducing the number of access operations to $R_k(*, *)$ required for computing $R_{k+1}(*, *)$. The main idea behind the optimization is that a sum of $R_k(*, *)$ values over a certain set of arguments is used for computing *several* values of $R_{k+1}(*, *)$ and can thus be effectively memoized for preventing repeated computation.

For an elaborate discussion on the subject, let us first introduce the notion of a *partial sums function.*

*Definition 2.* Let $f(*, *)$ be a binary function $X \times Y \to R$ and let $S$ be a finite subset in $X$: $S = \{x_1, x_2, \ldots, x_p\}$, $x_i \in X$, $i \in \overline{1,p}$. By partial sums for the function $f$ over the set $S$ we will call a unary function $Y \to R$ denoted as $\text{Partial}^f_S(*)$ and defined as follows:

$$\text{Partial}^f_S(y) = \sum_{x_i \in S} f(x_i, y) \;\; , \;\; y \in Y \;\; .$$

A partial sums function is introduced for being applied to SimRank iterative similarity scores computation:

PROPOSITION 5. *For $a \neq b$, $I(a) \neq \emptyset$ and $I(b) \neq \emptyset$, $R_{k+1}(a, b)$ can be computed iteratively as*

$$R_{k+1}(a, b) = \frac{C}{|I(a)||I(b)|} \sum_{j=1}^{|I(b)|} \mathit{Partial}^{R_k}_{I(a)}(I_j(b)) \;\; . \tag{13}$$

PROOF. The proposition is proved by simply swapping the summation signs in the iterative formula (3):

$$R_{k+1}(a, b) = \frac{C}{|I(a)||I(b)|} \sum_{j=1}^{|I(b)|} \underbrace{\sum_{i=1}^{|I(a)|} R_k(I_i(a), I_j(b))}_{\text{Partial}^{R_k}_{I(a)}(I_j(b))}$$

and by noting that the internal summation is the value of the partial sums function for $R_k(*, *)$ over $I(a)$ for argument $I_j(b)$. □

Although trivial in its proof, Proposition 5 provides the efficient speedup technique for SimRank computation, based on the following corollary:

COROLLARY 2. *For a given fixed node $a$, the same partial sums function $\mathit{Partial}^{R_k}_{I(a)}(*)$ is used for computing $R_{k+1}(a, b)$ for every node $b$ in a graph.*

The key point in optimizing $R_{k+1}(a,*)$ computation via a partial sums function $\text{Partial}_{I(a)}^{R_k}(*)$ is that once calculated partial sums values are memoized and are thus not recalculated when subsequently required. For example, if $I_j(b) = I_l(d) = u$ for some nodes $b$ and $d$, then the partial sum value $\text{Partial}_{I(a)}^{R_k}(u)$ is calculated once and is used in both $R_{k+1}(a,b)$ and $R_{k+1}(a,d)$ computation. Since for some nodes $w$, the partial sum values $\text{Partial}_{I(a)}^{R_k}(w)$ will probably not be required for computing the $R_{k+1}(a,*)$ values, it is reasonable to calculate the partial sums function in a delayed fashion.

Analyzing computational complexity, the straightforward iterative SimRank computation involves $|I(a)| \cdot |I(b)|$ access operations to $R_k(*,*)$ for computing $R_{k+1}(a,b)$ for a single pair of nodes $(a,b)$, resulting in $n^2 \text{avg}_{a,b}(|I(a)||I(b)|)$ operations per iteration [7], being $O(n^4)$ in the worst case [17]. For comparison, computing a single partial sum value involves $|I(a)|$ access operations to $R_k(*,*)$; and at most $n$ partial sums values are required for computing $R_{k+1}(a,*)$ for a fixed node $a$ as the first argument and *every* other node as the second one. The latter approach requires only $O(n^3)$ operations per iteration in the worst case, which is a definite improvement achieved by partial sums.

Partial sums allow additionally speeding up SimRank computation by $R_k(*,*)$ values clustering. Precisely, before partial sums were introduced, computing $R_{k+1}(a,b)$ for a single pair of argument nodes generally required accessing $R_k(*,*)$ for a Cartesian product of $I(a) \times I(b)$; such argument values spread hardly made any clustering strategy provide access time speed up. For comparison, partial sums usage allows clustering the underlying storage for $R_k(*,*)$ values by the first argument. Indeed, due to $R_k(*,*)$ symmetry, $\text{Partial}_{I(a)}^{R_k}(u)$ can be computed from $R_k(u,*)$ involving a single node for the first argument:

$$\text{Partial}_{I(a)}^{R_k}(u) = \sum_{i=1}^{|I(a)|} R_k(I_i(a), u) = \sum_{i=1}^{|I(a)|} R_k(u, I_i(a)) \ .$$

Moreover, all other operations over $R_k(*,*)$ required throughout this paper for SimRank computation use batch access to $R_k(u,*)$, for a fixed first argument node and a *set* of second argument nodes:

- Essential node pairs selection facilitates obtaining all essential paired nodes for a given node $a$; calculating the set of essential paired nodes in accordance with (11) involves obtaining *all* non-zero values $R_k(u,*)$ for a given first argument node $u$.
- Corollary 2 facilitates computing $R_{k+1}(a,*)$ for a given first argument node and *all* second argument nodes from the essential paired nodes set before proceeding to another node as the first argument.

The above made observations allow achieving access operations speed up by clustering $R_k(*,*)$ by the first argument from the underlying storage viewpoint.

## 4.3 Threshold-Sieved Similarity

For certain graphs classes like scale-free graphs [8], the iterative similarity function $R_k(*,*)$ has an abundance of non-zero values after just a few iterations. However, many of these values, although being non-zero, denote low similarity between node pairs and thus contain little practical information for the result similarity scores. On the other hand, keeping all these small but nevertheless non-zero similarity scores requires considerable storage amounts and slows down subsequent iterations.

Since scale-free graphs constitute the underlying representation for many practical corpora including Wikipedia [18] and the Web, we propose the notion of a threshold-sieved similarity function for effectively handling desired similarity scores. The necessary theoretical results are presented to ensure that a threshold-sieved similarity function provides a user-controlled effect over the result similarity scores.

Let us choose some non-negative parameters $\delta_1, \delta_2, \ldots, \delta_K$, where each $\delta_k$ is treated as a threshold for iterative similarity scores on the $k$-th iteration. Conceptually, similarity score for a pair of nodes $(a,b)$ on the $k$-th iteration will be treated as zero if this value is not greater than the threshold $\delta_k$, and the thus formed threshold-sieved similarity score for $(a,b)$ was zero for the previous iteration as well.

Formally, let us define a threshold-sieved iterative similarity function $R_k^{\delta_k}(*,*)$ over a set of threshold parameters $\{\delta_k\}$ as follows:

$$\begin{aligned}
R_0^{\delta_0}(a,b) &= R_0(a,b) \ ; \\
R_{k+1}^{\delta_{k+1}}(a,a) &= R_{k+1}(a,a) = 1 \ ; & (14) \\
R_{k+1}^{\delta_{k+1}}(a,b) &= 0 \ , \text{ if } I(a) = \emptyset \text{ or } I(b) = \emptyset \ ; & (15) \\
R_{k+1}^{\delta_{k+1}}(a,b) &= \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_k^{\delta_k}(I_i(a), I_j(b)) \ , \\
&\quad \text{if either (right-hand side} > \delta_{k+1}) \\
&\quad \text{or } R_k^{\delta_k}(a,b) \neq 0 \ ; & (16) \\
R_{k+1}^{\delta_{k+1}}(a,b) &= 0 \ , \text{ otherwise} \ . & (17)
\end{aligned}$$

In $R_{k+1}^{\delta_{k+1}}(a,b)$ definitions (15) – (17), $a$ and $b$ are assumed to be different nodes; when $a$ and $b$ are the same node, the definition of $R_{k+1}^{\delta_{k+1}}(a,a)$ is given separately in (14).

For textually distinguishing $R_k(*,*)$ and $R_k^{\delta_k}(*,*)$, we will further refer to $R_k(*,*)$ as *conventional* iterative similarity.

It can easily be proven by mathematical induction that conventional similarity function is an upper bound for a threshold-sieved one, i.e.

$$R_k^{\delta_k}(a,b) \leq R_k(a,b) \ , \ \forall a,b \ , \ \forall k \ . \tag{18}$$

Moreover, the following estimate for threshold-sieved iterative similarity function with respect to conventional iterative similarity function can be established:

PROPOSITION 6. *For every iteration $k = 0,1,2,\ldots$ and for every two nodes $a,b \in V$ the following estimate holds:*

$$R_k(a,b) - R_k^{\delta_k}(a,b) \leq \Delta \ ,$$

*where*

$$\Delta = \sum_{m=1}^{k} C^{k-m} \delta_m \ . \tag{19}$$

PROOF. For $a = b$, $I(a) = \emptyset$ or $I(b) = \emptyset$, the same reasoning as for Proposition 1 applies. We thus further consider $a \neq b$, $I(a) \neq \emptyset$ and $I(b) \neq \emptyset$ and prove the proposition by induction over the iteration number $k$.

**Induction Basis** For $k = 0$, the estimate obviously holds, as $R_0(a,b) - R_0^{\delta_0}(a,b) = 0$.

**Inductive Step** During the inductive step, we refer to $\Delta$ as $\Delta(k)$ when stressing that summation is performed from 1 to $k$, and as $\Delta(k+1)$ when stressing that summation is performed from 1 to $(k+1)$.

Provided that the proposition holds for $k$, let us estimate the difference $R_{k+1}(a,b) - R_{k+1}^{\delta_{k+1}}(a,b)$ for $(k+1)$.

Two possible cases will be considered separately: the one for $R_{k+1}^{\delta_{k+1}}(a,b) = 0$ and the other for $R_{k+1}^{\delta_{k+1}}(a,b) \neq 0$:

1. If $R_{k+1}^{\delta_{k+1}}(a,b) = 0$, then it follows from (16) and (17) that

$$\frac{C}{|I(a)||I(b)|}\sum_{i=1}^{|I(a)|}\sum_{j=1}^{|I(b)|} R_k^{\delta_k}(I_i(a),I_j(b)) \le \delta_{k+1} \ , \tag{20}$$

and the difference $R_{k+1}(a,b) - R_{k+1}^{\delta_{k+1}}(a,b)$ is estimated thus:

$$\begin{aligned}
& R_{k+1}(a,b) - R_{k+1}^{\delta_{k+1}}(a,b) = R_{k+1}(a,b) \le \\
\le\ & \{\text{using (20)}\} \le R_{k+1}(a,b) + \delta_{k+1} - \\
& -\frac{C}{|I(a)|\,|I(b)|}\sum_{i=1}^{|I(a)|}\sum_{j=1}^{|I(b)|} R_k^{\delta_k}(I_i(a),I_j(b)) = \\
=\ & \delta_{k+1} + \frac{C}{|I(a)|\,|I(b)|}\sum_{i=1}^{|I(a)|}\sum_{j=1}^{|I(b)|}\{ \\
& \underbrace{R_k(I_i(a),I_j(b)) - R_k^{\delta_k}(I_i(a),I_j(b))\}}_{\le\Delta(k)\text{ by inductive hypothesis}} \le \\
\le\ & \delta_{k+1} + C\sum_{m=1}^{k} C^{k-m}\delta_m = \Delta(k+1) \ .
\end{aligned}$$

2. Otherwise $R_{k+1}^{\delta_{k+1}}(a,b) \neq 0$, and thus it is defined by (16) and consequently

$$\begin{aligned}
& R_{k+1}(a,b) - R_{k+1}^{\delta_{k+1}}(a,b) = \\
=\ & \frac{C}{|I(a)|\,|I(b)|}\sum_{i=1}^{|I(a)|}\sum_{j=1}^{|I(b)|}\{ \\
& \underbrace{R_k(I_i(a),I_j(b)) - R_k^{\delta_k}(I_i(a),I_j(b))}_{\le\Delta(k)}\ \} \le \\
\le\ & C\sum_{m=1}^{k} C^{k-m}\delta_m \le \Delta(k+1) \ .
\end{aligned}$$

The latter finishes the induction. □

The parameter $\Delta$ is intended as a user control over maximum potential difference between threshold-sieved and conventional iterative similarity functions, and thus $\Delta$ is generally chosen by a user. Provided that $\delta_1, \delta_2, \ldots, \delta_k$ are selected to fulfill (19), Proposition 6 states that difference between threshold-sieved and conventional similarity scores does not exceed $\Delta$.

Note from the proposition proof that the equation (19) gives the worst-case upper bound. In practice, we noted the differences between $R_k(*,*)$ and $R_k^{\delta_k}(*,*)$ for $k = 5$ being smaller than $\Delta$ by one order of magnitude.

It should be noted that if $\Delta$ is chosen to be zero, then $\delta_1 = \delta_2 = \ldots = \delta_K = 0$ and a threshold-sieved iterative similarity function $R_k^{\delta_k}(*,*)$ becomes a conventional iterative similarity function $R_k(*,*)$. From this perspective, a threshold-sieved similarity can be considered as a generalization for a conventional similarity.

One of the possible ways for choosing threshold parameters $\delta_1, \delta_2, \ldots, \delta_k$ is to specify every iteration make an equal contribution to the $\Delta$ value:

$$C^{k-m}\delta_m = \frac{\Delta}{k} \ , \quad m = \overline{1,k} \ ,$$

which gives

$$\delta_m = \frac{\Delta}{kC^{k-m}} \ , \quad m = \overline{1,k} \ . \tag{21}$$

Due to general commonalities between a threshold-sieved and a conventional similarity functions, previously stated Propositions 3 and 4 straightforwardly apply to threshold-sieved similarity function. With partial sums introduced in the previous subsection, equations (16) and (17) are rewritten correspondingly as:

$$\begin{aligned}
R_{k+1}^{\delta_{k+1}}(a,b) &= \frac{C}{|I(a)||I(b)|}\sum_{j=1}^{|I(b)|}\text{Partial}_{I(a)}^{R_k^{\delta_k}}(I_j(b)) \ , \\
&\quad \text{if either (right-hand side} > \delta_{k+1}) \\
&\quad \text{or } R_k^{\delta_k}(a,b) \neq 0 \ ; & (22) \\
R_{k+1}^{\delta_{k+1}}(a,b) &= 0 \ , \text{ otherwise} \ . & (23)
\end{aligned}$$

The combination of Propositions 1 and 6 provides the upper bound for a maximum potential difference between a threshold-sieved similarity function and the theoretical similarity function:

Proposition 7. *For every pair of nodes $(a,b)$ and for every iteration number $k = 0, 1, 2, \ldots$, the following estimate holds:*

$$s(a,b) - R_k^{\delta_k}(a,b) \le \epsilon \ ,$$

*where*

$$\epsilon = C^{k+1} + \Delta \ , \tag{24}$$

*and threshold parameters $\delta_1, \delta_2, \ldots, \delta_k$ are chosen with respect to $\Delta$ as specified by* (19).

Proof. The proposition immediately follows from Propositions 1 and 6. □

In combination with (18) and SimRank Property 2, the estimate stated in Proposition 7 has the following form:

$$0 \le s(a,b) - R_k^{\delta_k}(a,b) \le \epsilon \ .$$

Based on Proposition 7, the analogue of Proposition 2 for ranking accuracy estimate can be straightforwardly stated for a threshold-sieved similarity function, replacing $C^{k+1}$ with $\epsilon$.

A threshold-sieved iterative similarity function $R_k^{\delta_k}(*,*)$ may at first seem as an approximation for a conventional iterative similarity function $R_k(*,*)$; however, Proposition 7 shows that both actually follow the same nature of uniformly converging to the theoretical similarity function $s(*,*)$ for

$k \to \infty$, $\Delta \to 0$. When a user wishes to achieve a desired accuracy $\epsilon$, she or he is free to follow on of the two possible options. The user can turn threshold sieving off by specifying $\Delta = 0$, which in accordance with (24) would result in a fewer number of iterations $K$ required for achieving accuracy $\epsilon$:

$$K = \lceil \log_C \epsilon \rceil - 1 \ .$$

Alternatively, the user can perform one more iteration and assign additional accuracy tolerance to $\Delta$:

$$\begin{aligned} K &= \lceil \log_C \epsilon \rceil \ ; \\ \Delta &= \epsilon - C^{K+1} \ . \end{aligned}$$

As our experimental results presented in Sect. 6 show, the latter option provides a faster computation due to a more freedom for applying the suggested optimization techniques, even though the additional SimRank iteration is performed.

### 4.4 The Optimized SimRank Computation Algorithm

All optimization techniques suggested in the previous subsections are collected within Algorithm 1 for SimRank optimized computation. As different strategies for choosing $K$ and $\Delta$ for achieving the desired accuracy can be applied, the algorithm accepts both parameters as input.

**Algorithm 1** SimRank optimized computation

```
Input: $G(V,E)$, $C$, $K$, $\Delta$
Output: $R_K^{\delta_K}(*,*)$
 1: Calculate $\delta_1$, $\delta_2$, ..., $\delta_K$
 2: Initialize $R_0(*,*)$
 3: for $k = 0$ to $K-1$ do
 4:   for all $a \in V$ do
 5:     if $O(a) = \emptyset$ and $k \neq K-1$ then
 6:       Continue for next $a$
 7:     end if
 8:     Initialize $\mathrm{Partial}_{I(a)}^{R_k}(*)$
 9:     Calculate $\mathrm{Essential}_{R_k}(a)$
10:     for all $b \in \{a\} \cup \mathrm{Essential}_{R_k}(a)$ do
11:       Calculate $R_{k+1}^{\delta_{k+1}}(a,b)$
12:     end for
13:     Free $\mathrm{Essential}_{R_k}(a)$
14:     Free $\mathrm{Partial}_{I(a)}^{R_k}(*)$
15:   end for
16: end for
```

As algorithm statements are described in high-level terms, the following list collects references to underlying formulae and theoretical justifications given earlier in the paper and implied in a corresponding algorithm line:

- In line 1, each $\delta_m$ can be calculated by equation (21). A different strategy for choosing $\delta_1$, $\delta_2$, ..., $\delta_K$ can be used; the only requirement is that (19) be fulfilled.
- In line 2, $R_0(*,*)$ is defined by (2).
- In lines 5 – 7, the conditional expression is justified by Corollary 1.
- In line 9, $\mathrm{Essential}_{R_k}(a)$ is calculated by (11), (12).
- In line 10, the condition in the header of the for loop is justified by Proposition 3.
- In line 11, $R_{k+1}^{\delta_{k+1}}(a,b)$ is calculated by (14), (15), (22) and (23).
- In lines 13 and 14, the free statement is used to denote that $\mathrm{Essential}_{R_k}(a)$ and $\mathrm{Partial}_{I(a)}^{R_k}(*)$ are not required for further computation and can be dropped.

Collecting complexity analysis for all suggested optimization techniques, it follows that computing $R_{k+1}^{\delta_{k+1}}(a,*)$ requires quadratic time and linear intermediate memory in the number of nodes in a graph in the worst case. Performing a single iteration for all node pairs in a graph thus requires $O(n^3)$ time and $O(n)$ intermediate memory; intermediate memory consumptions are negligible with respect to storage involved in keeping computed similarity scores.

A complete sequence of iterations consequently has computational complexity of $O(Kn^3)$. As it was shown in Propositions 1 and 7 that the number of iterations $K$ required for achieving the desired accuracy does not depend on the number of nodes $n$ in the graph, we finally obtain that SimRank computation in the presence of the suggested optimization techniques is $O(n^3)$.

## 5. RELATED WORK

Due to practical importance of measuring object-to-object similarity, different approaches to defining similarity measures were suggested in scientific literature, e.g. the ones based on domain hierarchies [5], information theory [9], network flow computation [11]. With respect to the focus of this paper, a detailed discussion is given to related work that either correlate with SimRank or present similarity measures with complexity analysis claimed applicable for large data corpora.

Xi et al. suggested a similarity-calculating algorithm called *SimFusion* that aims at "combining relationships from multiple heterogeneous data sources" [17]. The basic intuition behind SimFusion approach somewhat resembles the one for SimRank: "the similarity between two data objects can be reinforced by the similarity of related data objects from the same and different spaces" [17]. SimFusion provides the following extensions with respect to SimRank: (i) support for different kinds of intra-nodes relations, e.g. outgoing links, content commonality; (ii) support for different weights associated with different kinds of relations; (iii) support for several information spaces.

Iterative similarity computation formula for SimFusion has much in common with the one for SimRank. Indeed, with $L_{urm}$ being a row-stochastic matrix that combines all the relationships between nodes, SimFusion reinforcement assumption is reified as follows:

$$S_{usm}^k = L_{urm} S_{usm}^{k-1} L_{urm}^T \ , \qquad (25)$$

where $S_{usm}$ is a "unified similarity matrix" that represents similarity values between node pairs [17]. Let us denote a row in $L_{urm}$ that corresponds to node $a$ as $L_{urm}(a)$, and a matrix element in $S_{urm}^k$ that corresponds to a pair of nodes $(a,b)$ as $S_{usm}^k(a,b)$. If we then consider node relations of the kind "has an incoming link" and treat all incoming links as of an equal priority, then $L_{urm}(a)$ contains $\frac{1}{|I(a)|}$ in column $I_i(a)$, $i = \overline{1,n}$ and zeroes in all remaining columns. In accordance with (25), SimFusion iterative similarity value between nodes $a$ and $b$ thus takes the form:

$$S^k_{usm}(a,b) = L_{urm}(a) S^{k-1}_{usm}(L_{urm}(b))^T = \\ = \frac{1}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} S^{k-1}_{usm}(I_i(a), I_j(b)) \ , \quad (26)$$

which is the same as for SimRank iterative formula minus the decay factor $C$. Unlike SimFusion, SimRank similarity score for any node with itself always equals to 1, and these initial similarity scores are iteratively propagated to the other node pairs. In SimFusion, initial similarity scores are redistributed in a flow fashion through node relations, and thus node similarity with itself may not be equal to 1.

If a node has no incoming links, it has zero SimRank similarity score with every other node except for itself. In SimFusion, a node with no relationship to the other nodes in data space has each element in the corresponding row of relationship matrix set to $1/n$ for preventing similarity sinks. This treatment has the same effect as introducing a source of rank in PageRank [13].

In spite of the noted differences between SimRank and SimFusion, equation (26) shows that the overall iterative formulae for SimRank and SimFusion have much resemblance to each other. Consequently, some of the optimization techniques presented in this paper should apply to SimFusion computation as well, e.g. essential node pairs selection in terms of sparse matrices. We also believe that there should exist an analogous accuracy estimate for SimFusion as the one revealed for SimRank.

Computational complexity for SimFusion is $O(Kn^3)$ in the worst case [17]. Cubic computational complexity in the number of nodes for SimFusion directly follows from matrix representation (25); as for SimRank, the double summation in the iterative formula (3) misleaded SimRank computational complexity, which was previously considered $O(Kn^4)$ in the worst case [7, 17]. The idea of partial sums suggested in this paper fixes this inconsistency between SimRank and SimFusion, for now both are known to have cubic computational complexity in the number of nodes.

Fogaras and Rácz [3] suggested a scalable framework for SimRank computation based on Monte Carlo method. The main idea of their approach is to generate reversed random walks for each node in a graph, calculate the first meeting time $\tau_{a,b}$ for a pair of random walks started in nodes $a$ and $b$, and estimate $s(a,b)$ by $C^{\tau_{a,b}}$. In their work, Fogaras and Rácz suggest several excellent ideas, in particular, fingerprint trees, random permutations on graph nodes for effectively generating coupled random walks, parallelization possibilities for SimRank computation under their framework. The probabilistic approach they took allowed them significantly reduce the computational complexity and create a framework for similarity computation scalable enough for performing SimRank precomputation phase for a graph with 79M nodes. Fogaras and Rácz provide a solid theoretical basis for approximations they make and a (probabilistic) error estimate for their Monte Carlo similarity function. Valuable experimental results are obtained for path length and the decay factor value; surprisingly however, Monte Carlo similarity was not compared with iterative SimRank similarity from the scores quality perspective. The differences in our approaches are that Fogaras and Rácz initially base their framework on Monte Carlo method, and thus their computation is inherently probabilistic, whereas our work is focused on iteratively computing the exact similarity scores.

For improving time and space requirements for SimRank computation, Jeh and Widom [7] suggested pruning the logical graph $G^2$. Their proposal is to "set the similarity between two nodes far apart to be 0, and consider node-pairs only for nodes which are near each other" [7]. This technique is based on the assumption that "it is very likely that the neighborhood (say, nodes within a radius of 2 or 3) of a typical node will be a very small percentage ($< 1\%$) of the entire domain" [7]. Firstly, this assumption does not hold for scale-free graphs, as these have a very small average distance (or diameter) between nodes [2]. As our early experimental studies over practical scale-free graphs showed, even the neighbourhood within the radius of 2 contains a considerable proportion of graph nodes.

Secondly, pruning graph $G^2$ in the suggested way is an approximation; Jeh and Widom provide no theoretical argument about the error of approximating [3], but admit that "the quality of the approximation needs to be verified" [7]. Finally, removing node pairs not near each other from consideration undermines the basic SimRank design principle of being a generalization to a conventional immediate in-neighbours analysis [15]. As verified by Fogaras and Rácz in their experiments, "the multi-step neighborhoods of pages[2] contain valuable similarity information" [3].

For comparison, optimization techniques suggested in this paper are free from the above mentioned drawbacks. Selecting essential node pairs does not affect iterative similarity scores. Threshold sieving has a controllable effect over iterative scores precision that can be restored by performing an additional iteration. The techniques do not decrease the radius of a node neighbourhood considered for similarity scores computation.

Maguitman et al. [12] introduce an information-theoretic measure of *semantic similarity* that exploits human-generated topical directories metadata and relies on "both hierarchical and non-hierarchical structure of an ontology" [12]. Maguitman et al. suggest a flexible mechanism for extending the previously existing tree-based semantic similarity measures to a graph-based semantic similarity; however, their results are based on the assumption that a graph necessarily has a "hierarchical (tree) component" $T$.

Maguitman et al. claim computational complexity $O(n^3)$ for their semantic similarity measure for $n$ topics [12]; however, that is computational complexity for just a single matrices product $A \odot B$ used in their reasoning. Precisely, semantic similarity measure requires the closure matrix $T^+$ computed; the latter is defined as $T^+ = \lim_{r\to\infty} T^{(r)}$, with $T^{(r+1)} = T \odot T^{(r)}$, $T^{(0)} = T$. Consequently, obtaining $T^+$ alone implies computational complexity $O(n^3 h)$, where $h$ is "the maximum depth of the tree $T$" [12]. Generally, $h$ depends on $n$, and the worst case computational complexity for $T^+$ is actually $O(n^4)$, not $O(n^3)$.

Maguitman et al. required significant computational and storage resources for computing similarity scores for their semantic similarity measure for a data corpus consisting of 0.5M topics containing totally 1.23M pages [12]. For comparison, our claim is that SimRank optimization techniques suggested in this paper allow computing SimRank similarity scores on a desktop machine in reasonable time even for a larger data corpora, as our experimental results illustrate.

[2] The term "page" there is used in the same semantics as the term "node" throughout this paper.

Using once computed semantic similarity scores as a baseline, Maguitman et al. introduce and compare several approximation measures. Notably, link-based similarity measures systematically produced better correlation with semantic similarity measure and correspondingly with human judgments compared to text-based measures [12]. This important result can in particular serve as an approval for SimRank being a purely link-based similarity measure.

Lin et al. [10] suggest a similarity measure based on PageRank scores propagation through link paths. With $r$ standing for the propagation radius and $d$ for an average node degree, finding similar nodes to a given node with respect to that measure has computational complexity of $O(d^{2r})$, which we believe being too ineffective for on-line computation assumed in [10].

Geerts et al. [6] introduce the concept of a database graph for expressing relationships between partial tuples in a relational database and explore methods for ranking partial tuples. Defining similarity measures in a database graph is pointed out as an interesting question for future work. We believe that SimRank can be a candidate measure for this domain, since a database graph contains enough information for computing SimRank similarity scores for partial database tuples, and the suggested optimization techniques provide viable computational complexity.

# 6. EXPERIMENTAL RESULTS

In this section, experimental results are presented for illustrating the practical quantitative effect of applying the optimization techniques presented in this paper.

We implemented a prototype that provides SimRank similarity scores computation and incorporates the suggested optimization techniques in accordance with algorithm 1 given in Sect. 4. From the implementation perspective, each graph node is identified by a distinct non-negative integer. Iterative SimRank similarity function for each iteration is represented by a square matrix, with a matrix element standing for the similarity score between nodes identified by the corresponding row and column numbers. From the storage perspective, a matrix is sparse, in that constant 1s across the main diagonal and zero similarity scores are not stored. Matrix storage in external memory is implemented on top of Oracle Berkeley DB[3]. As noted in Subsect. 4.2, the suggested optimization techniques facilitate clustering similarity function values by the first argument, corresponding to clustering a matrix by rows in matrix terms. For each row number, the associated (column_number, similarity_score) pairs are stored adjacently. Each of the mappings $I(v)$ and $O(v)$ is implemented as an association between a node identifier for $v$ and a list of node identifiers for the corresponding in-neighbours and out-neighbours respectively.

Two kinds of experiments are reported in this section: one for generated graphs and the other for the English Wikipedia corpus.

## 6.1 Experiment over Generated Graphs

The purpose of the experiment over generated graphs is to investigate the dynamics in SimRank computation time with respect to the number of nodes in a graph and particular optimization techniques employed. We have chosen scale-free graphs for this experiment, because scale-free graphs have a very small diameter [2], and a node thus generally has non-zero similarity scores with a significant proportion of other nodes in the graph after several SimRank iterations, rather than with a fixed number of nodes. The described feature of a scale-free graph thus allows investigating a pessimistic case in SimRank computation, rather than an optimistic case.

| Number of nodes in a graph | Computation time, seconds | | | |
|---|---|---|---|---|
| | No optimization | Selecting essential node pairs | turn on Partial sums | turn on Threshold sieving, $K = 6$ |
| 1000 | 42 | 11 | 2 | 2 |
| 2000 | 348 | 157 | 25 | 13 |
| 5000 | 8061 | 5181 | 588 | 309 |
| 10000 | 165902 | 131675 | 8145 | 2799 |

**Table 1: SimRank computation time w.r.t. the number of nodes in a graph. For each subsequent column, another optimization technique is turned on. For the last column, $\Delta < C^6 - C^7$.**

Two sets of generated graphs were used: one set was produced by scale-free graph generator[4] and the other – by XMark generator [14]. For the purposes of the experiment, the implementation was made configurable, with the ability of turning each optimization technique on and off at compile-time. For properly taking access operations time into account, unbiased by cache speed, the cache size in Oracle Berkeley DB was chosen proportionally to the number of nodes in a graph and sufficient for keeping just several matrix rows. The following machine configuration was used: 2.1GHz Intel Pentium processor, 1Gb RAM and Linux OS.

The averaged computation time with respect to the number of nodes in a graph and the particular optimization techniques used is shown in Table 1. For a correspondence with experiment conditions performed by Jeh and Widom [7], the decay factor $C$ was chosen as 0.8; the number of iterations $K$ was set to 5 for all table columns except for the last one. First computing SimRank with no optimization, optimization techniques are consequently turned on one by one for each subsequent column in Table 1, thus illustrating the speedup achieved by each individual optimization technique. For the last column, six iterations were performed instead of five, with threshold sieving turned on for $\Delta$ chosen as $\Delta = 0.05 < C^6 - C^7$, which in accordance with Proposition 7 gives the same total accuracy of computed similarity scores as for all the remaining columns.

Two important conclusions can be drawn from the experimental results presented in Table 1. Firstly, even for a relatively small graph sizes considered, using partials sums exposes radical speedup that fully corresponds with the theoretical expectations presented in Sect. 4. Secondly, SimRank computation benefits from using threshold-sieved similarity functions, even with the additional iteration performed for recovering the total accuracy of similarity scores. The latter result approves the recommendation made in the end of Subsect. 4.3.

For graphically illustrating the effect of introducing par-

[3] http://www.oracle.com/technology/products/berkeley-db/index.html

[4] Dreier, D. Manual of Operation: Barabasi Graph Generator v1.0. University of California Riverside, Department of Computer Science. (2002)

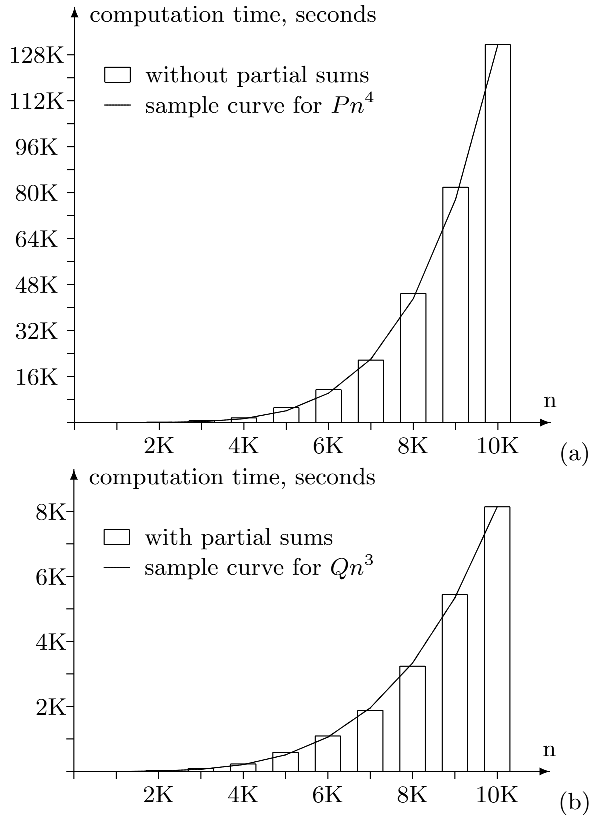

**Figure 2: SimRank computation time with and without partial sums usage w.r.t. the number of nodes $n$ in a graph.**

tial sums for SimRank computation, Fig. 2 shows computation time with respect to the number of nodes in a graph in a more detail in the form of bar charts, for SimRank computed (a) without and (b) with partial sums. For illustrating the correlation between the computation time and the number of nodes in a graph, each bar chart is approximated by a polynomial curve. Note that different scale is chosen across the vertical axis in Fig. 2 (a) and (b) for providing a more illustrative look for each curve shape.

Constants $P$ and $Q$ in Fig. 2 were calculated in accordance with minimum mean square error estimator; quartic and cubic functions respectively showed the best approximation for SimRank computation time without and with partial sums, compared to the other polynoms in the number of nodes in a graph. This experimental result fully agrees with the theoretical considerations made in Sect. 4.

## 6.2 Experiment over Wikipedia Corpus

Our practical interest for implementing the suggested optimization techniques was to compute SimRank over the complete set of articles from English Wikipedia corpus.

Wikipedia is an open content online encyclopedia project that is "created in a collaborative effort of voluntary contributors"[5]. Wikipedia is available in many languages, with the English version being the largest one, containing 2.2M articles. In addition to being a popular online encyclopedia, Wikipedia has recently obtained a big academic interest as an information corpus by itself[6], e.g. [16, 4]. However, to the best of our knowledge, nobody has yet reported the experience in computing SimRank scores for English Wikipedia corpus.

Since each Wikipedia article is generally dedicated to describing a single encyclopedic concept, we have naturally chosen an individual article to be a node in SimRank model. As Wikipedia articles are organized into categories being articles themselves, we chose the relationship "a category contains an article" to be a link from the category to the article. We will further refer to a thus built graph as the Wikipedia graph. Computing SimRank scores over the Wikipedia graph has the semantics of obtaining similarity scores for encyclopedic concept pairs. Note that the Wikipedia graph covers only a subset of Wikipedia corpus, since category links constitute a subset of links available in Wikipedia.

As we had a practical interest in computing SimRank over the Wikipedia graph from the beginning of our research, our implementation evolved throughout the optimization techniques development. Before any optimization techniques were introduced, preliminary experiments showed that SimRank computation over the Wikipedia graph would have taken months at least to complete, which was unacceptable.

After the set of optimization techniques presented in this paper has been worked out, SimRank computation over the Wikipedia graph takes approx. 17 hours to complete on a single machine, making it possible to perform the computation on a nightly basis. We use a machine with 3GHz Intel Pentium 4 processor, 4Gb RAM and 32-bit Linux OS; the cache size of 256Mb is specified for Oracle Berkeley DB. The following SimRank computation parameters are used: $C = 0.6$, $K = 5$, $\Delta = 0.05$. The chosen parameters values provide reasonable accuracy $\epsilon = 0.6^{5+1} + 0.05 < 0.1$ for computed SimRank scores. Note that due to the polynomial dependence between the decay factor $C$ and accuracy $\epsilon$, even a relatively small decrease in the decay factor results in a considerable improvement in accuracy.

Note that the Wikipedia graph provides a practical illustration for Corollary 1 presented in Subsect. 4.1: Wikipedia graph nodes not being categories have no outgoing links, and thus SimRank scores for them are computed on the last iteration only, while correctly preserving the semantics of SimRank iterative model.

SimRank similarity scores for Wikipedia concepts pairs provide a valuable practical source of information. We are using the computed scores for extending search engines functionality and for word sense disambiguation. Our further plans include using the computed scores for automatic news feeds classification.

# 7. FUTURE WORK

Although optimization techniques presented in this paper provide a considerable improvement to SimRank computational complexity, our profiling experiments revealed that a significant proportion of computation time is occupied by graph access operations $I(v)$ and $O(v)$. Although SimRank computational complexity is guaranteed to remain at most cubic in the number of nodes, graph access operations may become a performance bottleneck in the growing size of input graphs. Our future work thus involves developing fur-

[5] http://wikipedia.org/

[6] Wikipedia in academic studies. – http://en.wikipedia.org/wiki/Wikipedia:Wikipedia_in_academic_studies

ther optimization techniques for speeding up graph access operations. Our current vision to achieving scalability in the growing size of input graph is splitting the graph into several (generally, intersecting) subgraphs in such a way that

1. each subgraph could be passed to its own parallel computation instance as if the complete graph without changing the result of similarity scores computation for node pairs processed by that instance; and
2. each subgraph is small enough to fit into main memory for maximizing the speed of access operations.

Our future plans include developing a systematic procedure for splitting a general graph into subgraphs that satisfy the above listed requirements. In particular, we believe that scale-free graphs theory [8] can be exploited for developing such a procedure for scale-free graphs, which constitute an underlying model for many existing practical domains.

## 8. CONCLUSION

The paper addresses the issues missing for similarity measure SimRank, namely, accuracy estimate and optimization techniques, for facilitating SimRank wider application.

A precise accuracy estimate for SimRank iterative computation is established. The estimate reveals that SimRank computation parameters suggested in the original SimRank proposal implied a relatively low accuracy, and the choice for different parameter values is suggested. The accuracy estimate allows a-priori finding out the correct number of iterations required for achieving a desired accuracy. The number of iterations turns out to be independent of input graph characteristics, the fact to benefit scalability.

Optimization techniques are suggested and integrated into the general algorithm to provide a systematic improvement for SimRank computational complexity.

Experimental results show a 50 times speedup achieved by the optimization techniques for a graph with 10K nodes, and relative improvement in computation time further increases for larger graphs. The experience in computing SimRank scores over the English Wikipedia corpus exhibits practical viability of the approach for relatively large data corpora.

We believe that the results presented in the paper would facilitate a wider application of SimRank to computer science techniques, as this similarity measure definitely deserves.

## 9. ADDITIONAL AUTHORS

## 10. REFERENCES

[1] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[2] R. Cohen and S. Havlin. Scale-free networks are ultrasmall. *Physical Review Letter*, 90(5):058701, 2003.

[3] D. Fogaras and B. Rácz. Scaling link-based similarity search. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 641–650, New York, NY, USA, 2005. ACM.

[4] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of The Twentieth International Joint Conference for Artificial Intelligence*, pages 1606–1611, Hyderabad, India, 2007.

[5] P. Ganesan, H. Garcia-Molina, and J. Widom. Exploiting hierarchical domain structure to compute similarity. *ACM Transactions on Information Systems*, 21(1):64–93, 2003.

[6] F. Geerts, H. Mannila, and E. Terzi. Relational link-based ranking. In *VLDB'2004: Proceedings of the Thirtieth international conference on Very large data bases*, pages 552–563. VLDB Endowment, 2004.

[7] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM Press, 2002.

[8] L. Li, D. Alderson, R. Tanaka, J. C. Doyle, and W. Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications (extended version). *CoRR*, abs/cond-mat/0501169, 2005.

[9] D. Lin. An information-theoretic definition of similarity. In *Proc. 15th International Conf. on Machine Learning*, pages 296–304. Morgan Kaufmann, San Francisco, CA, 1998.

[10] Z. Lin, I. King, and M. R. Lyu. Pagesim: A novel link-based similarity measure for the world wide web. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 687–693, Washington, DC, USA, 2006. IEEE Computer Society.

[11] W. Lu, J. Janssen, E. E. Milios, and N. Japkowicz. Node similarity in networked information spaces. In D. A. Stewart and J. H. Johnson, editors, *CASCON*, page 11. IBM, 2001.

[12] A. G. Maguitman, F. Menczer, F. Erdinc, H. Roinestad, and A. Vespignani. Algorithmic computation and approximation of semantic similarity. *World Wide Web*, 9(4):431–456, 2006.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web, 1998.

[14] A. R. Schmidt, F. Waas, M. L. Kersten, M. J. Carey, I. Manolescu, and R. Busse. XMark: A Benchmark for XML Data Management. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 974–985, Hong Kong, China, 2002.

[15] H. Small. Co-citation in the scientific literature: a new measure of the relationship between two documents. *Journal of the American Society for Information Science*, 24(4):265–269, 1973.

[16] M. Strube and S. Ponzetto. WikiRelate! Computing semantic relatedness using Wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pages 1419–1424, Boston, Mass., July 2006.

[17] W. Xi, E. A. Fox, W. Fan, B. Zhang, Z. Chen, J. Yan, and D. Zhuang. Simfusion: measuring similarity using unified relationship matrix. In *SIGIR '05: Proceedings of the 28th international ACM SIGIR conference on Research and development in information retrieval*, pages 130–137, New York, NY, USA, 2005. ACM.

[18] T. Zesch and I. Gurevych. Analysis of the Wikipedia Category Graph for NLP Applications. In *Proceedings of the TextGraphs-2 Workshop (NAACL-HLT 2007)*, pages 1–8, 2007.