An update to this article is included at the end

Computers & Industrial Engineering 62 (2012) 732–739

Contents lists available at SciVerse ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Searching Steiner trees for web graph query

Wookey Lee [a,*], Woong-Kee Loh [b,1], Mye M. Sohn [c]

[a] Department of Industrial Engineering, Inha Univ., Incheon 402-751, Republic of Korea
[b] Department of Multimedia, Sungkyul University, Anyang 430-742, Republic of Korea
[c] Department of Systems Management Engineering, Sungkyunkwan University, Suwon 440-746, Republic of Korea

ARTICLE INFO

Article history:
Available online 19 December 2011

Keywords:
Search engine
Steiner tree
PageRank
tf-idf
Linear Programming

ABSTRACT

The enormous growth in information technology has revolutionized the way people can access information sources. Web search engines have played an important role to support what the user wants precisely and efficiently from the vast web database. Different from conventional search engine approaches, searching the structure of the web, where the answer comprises more than a single page connected by hyperlinks, needs to be meritoriously developed. We propose Linear Programming models in order to generate the optimal structured web objects searching for relevant web graphs. In the model, the web objects with node and edge weights that represent the ranking measures for Webpages and hyperlinks are devised to rank the relevance in terms of keyword vectors. We also developed a tree-filtering algorithm and top-*k* Steiner tree algorithm that is used to provide the search recommendations in practical applications. With real web databases, the experimental study shows that the LP approach outperforms the conventional search engines with respect to execution time and quality of results.

© 2012 Published by Elsevier Ltd.

1. Introduction

The enormous growth in information technology has revolutionized the way people can access information sources. Web search engines have played an important role to support what the user wants precisely and effectively from the massive web databases. The search engines usually return a ranked list of web objects when a user issues a search query where they utilize their own weighting measurement methods to determine the ranks (Cilibrasi & Vitányi, 2007; Gloor & Dynes, 1998; Hammami, Chahir, & Chen, 2006). The conventional semantics for processing a user query is to find a set of top-*k* web pages such that each page contains all user keywords.

It has been observed that user satisfaction over search engine results has been a debatable issue and that much of the traffic on internet is materialized from search engines. As long as the web search results are a set of individual lists, it seems appropriate to display each and every web results to display a web page one by one. However, just in case the search results are a collection of multiple interrelated web pages, then there needs a new mechanism for the linked web pages at a time. There are several reasons why the traditional search engines are not satisfied enough. First, due to the decade-old keyword based search methodology, the engines have weighed each query keyword as *a discrete term* than as a cohering set of keywords that define the user intent and requirement. Second, the contemporary search engine philosophy ignores the structural relationship that exists between the Webpages when finding answers to the user request. The problem is how to find the right answer for the *graph query* which consists of a set of ***Significant nodes*** with any of the keywords and *a set of* ***Steiner nodes*** connected by the *significant nodes* via edges (ex: hypertext links).

Why we are interested in finding the linked result? It is because the information sources are intrinsically distributed. Web pages, for example, are not independent individual documents nor well defined database tuples, but are linked to each other containing partly informed, far from just a 'page of a book.' This means they share or distribute their '*significance*' via hypertext links. In the query processing, the key issue is to find *the connection structure with the Steiner nodes* effectively and efficiently, while the *significant nodes*, can easily be detected by inverted indexes, vice versa. In this paper, we define a unit of web pages (Lee & Lim, 2007; Li, Candan, Vu, & Agrawal, 2001; Vu, Ooi, Papadias, & Tung, 2008) derived from the Steiner tree in order to focus on the user's interest in terms of each keyword that the user requests. The Steiner tree includes web pages that have a set of specific keywords calculated by the weight from which the solutions are extracted, even if a web page does not include all the keywords so that the related hypertext linked web pages are derived and displayed onto the web browser.

Hence, it is crucial that a graph ranking measure should orient the search and the corresponding result can efficiently be evaluated. Since it is inevitable for search engines that searches based on the

* Corresponding author. Tel.: +82 32 860 7371; fax: +82 32 867 1605.
E-mail addresses: trinty@inha.ac.kr (W. Lee), woong@sungkyul.edu (W.-K. Loh), myesohn@skku.edu (M.M. Sohn).
[1] Formerly at KAIST, Republic of Korea.

0360-8352/$ - see front matter © 2012 Published by Elsevier Ltd.
doi:10.1016/j.cie.2011.11.013

ranking measure, such a measure plays a substantial part in what the search engine should prioritize or discard. For example, if the frequency of keywords is highly regarded in a search engine, then the ranking measure should effectively count the term frequency and/or divided by inverse document frequency *tf-idf* (Hammami et al., 2006). If the search engine relies mainly on the number of hypertext links, then the ranking measure should count on the number of in-links such as PageRank (Hou & Zhang, 2003). If the search engine considers data graph, then the semantic relationships between them should be counted (Hou & Zhang, 2003). If the keyword population is important, then the keyword distribution model such as the Two Poisson model will be reckoned (Cilibrasi & Vitányi, 2007). If the search is focused on the number of access paths of the graph, then the weight values of the edges should be devised (Vu et al., 2008; Weigel, Panda, Riedewald, Gehrke, & Calimlim, 2008; Zhang, Lin, Zhu, Zhang, & Lin, 2010). In the graph queries, the form (the Steiner tree results) should follow the function (the graph ranking measure).

Since the Steiner tree should be composed of the most relevant web objects according to web user's search interest as well as the current location of the user in a web site that will be a root node. Also, comparison criteria are needed to determine the "best" Steiner tree for the given search domain, if there are several Steiner trees with the same root node. Finally, it is necessary to evaluate the possibility of using the Steiner tree in practical web searching applications.

More precisely, we introduce a next-generation search engine based on the keyword vector. Such an engine considers all the entered query keywords as *a set of coherent term* (which should be processed along with the hyperlink structure of the web for the correct enumeration of pages). As a result, the answer is in the form of optimal nodes, graphs, and trees instead of a list of individual Webpages. By the page limitation, we suggest the following two models: a naïve Vector Space Model-based LP (VSM_LP) and an enhanced Steiner tree-based model (Tree_LP). Note that, in terms of the problem complexity and the performance, the naïve model -namely, VSM_LP—shows the same high-quality solution as the conventional vector space model (VSM).

This paper describes as follows. In Section 2, the search domain and various ranking measurements are introduced for the effective search, where an edge values are suggested with respect to a global as well as a local measurement. In Section 3, the information retrieval linear programming models are generated, a naïve VSM_LP and that is enhanced by Tree_LP for some strong constraints. In Section 4, the corresponding Steiner tree algorithm is described. Then the experiments, related works, conclusions are described in Sections 5–7, respectively.

2. Search domain and the ranking measurements

2.1. Search domain

In this paper we restrict the problem domain to a set of reachable nodes from the root node, where the root node can be derived from a search engine and/or any location of interest to the user. The problem domain can then be represented as $G^* = (N^*, E^*)$ that is a *subgraph* of G induced by $N^* \subseteq N$ if $E^* \subseteq E$ contains each directed edge of E with both endpoints in node sets N^*. Since all other nodes can be reachable from a root node and hyperlinks information of a web site can be maintained with the Steiner tree, we can use it as a Steiner tree for the structured web search if the Steiner tree can be obtained over the defined search domain. See Fig. 1.

Example 1. Fig. 1 shows the search domain defined by $N^* = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, where all the nodes are considered for search domain, whereas the directed subgraph induced by node 4 as the root node $\{4, 6, 7, 8, 9\}$.

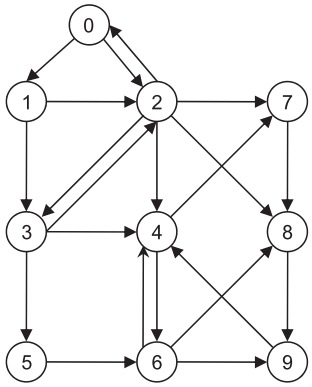

Fig. 1. Example web graph given the search domain with 0 as the root node.

2.2. Node-based node weight

Here two node weight measurements methods are described with the main concept of PageRank and *tf-idf*. Then, a new edge weight measurement method, called a *bidirectional edge weight measurement*, which uses the weights for nodes to compute edge weights will be given. *tf-idf* is one of the famous weight measurements for web objects with a user's specific query in terms of vector space model (Hammami et al., 2006; Lee, Leung, & Lee, 2011). User's search query and web objects are conceptually represented as vectors. If m distinct keywords are available to identify contents of a web object, a web object i (that is, node i) is represented as a normalized m-dimensional vector, $\boldsymbol{D}(\boldsymbol{i})$, where the terms is the weight assigned to the keyword. If the kth keyword does not exist in a web object i, then the component will be zero. To compute the m-dimensional vector $\boldsymbol{D}(\boldsymbol{i})$, an un-normalized vector $\boldsymbol{D}'(\boldsymbol{i}) = \langle \acute{\omega}_1(i), \acute{\omega}_2(i), \ldots, \acute{\omega}_m(i) \rangle^T$ is first obtained, where each $\acute{\omega}_k(i)$ is the product of a keyword frequency (*tf*) factor and an inverse document frequency (*idf*) factor. The *tf* factor is equal to the frequency of the kth keyword within the search domain. Typically, *idf* factor is computed by $log[n/(d_k+1))$, where n is the number of web objects in the search domain and d_k is that of web objects that has the kth keyword. Once un-normalized vector $\boldsymbol{D}'(\boldsymbol{i})$ is computed, the normalized vector $\boldsymbol{D}(\boldsymbol{i})$ is typically obtained by dividing each $\acute{\omega}_k(i)$ by its 2-Norm. A user's search query can also be represented as normalized vector over the same m distinct keywords space, $Q = \langle q_1, q_2, \ldots, q_m \rangle$, where $q_k = 1$ if the kth keyword t_k exists in the user's query, otherwise $q_k = 0$. With the two normalized vectors $\boldsymbol{Q}$ and $\boldsymbol{D}(\boldsymbol{i})$, we can obtain the weight of a web object i as the inner product of the two vectors.

$$R^{tf-idf}(i) = \boldsymbol{Q} \cdot \boldsymbol{D}^T(\boldsymbol{i}) \tag{1}$$

Example 2. Given a user's query $Q = \langle 1, 1 \rangle$, and the web pages in Fig. 1 are defined as the following un-normalized vectors: $D'(0)$–$D'(9)$ are $\langle 1, 1 \rangle$, $\langle 1, 0 \rangle$, $\langle 0, 1 \rangle$, $\langle 1, 2 \rangle$, $\langle 2, 3 \rangle$, $\langle 3, 4 \rangle$, $\langle 4, 1 \rangle$, $\langle 2, 3 \rangle$, $\langle 3, 4 \rangle$, and $\langle 4, 5 \rangle$, respectively. Resulting normalized vectors for the web objects from *tf-idf* method will be as $D(0)$–$D(9)$: $\langle 0.71, 0.71 \rangle$, $\langle 1.00, 0.00 \rangle$, $\langle 0.00, 1.00 \rangle$, $\langle 0.45, 0.89 \rangle$, $\langle 0.56, 0.83 \rangle$, $\langle 0.60, 0.80 \rangle$, $\langle 0.97, 0.24 \rangle$, $\langle 0.56, 0.83 \rangle$, $\langle 0.60, 0.80 \rangle$, $\langle 0.63, 0.78 \rangle$. So the *tf-idf* weight for the node "0" will be $R^{tf\text{-}idf}(0) = \langle 1, 1 \rangle \cdot \langle 0.71, 0.71 \rangle^T = 1.41$. Similarly, the weights for the nodes by *tf-idf* are $R^{tf\text{-}idf}(1) = 1.0$, $R^{tf\text{-}idf}(2) = 1.0$, $R^{tf\text{-}idf}(3) = 1.34$, $R^{tf\text{-}idf}(4) = 1.39$, $R^{tf\text{-}idf}(5) = 1.40$, $R^{tf\text{-}idf}(6) = 1.21$, $R^{tf\text{-}idf}(7) = 1.39$, $R^{tf\text{-}idf}(8) = 1.40$, and $R^{tf\text{-}idf}(9) = 1.41$, respectively.

2.3. Edge-based node weight measurement

We can define an adjacency matrix $\boldsymbol{A}$ for the directed graph with elements $\boldsymbol{A}[i, j] = 1$ if there is a directed edge from node i to

node j, otherwise $\boldsymbol{A}[i,j] = 0$, where $i, j \in N^*$. Based on the adjacency matrix $\boldsymbol{A}$, many edge-based node weight measurement methods have been developed (Hou & Zhang, 2003; Hsu & Hwang, 2004).

In the PageRank method introduced by Google (Langville & Meyer, 2006), the weight of node i, $R^{PR}(i)$, is derived with Eq. (2) as given below. In the equation, a_{ij} are the elements of Markov transition matrix obtained with the adjacency matrix $\boldsymbol{A}$ and (i,j) represents a directed edge from node i to node j. Also, $O(i)$ is defined as a set of nodes whose elements are all nodes that have any incoming edges from the node i. In addition, d is a damping factor that is used to control the speed of convergence of weights for nodes when computing the weights iteratively (Hou & Zhang, 2003; Langville & Meyer, 2006). Note that the PageRank is mostly utilized as a *global* weight with off-line mode in that a specific local keyword access is not supported.

$$R^{PR}(i) = (1-d)\left[\frac{1}{n}\right]_{n\times 1} + \sum_{\substack{(i,j)\in E \\ j\in O(i)}} w_{ij} R^{PR}(j) \tag{2}$$

Example 3. With the PageRank method, we can obtain the following node weights for the example web site shown in Fig. 1a. The PageRank values for $R^{PR}(0)$–$R^{PR}(9)$ are as follows: 0.2077, 0.238, 0.340, 0.309, 2.378, 0.281, 0.389, 1.219, 2.365, and 2.270, respectively.

2.4. Bidirectional edge weight measurement

The above two measurement approaches are orthogonal since the letter exploits only the hyperlink structure without considering the content of web objects, whereas the former uses the content explicitly not considering the hyperlink structure. We call the node weights obtained from the edge-based measurement methods as *global* weight since the node weights are computed with the whole structure of web. On the other hand, the node-based measurement method needs users' query terms: so this can be called a *local* weight. These two *global* and *local* weights can be merged into one integrated weight for search engines. Note that it is an open problem to select the best combination of the global and local weight for effective searching (Lee, Leung et al., 2011; Lee & Lim, 2007). In this research, we suggest one method for measuring edge weights using the global and local node weights. The following equation shows the edge weight measurement method, called a *bidirectional edge weight measurement*.

$$w_{(i,j)} = \alpha * R(i) + (1-\alpha) * R(j) \tag{3}$$

$$R(i) = R^{PR}(i) * R^{tf-idf}(i) \tag{4}$$

As shown in Eq. (3), the edge weight is computed as a linear combination of the weights for two nodes i and j. Here, the two nodes weights, $R(i)$ and $R(j)$, are obtained by the global and local weights for the nodes as given in (4). For example, once the converged node weights are obtained using the PageRank method or *tf-idf* method with which we can derive the edge weights to revamp the Eq. (3). The edge weight $w_{(i,j)}$ can be used as a measure for the degree of relevance between two linked web objects (that are source node and destination node) of the edge (i,j) over a given keyword vector. Therefore, it can be assumed that the magnitude of the edge weight represents how close these two nodes are. So, the *bidirectional edge weights* can consider both weights for the source and destination nodes when computing weights for directed edges. Notice that the PageRank algorithm has been proved to converge, the Eq. (3) as a linear combination of the two already converged results, i.e., $R(i)$ and $R(j)$, with fixed parameters that can trivially be shown to be converged (Langville & Meyer, 2006; Pilarski & Kameda, 1993). Due to space limitation, we do not explain more on this issue, but this approach can be a well-defined and popular application for the node and edge weight measurement from the typical information retrieval perspective (Lee, Lee, Kim, & Leung, 2009; Xue et al., 2003; Zhang et al., 2010).

Example 4. From Examples 2 and 3, we can get the integrated node weights as follows: the weight for the node "0" will be $R(0) = R^{PR}(0) * R^{tf\text{-}idf}(0) = 0.2077 * 1.40 = 0.293$. So $R(1)$–$R(9)$ as: 0.238, 0.340, 0.414, 3.306, 0.394, 0.470, 1.693, 3.311, 3.201. Thus, the bidirectional edge weight from node 0 to 1, $w_{(0,1)}$, by the above Eq. (3) will be 0.246 for $\alpha = 0.15$ that is just like the damping factor of the Google PageRank (Pandurangan, Raghavan, & Upfal, 2002; Cilibrasi & Vitányi, 2007).

3. Web Information retrieval LP model

3.1. Notations and naïve VSM_ LP model

The conventional VSM based document retrieval system selects the documents with highest node weights depending on the number of documents required for the solution set. Hence a straight forward mathematical formulation can be to compute the node weights based on the queried keywords and then arrange the node ids in decreasing order of the node weights and finally selecting the top-k nodes. But this solution is practically impossible due to the large amount of data on the web and requires repetition of this entire process even for a small amount of change in the web. However there exists a distinct problem solving technique known as Linear Programming approach. In order to justify the use of LP for web search we first present the basic model that is analogous to a naïve VSM approach. Hence the objective of the model is to select those nodes that maximize the total node weights given as in Eq. (4). The decision variables are simple Boolean node variables that will be restricted to take only two values 0 or 1 for all the nodes in the search domain given by N^*, represented as Eq. (7) and finally the restriction of node variables equal to the top-k solution required by the user is given in Eq. (6). Due to the nature of the model being similar to the VSM approach for document retrieval, we call it VSM_LP.

$$[\textbf{VSM\_LP}]$$

$$\text{maximize } z = \sum R(i) \cdot n_i \tag{5}$$

$$\text{subject to} \sum_{i\in N*} n_i = k \tag{6}$$

$$0 \leqslant n_i \leqslant 1 \quad \forall i \in N* \tag{7}$$

where n_i is a node variable having a value when the node is selected, otherwise 0.

Example 5. The example consists of 10 nodes and there would be 10 decision variables from n_0 to n_9. Let us say we want the top four nodes from the example, then the total of all the decision variables should be equal to 4 and the objective function is to select the nodes with maximal weight. The corresponding VSM_LP can be formulated as:

$$\begin{aligned}\text{maximize } z = {} & 0.293n_0 + 0.238n_1 + 0.34n_2 + 0.414n_3 \\ & + 3.306n_4 + 0.394n_5 + 0.47n_6 \\ & + 1.693n_7 + 3.311n_8 + 3.201n_9\end{aligned}$$

$$\text{subject to } n_0 + n_1 + n_2 + n_3 + n_4 + n_5 + n_6 + n_7 + n_8 + n_9 = 4$$

$$0 \leqslant n_i \leqslant 1 \quad \forall i = 0\text{–}9$$

The solution for the above VSM_LP model is obtained as $n_4 = n_7 = n_8 = n_9 = 1$ and all the other $n_i = 0$ where the optimal cost is 11.511.

The former document retrieval systems as well as the current search engine implementations share a prominent performance hazard to process the entire query execution phase from scratch even if there is a small change in any one of the single nodes. However our approach to utilize LP technique for web search has the distinct advantage of conducting a post-optimality analysis, also known as sensitivity analysis that helps to analyze the optimal solution. This helps to provide answers to various "what–if" kinds of question without repeating the entire query processing phase. In simple terms it helps to predict the change in the optimal solution for the corresponding change in the input parameters.

3.2. Sensitivity analysis

The former document retrieval systems as well as the current search engine implementations share a prominent performance hazard to process the entire query execution phase from scratch even if there is a small change in any of the single nodes. However, our novel approach to utilize LP technique for Web search has the distinct advantage of conducting a post-optimality analysis, also known as *sensitivity analysis*, which helps to analyze the optimal solution. This helps to provide answers to various "what–if" kinds of question without repeating the entire query processing phase. In simple terms, it helps to predict the change in the optimal solution for the corresponding change in the input parameters. We explain the sensitivity analysis for the example graph here and will elucidate its usage for real Web implementation later in the experimental section. The result of the post-optimality analysis for Example 3 is given in Table 1 for the selected node sets. The table provides the range of values for each of the node variables.

Another significant aspect from utilizing LP techniques over the other renowned techniques like Steiner tree method is that LP can solve the problems more efficiently and quickly. Since efficiency and performance of an algorithm is considered synonymous to its time complexity, we refer to Spielman and Teng (2001). They have proposed a new kind of analysis tool called "Smoothed analysis" through which they claim that the simplex algorithm for solving LP models always take time polynomial to the input of the problem. The simplex algorithm is one of the most popular and frequently used algorithms for solving LP models and is subsequently the algorithm used by ILOG and in this research.

Even though VSM_LP has well defined and definite advantages over other search approaches, but it fails to include the structural relationship of the nodes while processing user query. The next vital aspect is that it can only subsume those nodes in the solution set which contains all the query keywords on a single webpage. If the query keywords are distributed across more than two nodes, then the model fails to consider them as the desired solution candidate.

3.3. Tree LP model

Followings are the notations used in the Steiner tree LP model.

i, j Indices for nodes included in search domain, $i, j = \{0, 1, \ldots, n\}$. (Here, node zero is the root node.)

$w_{(i,j)}$ Edge weight for (i, j) that represents the degree of relevance, *where* $\widehat{w}_{(i,j)} = \exp(-w_{(i,j)})$.

$x_{(i,j)}$ Decision variable that equals 1 if directed edge (i, j) is selected in the tree, and 0 otherwise.

Table 1
Sensitivity of the example result.

Top 4 nodes	Lower range	Upper range
4	3.306	1.00E+20
7	0.47	0.9
8	0.47	1.00E+20
9	0.47	1.00E+20

[**Tree_LP**]

$$\text{minimize} \sum_{\substack{i,j \in N^* \, j \geqslant 1 \\ (i,j) \in A^*}} \widehat{w}_{(i,j)} x_{(i,j)} \tag{8}$$

$$\text{subject to} \sum_{\substack{i \in N^* \\ (i,0) \in A^*}} x_{(i,0)} = 0 \tag{9}$$

$$\sum_{\substack{i \in N^* \\ (i,j) \in A^*}} x_{(i,j)} = 1 \quad \text{for all } j \neq 0 \tag{10}$$

$$x_{(i,i)} = 0 \quad \text{for all } i \tag{11}$$

$$\sum_{\substack{i \in N* \\ (i,j) \in A*}} x_{(i,j)} - \sum_{\substack{j \in N* \\ (i,j) \in A*}} x_{(i,j)} = \begin{cases} +1, & \text{if } i = 0 \\ -1, & \text{if } i = t \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

$$0 \leqslant x_{(i,j)} \leqslant 1 \quad \text{for all } i, j \tag{13}$$

The model consists of an objective function and 8 constraints as above. The objective function is to minimize problem that is converted from the maximization cost denoted by the sum of edge weights where the edge weight derived by the Eq. (3). Constraints (9)–(13) are all required to satisfy conditions for feasible Steiner tree. Constraint (9) ensures that any edge toward the root node should not be included. Constraint (10)ensures that every node should have only one node as its parent in the tree and constraint (11)ensures that no self-cycle is allowed. Constraint (12)represents variables that determine paths from the root node to terminal nodes. Constraint (13)is needed to limit each directed path for the tree.

Example 6. From Example 4, we can get the final edge weight by the equation $\widehat{w}_{(i,j)} = \exp(-w_{(i,j)})$. So the weight of edge $(0, 1)$ is: $\widehat{w_{(0,1)}} = \exp(-w_{(0,1)}) = \exp(-0.246) = 0.78$. All the edges weights are represented in Fig. 2.

4. Top-*k* Steiner tree algorithm

From the optimal result by Tree_LP, the top-k is practically desirable as follows. Since the whole tree is too huge to use for each and every individual user, so that the tree algorithm finds, at first, all directed edges included in Tree_LP are sorted in descending order of edge weights (lines ⟨2⟩ and ⟨3⟩). From the

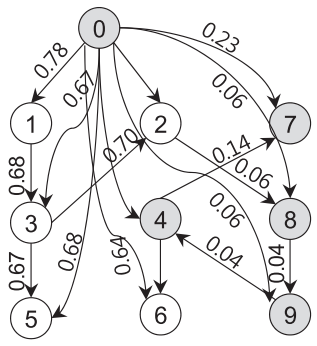

Fig. 2. Steiner tree with virtual edges in the example, where the shaded nodes are significant nodes and white node are Steiner.

sorted directed edges, the top-"k" directed edges (that means top-"k" nodes excluding the root node) are selected. Then the result is created with both the selected k directed edges and its k destination nodes using the same method (line ⟨4⟩).

In other words, if a directed edge is not selected, its destination node is also to be deleted. Instead of the deleted directed edge, a new directed edge from the source node of the deleted directed edge to each succeeding node of the deleted (destination) node is created if the directed edge from the deleted node to the succeeding node is included in the top-k directed edges.

Algorithm. top_k_tree

Input: OP= Result from [Tree_LP], k ; Output: result top-k tree
⟨1⟩ **top_k_ tree**(n, k) {
⟨2⟩ Make a list of Nodes from OP and the edges included in *EdgeList*;
⟨3⟩ Sort *EdgeList* in descending order of edge weights;
⟨4⟩ Select the first k directed edges from the sorted *EdgeList* and make the result with the selected k directed edges and $k+1$ nodes that are connected by the selected directed edges including the root node;
⟨5⟩ Tie-break if there exist multiple edges within the top-k limit.
⟨6⟩ } **// End top_k_tree(·)**

Example 7. Upon the Example 6 and Fig. 2, the Tree_LP model can generate the optimal tree result from which the algorithm *top_k_tree* can filter out the top-4 dynamic Steiner tree. It is depicted in Fig. 3.

5. Experiments

In this section, an empirical analysis for the efficacy and performance of various LP models is presented. We have conducted a comprehensive set of experiment to highlight the various aspects of the LP models and how they can improve the performance of the information retrieval systems. The next step is to validate the application of VSM_LP for document retrieval as accomplished by the textual relevancy based VSM with improved proficiency in query processing. Lastly, we estimate the usefulness of Tree_LP model to produce a ranked list of web page pairs in place of single Webpage answer to user query.

5.1. Datasets and query sets

We have used four well-known data sources from the real Web: DBLife,[2] IMDB,[3] Stanford,[4] and aclu.org.[5] The data set for DBLife consists of 10,750 nodes and 90,278 arcs, IMDB data set contains 10,000,054 ratings and 95,580 tags applied to 10,681 movies by 71,567 users of the online movie recommender service, cs.Stanford.edu consists of 537 nodes, 26,698 arcs, and aclu.org 1,455 nodes with 21,502 arcs, respectively.

The experiments have been performed on an Intel Core 2 Duo personal computer with 2.13 GHz processing power and 1 GB of RAM. The three algorithms, for computing the arc weights and the other two Steiner Graph, Information Units for comparison have been implemented using C# whereas the LP based optimization models have been programmed using the ILOG CPLEX 8.1 software with Concert technology (i.e., C++ interface).

[2] http://dblife.cs.wisc.edu/download/crawledFiles-20100330.tgz.
[3] http://www.grouplens.org/node/ml-data-10M100K.tar.gz.
[4] http://cs.stanford.edu.
[5] http://www.aclu.org.

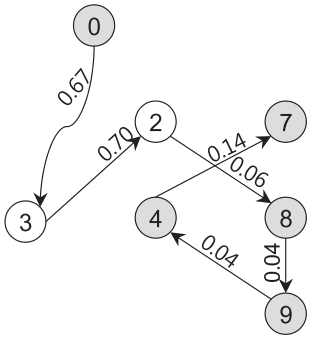

Fig. 3. Top-k dynamic Steiner tree ($k = 4$).

We selected 11 queries as listed in Table 2. The queries find a graph that one of the two keywords is in one Webpage and the other keyword on different but connected Webpages, so that among the co-existing Webpages the maximum *tf-idf* valued keyword is selected. The size of the dataset and the distribution of the keywords are varied. The number of the keyword appeared from 75 to 10,000 Webpages.

5.2. Evaluation of VSM_LP

In this section, we examine the performance of VSM_LP over Web data set so as to substantiate the use of LP models for the fundamental information retrieval system. The eminent use of *tf-idf* values in information retrieval system made it an important benchmark for comparison with VSM_LP model. The evaluation for VSM_LP is executed on DBLife and IMDB that are relatively larger than the others. We first computed the *tf-idf* scores for selected queries (Q1–Q10). The objective values for top-k results were compared with summation of *tf-idf* scores. Fig. 4 confirms our claim and clearly indicates that the results obtained by both the approaches are equivalent. The equality of the values obtained by VSM_LP and the *tf-idf* scores shows that the same set of documents are returned and that the LP model provides as good as result as the conventional document retrieval system. The other performance measure is efficiency which conforms to faster query processing time as in Fig. 5. Thus the VSM_LP represents no less performance than the VSM and is thus desirable for use in information retrieval system.

Next we enumerate the efficiency of VSM_LP in terms of query processing time. The query execution time includes LP model formation which commences with reading the *tf-idf* based scores from the input files, computing node weights, in addition to the constraints and objective function followed by model execution. The average query processing time for selected queries has been summarized in Fig. 5. We notice that the first top-10 results for any query can be achieved within 6 s which is a remarkable achievement for any search system. Even when the number of results is increased by increase in value of k, the query processing time increases by a much smaller ratio. The system follows a greedy approach to select the top-k optimal nodes by selecting the top-k maximal node weights and accordingly the nodes. We observe that VSM_LP gives an outstanding performance by achieving top 100 results in time frame of 10 s for all the selected queries with real web data.

The above experimental analysis asserts that LP models provides an efficient means to implement a text based document retrieval system with results equivalent to the most frequently implemented conventional document retrieval system based on VSM.

Table 2
Query set.

Dataset	Keyword 1	Keyword 2	Query id
DBLife	Conference	Integration	Q1
	Relational	Search	Q2
	Information	Dataspaces	Q3
	Knowledge	Integration	Q4
IMDB	Rumour	Actor	Q5
	Berkeley	Dream	Q6
Stanford	DB	IR	Q7
	GRE	Information	Q8
ACLU	Immigration	Liberty	Q9
	Terrorism	Private	Q10

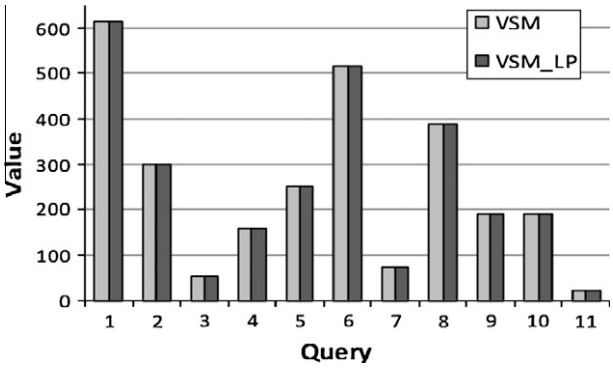

Fig. 4. Average performances for VSM and VSM_LP.

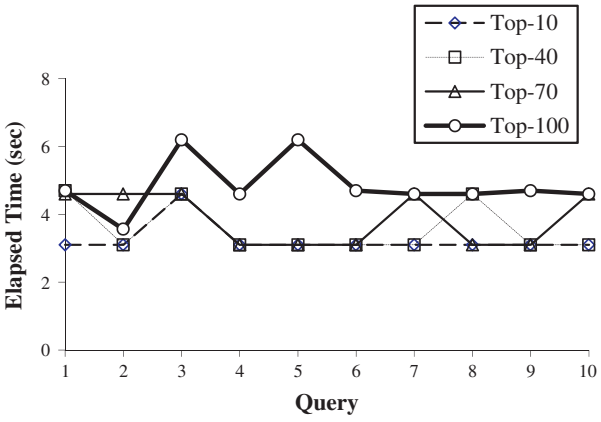

Fig. 5. Average processing time for the selected four queries by VSM_LP.

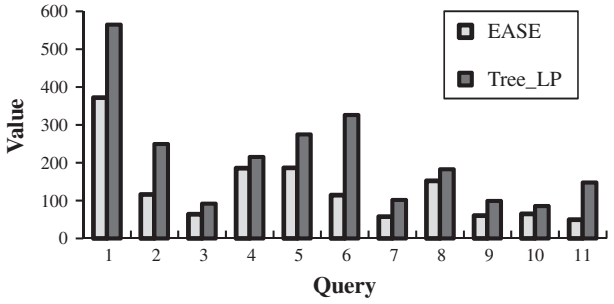

Fig. 6. Top-10 result for the query.

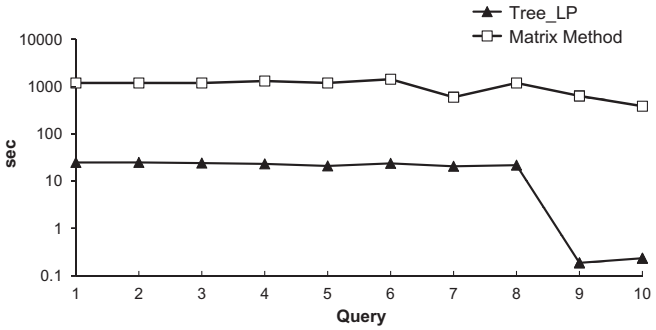

Fig. 7. Average query processing time.

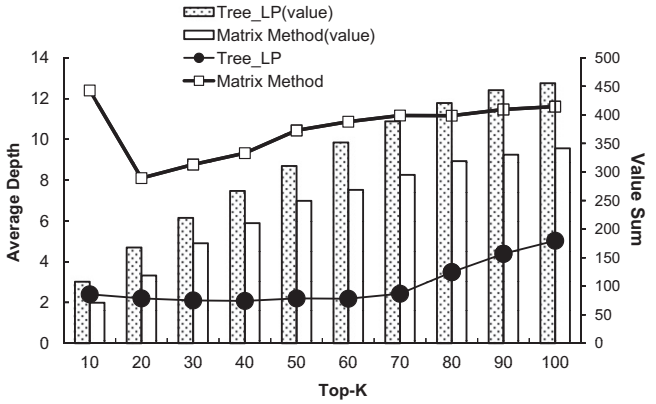

Fig. 8. Average depths and value sums with respect to top-*k* for Q10.

5.3. Evaluation of Tree_LP

We next present empirical analysis for "Tree_LP" to produce optimal pairs of nodes with complete information instead of single nodes (Webpages) with partial information. The model performance is evaluated against the conventional Matrix based method (Vu et al., 2008) that mainly focused on finding a Steiner graph by the summation of matrix transition. We here name it "Matrix Method". The first performance yardstick is query processing time and the second refers to the solution set in the optimal tree structure. We conducted experiments with web data for 11 selected queries listed in Table 2.

We can see that Tree_LP outperforms the other method for the selected queries in Figs. 6 and 7. The result in Fig. 7 indicates that Graph consumes a lot of time since they follow all possible paths from a node to the target node so that the number of paths can be increased very much. We presented here the result for Stanford data set, even though similar pattern has been found in the other entire query set due to the higher arc/node ratio for Stanford data set (4971.7%) as compared to DBLife (211.0%), IMDB (437.7%), or ACLU (284.3%).

Fig. 7 depicts the query processing time to get top-10 results for the selected query set. The Matrix Method uses the index score that consists of two components: (1) keywords sum multiplied by (2) the (sum of) inverse square of path(s). In order to get the keywords sum for the connected nodes, the Matrix Method uses the adjacent matrix multiplication which consumes time in proportion to the node size of the graph. Unlike the Matrix Method, Tree_LP does not require any additional preprocessing procedures. It consumes less than 20 seconds for all the queries. Q1–Q6 have been executed on DBLife and IMDB that has 3365 and 10 million nodes respectively, so the Matrix Method needs fairly long time. Q7, however, consumed relatively small amount of time since

the matrix extension stopped within low number, because the keyword 'Berkeley' is quite rare so that not much path extension is required and finished soon. Q9 and Q10 have been applied to relatively small node sizes, so the time reduced a lot.

In Fig. 8, the average depths and the value sums are depicted for the Q10 with respect to top-*k*. With this figure, Tree_LP finds relatively lower depths than the Matrix Method which means that the Tree_LP tries to find the optimal structure throughout the whole graph whereas the other remains closer to the root node and hence produces sub-optimal results. The value sum trajectories, however, are eye-catching because Tree_LP value sums are much higher than those of the Matrix Method. In other words, Tree_LP achieves more user satisfaction by presenting information from multiple sources against the other, since the Matrix Method provides an approximate solution with sub-optimal results.

6. Related works and discussions

As the web object measurements, the *PageRank* (O'Grady & Liang, 1998) with *tf-idf* (Hammami et al., 2006) is one of the most popular alternatives in the information retrieval community. The PageRank assumes that the graph G is a strongly coupled component that is, all other nodes can be reachable from any one node of the directed graph and the method uses a damping factor in order to guarantee the non-negativity, irreducibility and aperiodic properties of nodes weights when computing the weights using the Markov transition technique (Pilarski & Kameda, 1993). The HITS algorithm can be an alternative for a global weight, but none has been implemented as a web search engine (Kleinberg & Tardos, 2008), except the social network analysis. There have been many local measures for web objects such as *cosine* (Hammami et al., 2006), probabilistic (Cilibrasi & Vitányi, 2007), and anchor text (Eiron, McCurley, & Tomlin, 2004). These kinds of measures have similar limitations exposed on content manipulations, called content spamming, so we merged the *global* and the *local* measure to devise a *bidirectional edge measure*.

The graph theoretical approaches are useful to gain the overall size of the web and properties of the graph (Faloutsos, Faloutsos, & Faloutsos, 1999; Kleinberg & Tardos, 2008; Lee, Song, & Leung, 2011; Pandurangan et al., 2002; Ravasz & Barabasi, 2003). Among them, the hierarchical web structuring for various types of the web-as-a-graph approaches within a web site has the many advantages (Cilibrasi & Vitányi, 2007; Garofalakis, Kappos, & Mourloukos, 1999; Gloor & Dynes, 1998; Langville & Meyer, 2006). First, the tree structure can drastically reduce the search steps, roughly from $\boldsymbol{n}$ target objects to $\boldsymbol{\log n}$ (Jaillet, 1988; Lee & Lim, 2007; Murat & Paschos, 1999; Tarjan, 1982). Recall that millions of data objects can be accessed within 10 steps. Therefore the tree structure can improve searching efficiency for human users as well as for web robots. Additionally, the tree data structure is one of the simplest extensible structures, so that web objects structured as a tree form have much flexibility when it is effectively used for the web crawlers with limited resources on the fly. As mentioned in Gloor and Dynes (1998) and Hsu and Hwang (2004), knowing the hyperlinks structure through visualization during searching for web contents can improve the effectiveness of the web search due to the enhanced spatial context orientation, even though they are static (Lee et al., 2009). So in the future, a graph indexing technique needs to be exploited.

In the Industrial Engineering research society, web "applications" have long been studied such as assembly lines (Chao & Chen, 2001; Vu et al., 2008), Image processing (Bosques, Rodriguez, Rondon, & Vasquez, 1997), Web services (Murat & Paschos, 1999; Pandurangan et al., 2002), Site map (Li et al., 2001), or Mobile devices (Lee, Kang, Lim, Shin, & Kim, 2007; Lee et al., 2009), but no research has been tackled on the core information search engine issues. So it can be said that this paper is the first trial using Linear Programming on the information retrieval area, except the earlier stages of our works (Lee et al., 2009; Lee et al., 2011; Lee & Lim, 2007) and a web log mining method (Li et al., 2001; Lin, 2006) that has tried to optimize small web site map. The web site optimization method, however, is built on a wrong assumption of "linearly independence" for the LP model, since the web page traversal may be strongly "dependent" on the previous visitation. In this paper, all the weight measures in this paper are linear and are independent so that the optimization model can soundly be applied (Spielman & Teng, 2001; Tarjan, 1982).

7. Conclusion

Contributions of this study can be summarized as follows. First, we devised a new web search engine weight ranking mechanism that explicitly uses various measurements such as node based node measurement and those of edge based ones with which the VSM_LP and Tree_LP for the web information retrieval have been exploited. Second, in order to evaluate the possibility of using the new web search mechanism, the *top-k Steiner tree* algorithm is generated. Finally, we opened a promising research area in which the Linear Programming techniques are embedded onto Web information retrieval and graph based results.

This research can be extended and applied to several directions. We need to evaluate the effectiveness of the search mechanism suggested in this approach in an actual web search engine settings not only within a web site domain but also into the whole web. Comparisons between a traditional search system and Social Network systems such as Google or Facebook, and our structured web search should be made for this purpose. In addition, we can develop user applications that apply the concept of the graph search.

References

Bosques, W., Rodriguez, R., Rondon, A., & Vasquez, R. (1997). A spatial data retrieval and image processing expert system for the world wide web. *Computers and Industrial Engineering, 33*(1), 433–436.

Chao, P. Y., & Chen, T. T. (2001). Analysis of assembly through product configuration. *Computers and Industrial Engineering, 44*, 189–203.

Cilibrasi, R., & Vitányi, P. (2007). The Google similarity distance. *IEEE Transactions on Knowledge and Data Engineering, 19*(3), 370–383.

Eiron, N., McCurley, K., & Tomlin, J. (2004). Ranking the web frontier. In *Proc world wide web* (pp. 309–318).

Faloutsos, M., Faloutsos, P., & Faloutsos, C. (1999). On power–law relationships of the internet topology. In *Proc SIGCOMM* (pp. 251–262).

Garofalakis, J., Kappos, P., & Mourloukos, D. (1999). Web site optimization using page popularity. *IEEE Internet Computing, 3*(4), 22–29.

Gloor, P. A., & Dynes, S. B. (1998). Cybermap – Visually navigating the web. *Journal of Visual Languages and Computing, 9*(3), 319–336.

Hammami, M., Chahir, Y., & Chen, L. (2006). WebGuard: A web filtering engine combining textual, structural, and visual content-based analysis. *IEEE Transactions on Knowledge and Data Engineering, 18*(2), 272–284.

Hou, J., & Zhang, Y. (2003). Effective finding relevant web pages from linkage information. *IEEE Transactions on Knowledge and Data Engineering, 15*(4), 940–951.

Hsu, C. C., & Hwang, S. L. (2004). A study of interface design improvement in an engineering data management system on the world wide web. *Computers and Industrial Engineering, 47*(1), 31–43.

Jaillet, P. (1988). A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research, 36*(6), 929–936.

Kleinberg, J. M., & Tardos, E. (2008). Balanced outcomes in social exchange networks. In *Proc STOC* (pp. 295–304).

Langville, A., & Meyer, C. (2006). A reordering for the PageRank problem. *SIAM Journal on Scientific Computing, 27*(6), 2112–2120.

Lee, W., Song, J. J., & Leung, C. (2011). Categorical data skyline using classification tree. In *Proc APweb* (pp. 181–187).

Lee, W., Lee, James J.-H., Kim, Y., & Leung, C. K.-S. (2009). AnchorWoman: Top-*k* structured mobile web search engine. In *Proc CIKM* (pp. 2089–2090).

Lee, W., Kang, S., Lim, S., Shin, M., & Kim, Y. (2007). Adaptive hierarchical surrogate for searching web with mobile devices. *IEEE Transactions on Consumer Electronics, 53*(2), 796–803.

Lee, W., Leung, C. S., & Lee, J. J. (2011). Mobile web navigation in digital ecosystems using rooted directed trees. *IEEE Transactions on Industrial Electronics, 58*(6), 2154–2162.

Lee, W., & Lim, T. (2007). Architectural measurements on the world wide web as a graph. *Journal of Information Technology and Architecture, 4*(1), 61–69.

Li, W., Candan, K., Vu, Q., & Agrawal, D. (2001). Retrieving and organizing webpages by 'information unit'. In *Proc WWW* (pp. 230–244).

Lin, C. (2006). Optimal web site reorganization considering information overload and search depth. *European Journal of Operational Research, 173*(3), 839–848.

Murat, C., & Paschos, V. T. (1999). The probabilistic longest path problem. In *Proc networks* (Vol. 33, pp. 207–219).

O'Grady, P., & Liang, W.-Y. (1998). An internet-based search formalism for design with modules. *Computers and Industrial Engineering, 35*(1–2), 13–16.

Pandurangan, G., Raghavan, P., & Upfal, E. (2002). Using PageRank to characterize web structure. In *Proc COCOON* (pp. 330–339).

Pilarski, S., & Kameda, T. (1993). Simple bounds on the convergence rate of an ergodic Markov chain. *Information Processing Letters, 45*(2), 81–87.

Ravasz, E., & Barabasi, A. L. (2003). Hierarchical organization in complex networks. *Physical Review, 67*.

Spielman, M. M., & Teng, S. H. (2001). Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. In *Proc STOC* (pp. 296–305).

Tarjan, R. E. (1982). Sensitivity analysis of minimum spanning trees and shortest path trees. *Information Processing Letters, 14*(1), 30–33.

Vu, Q. H., Ooi, B. C., Papadias, D., & Tung, A. K. H. (2008). A graph method for keyword-based selection of the top-*k* databases. In *Proc SIGMOD* (pp. 915–926).

Weigel, F., Panda, B., Riedewald, M., Gehrke, J., & Calimlim, M. (2008). Large-scale collaborative analysis and extraction of web data. In *Proc VLDB* (pp. 1476–1479).

Xue, G., Zeng, H., Chen, Z., Ma, W., Zhang, H., & Lu, C. (2003). Implicit link analysis for small web search. In *Proc SIGIR* (pp. 56–63).

Zhang, Y., Lin, X., Zhu, G., Zhang, W., & Lin, Q. (2010). Efficient rank based KNN query processing over uncertain data. In *Proc ICDE* (pp. 28–39).

Update

Computers & Industrial Engineering
Volume 63, Issue 2, September 2012, Page 538

DOI: https://doi.org/10.1016/j.cie.2012.03.007

Computers & Industrial Engineering 63 (2012) 538

Contents lists available at SciVerse ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Corrigendum

Corrigendum to "Searching Steiner trees for web graph query" [Comput. Ind. Eng. 62 (2012) 732–739]

Wookey Lee [a,*], Woong-Kee Loh [b,1], Mye M. Sohn [c]

[a] Department of Industrial Engineering, Inha University, Incheon 402-751, Republic of Korea
[b] Department of Multimedia, Sungkyul University, Anyang 430-742, Republic of Korea
[c] Department of Systems Management Engineering, Sungkyunkwan University, Suwon 440-746, Republic of Korea

The authors regret that the following acknowledgements were omitted in the above article. This acknowledgement is included below:

"This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the Contract UD080042AD, Korea."

DOI of original article: http://dx.doi.org/10.1016/j.cie.2011.11.013
* Corresponding author. Tel.: +82 32 860 7371; fax: +82 32 867 1605.
E-mail addresses: trinty@inha.ac.kr (W. Lee), woong@sungkyul.edu (W.-K. Loh), myesohn@skku.edu (M.M. Sohn).
[1] Formely at KAIST, Republic of Korea.

0360-8352/$ - see front matter © All rights reserved.
http://dx.doi.org/10.1016/j.cie.2012.03.007