

Exploiting ID References for Effective Keyword Search in XML Documents

Bo Chen¹, Jiaheng Lu², and Tok Wang Ling¹

¹ School of Computing, National University of Singapore
{chenbo,lingtw}@comp.nus.edu.sg

² University of California, Irvine
jiahengl@uci.edu

Abstract. In this paper, we study novel *Tree + IDREF* data model for keyword search in XML. In this model, we propose novel *Lowest Referred Ancestor (LRA) pair*, *Extended LRA (ELRA) pair* and *ELRA group* semantics for effective and efficient keyword search. We develop efficient algorithms to compute the search results based on our semantics. Experimental study shows the superiority of our approach.

1 Introduction

Keyword search is a convenient way to query XML documents since it allows users to easily issue keyword queries without the knowledge of complex query languages and/or the structure of underlying data.

Existing approaches. Majority of the research efforts in XML keyword search focus on keyword proximity search in either *tree model* or *general digraph model*. Both approaches generally assume a smaller sub-structure of the XML document that includes all query keywords indicates a better result.

In tree model, *SLCA (Smallest Lowest Common Ancestor)* ([7,3,8,6]) is a simple and effective semantics for XML keyword proximity search. Each SLCA result of a keyword query is a smallest XML node¹ that 1) covers all keywords in its descendants and 2) has no single proper descendant to cover all query keywords. However, the SLCA semantics based on tree model does not capture ID reference information which is usually present and important in XML databases. As a result, it may return a large tree including irrelevant information. For example, Figure 1 shows an XML data for computer science department in a university. Consider keyword query “Smith Database”, which looks for whether Smith teaches some Database course. In this case, the SLCA is the overwhelming root of the whole XML database, which will frustrate the searcher.

On the other hand, XML documents can be modeled as digraphs to take into account ID reference edges. The key concept in digraph model is called *reduced subtrees* ([2,5]), which searches for minimal connected subtrees in graphs. However, the problem of finding all reduced subtrees and enumerating results by

¹ When we say an XML node in this paper, we refer to the subtree rooted at the node.

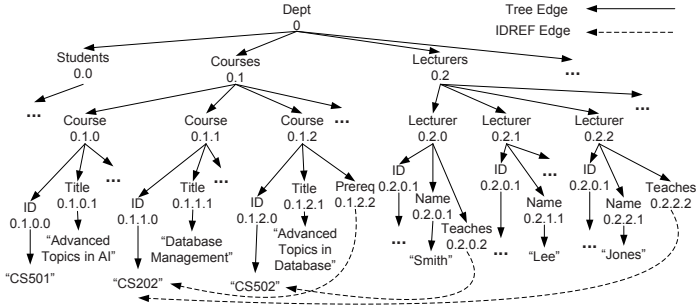


Fig. 1. Example XML document of computer science department (with Dewey IDs)

increasing sizes of reduced subtrees is NP-hard. Thus, most previous algorithms (e.g. [4]) on XML digraph model are intrinsically expensive, heuristics-based.

Our Approach. In this paper, we study a novel special graph, *Tree + IDREF* model, to capture the same ID references in digraph model which are missed in tree model; and meanwhile to achieve better efficiency than general digraph model by distinguishing reference edges from tree edges in XML.

In particular, we first propose novel *LRA pairs* (*Lowest Referred Ancestor pairs*) semantics. Informally, LRA pairs semantics returns a set of lowest ancestor pairs such that each pair in the set are connected by ID references and the pair together cover all keywords in their subtrees. For example, consider the query “Smith Database” in Figure 1 again. The result of LRA pairs semantics is the pair of nodes *Lecturer:0.2.0* and *Course:0.1.2* that are connected by ID reference and together cover all keywords. Then, we extend LRA pairs that are directly connected by ID references to node pairs that are connected via intermediate node hops by a chain of ID references; which we call *ELRA pairs* (*Extended Lowest Referred Ancestor pairs*). Finally, we further extend ELRA pairs to *ELRA groups* to define the relationships among two or more nodes which together cover all keywords and are connected with ID references.

Contributions and Organization. Our contributions are as follows:

- (1) We introduce *Tree + IDREF* data model for keyword proximity search in XML databases. In this model, we propose novel LRA pairs, ELRA pairs and ELRA groups as complements of well-known SLCA semantics (Sections 2 and 3).
- (2) We study and analyze efficient polynomial algorithms to evaluate keyword queries based on proposed semantics (Section 4).
- (3) We conduct extensive experiments with our keyword search semantics. Results prove the superiority of proposed model and search semantics (Section 5).

2 Background and Data Model

ID references in XML. In many XML databases, ID references are present and play an important role in eliminating redundancies and representing relationships between XML elements. For example, in Figure 1, references indicate

important *teach* relationships between Lecturer and Course elements. Without ID references, the relationships has to be expressed in further nested structures (e.g. each lecturer is nested and duplicated in each course she/he teaches or vice versa), potentially introducing harmful redundancies.

Data Model. We model XML as special digraphs, $Tree + IDREF$, $G = (N, E, E_{ref})$, where N is a set of nodes, E is a set of tree edges, and E_{ref} is a set of ID reference edges between two nodes. Each node $n \in N$ corresponds to an XML element, attribute or text value. Each tree edge denotes an parent-child relationship. We denote a reference edge from u to v as $(u, v) \in E_{ref}$. In this way, we distinguish the tree edges from reference edges in XML. The subgraph $T = (N, E)$ of G without ID reference edges, E_{ref} , is a tree. When we talk about parent-child (P-C) and ancestor-descendant (A-D) relationships between two nodes in N , we only consider tree edges in E of T .

3 Search Semantics

3.1 LRA Pairs Semantics

Definition 1. (reference-connection) *Two nodes u, v with no A-D relationship in an XML database have a reference-connection (or are reference-connected) if there is an ID reference between u or u 's descendant and v or v 's descendant.*

Definition 2. (LRA pairs) *In an XML database, LRA pairs semantics of a list of keywords K returns a set of unordered node pairs $\{(u_1, v_1), (u_2, v_2), \dots, (u_m, v_m)\}$ such that for any (u_i, v_i) in the set,*

- (1) u_i and v_i each covers some and together cover all keywords in K ; and
- (2) there is a reference-connection between u_i and v_i ; and
- (3) there is no proper descendant u' of u_i (or v' of v_i) such that u' forms a pair with v_i (or v' forms a pair with u_i resp.) to satisfy conditions (1) and (2).

For example, there is a reference-connection between nodes Lecturer:0.2.0 and Course:0.1.2 in Figure 1 since there is an ID reference edge between their descendants (Teaches:0.2.0.2 and ID:0.1.2.0). Therefore, Lecturer:0.2.0 and Course:0.1.2 form an LRA pair for query “Smith Advanced Database”, whose SLCA is the overwhelming root. Note reference-connected Lecturers:0.2 and Courses:0.1 do not form an LRA pair for “Smith Advanced Database” since their descendants already form a lower connected pair to cover all keywords.

3.2 ELRA Pairs Semantics

Now, we extend the direct reference-connection in LRA pairs to a chain of connections as *n-hop-connection* in *Extended LRA (ELRA) pairs* semantics.

Definition 3. (n-hop-connection) *Two nodes u, v with no A-D relationship in an XML database have an n-hop-connection (or are n-hop-connected) if there are $n - 1$ distinct intermediate nodes w_1, \dots, w_{n-1} with no A-D pairs in w_1, \dots, w_{n-1} such that $u, w_1, \dots, w_{n-1}, v$ form a chain of connected nodes by reference-connection;*

Definition 4. (ELRA pairs) In an XML database, ELRA pairs semantics of a list of keywords K returns a set of unordered node pairs $\{(u_1, v_1), (u_2, v_2), \dots, (u_m, v_m)\}$ such that for any (u_i, v_i) in the set,

- (1) u_i and v_i each covers some and together cover all keywords in K ; and
- (2) there is an n -hop-connection between u_i and v_i for an upper limit L of the number of intermediate hops; and
- (3) there is no proper descendant u' of u_i (or v' of v_i) such that u' forms a pair with v_i (or v' forms a pair with u_i resp.) to satisfy conditions (1) and (2).

ELRA pairs semantics returns a set of pairs such that two nodes in each pair are lowest n -hop-connected to together cover all keywords. The upper limit (L) of the length of n -hop-connection can be determined at system tuning phase. Then, if users are interested in more results, the system can progressively increase L .

For example, in Figure 1, Lecturer:0.2.0 and Course:0.1.1 are connected by a 2-hop-connection via node Course:0.1.2. Thus, with L set as 2 or more, this pair will form an ELRA pair for query “Smith Database Management”, which can be understood as Database Management is a prerequisite of the course that Smith teaches. We can see from this example that ELRA pairs semantics has better chance to find smaller results than LRA pairs semantics since LRA pairs are the lowest pairs with direct reference-connection while ELRA pairs are the lowest pairs with connections up to L intermediate hops including reference-connection.

3.3 ELRA Group Semantics

Finally, we extend ELRA pairs semantics to ELRA groups semantics to define relationships among two or more connected nodes that cover all keywords.

Definition 5. (ELRA groups) In an XML database, ELRA groups semantics of K keywords returns a set of node groups $\{G_1, G_2, \dots, G_m\}$ s.t. $\forall G_i (1 \leq i \leq m)$,

- (1) all nodes in G_i each covers some and together cover all K keywords; and
- (2) there is a node h_i to connect all nodes in G_i by n -hop-connection with up to L' as the number of intermediate hops; we call h_i as the hub for G_i ; and
- (3) there is no proper descendants of any node u in G_i to replace u to cover the same set of keywords as u and are n -hop-connected ($n \leq L'$) to the hub; and
- (4) there is no proper descendant d of G_i 's hub h_i such that d is the hub of another ELRA group G_d and $(G_d \cup \{h_i\}) \supseteq G_i$.

Similar to ELRA pairs, we can set the value for L' at system tuning phase for ELRA groups semantics and L' can be increased upon users' requests.

Compared to SLCA and ELRA pairs, ELRA groups can potentially find more and smaller nodes in the result. For example, in Figure 1, with L' set as two, for query “Jones Smith Database”, the node group Course:0.1.1, Course:0.1.2, Lecturer:0.2.0 and Lecturer:0.2.2 form an ELRA group result. Both node Course:0.1.1 and Course:0.1.2 can be considered as the hubs in this ELRA group result.

4 Algorithms for Proposed Search Semantics

4.1 Data Structures

The data structures that we adopt in this paper are *keyword inverted lists* and *connection table*. Keyword inverted lists are standard structures for keyword search. Each inverted list stores all the Dewey labels of a keyword.

The connection table maintains one connection-list, $List(u)$, for each node u in the XML document such that $List(u)$ contains all the lowest nodes (v) that have direct reference-connection to u in document order. From the Dewey label of v , we can easily get all v 's ancestors that are not ancestors of u so that they are also reference-connected to u . Indexes can be built on top of the connection table to facilitate efficient retrieval of the connection list for a given node.

For example, Figure 2 shows the B+ tree indexed connection table for the XML data in Figure 1. In Figure 2, we can see node 0.1.1 has reference connection to 0.1.2.2, thus we can tell that 0.1.1 is also reference-connected to 0.1.2.

4.2 Sequential-Lookup Algorithm

We present Sequential-Lookup algorithm, to compute all search results for the proposed semantics in Algorithm ???. The inputs to the algorithm are k inverted lists of keywords (I_1, \dots, I_k), the connection table CT and the upper limits for connection chain lengths of ELRA pairs (L) and ELRA groups semantics (L'). The outputs are SLCA, ELRA pair and group results. The main idea is to use three steps to compute SLCA's (line 1), ELRA pairs (line 3) and ELRA groups (line 4) by calling corresponding functions. Note we can get all LRA pairs by setting the limit L of ELRA pairs as one.

The time complexities of each step for Sequential-Lookup algorithm are:

- $O(kd|N_{min}| \log |N_{max}|)$ for SLCA (based on the analytical result of [8]),
- $O(d \sum_{i=1}^k |N_i|(|E_L| + kd|Q_L| \log |N_{max}|))$ for ELRA pairs and,
- $O(|Q_{L'}|d^2 \sum_{i=1}^k |N_i|(|E_{L'}| + kd|Q_{L'}| \log |N_{max}|))$ for ELRA groups,

where k is the number of keywords; d is the maximum depth of the XML documents; $|N_{min}|$, $|N_{max}|$ and $|N_i|$ are the sizes of shortest, longest and i^{th} inverted lists in the query respectively; E_L and Q_L are the maximum number of edges and nodes reached by depth-limited search with chain length limit L for ELRA pairs; and finally $E_{L'}$ and $Q_{L'}$ are the maximum number of edges and nodes reached by depth-limited search with chain length limit L' for ELRA groups.

4.3 Rarest-Lookup Algorithm

Since every ELRA pair (or ELRA group) must include at least one node (or its ancestor) from the shortest inverted list of query keywords, it is sufficient to only check the shortest inverted list for all results. Therefore, we further propose Rarest-Lookup algorithm to scan nodes in the shortest (rarest) inverted list and their connected nodes to compute ELRA pairs and groups. The only changes for

Algorithm 1. Sequential-Lookup

Input: Keyword lists I_1, I_2, \dots, I_k , connection table CT , the upper limits L, L' **Output:** SLCA, ELRA_P, ELRA_G

- 1 SLCA=computeSLCA(I_1, I_2, \dots, I_k); // adopt existing algorithms for SLCA
- 2 let I_{seq} be the sort-merged list of I_1, I_2, \dots, I_k ;
- 3 ELRA_P=computeELRA_P($I_{seq}, I_1, I_2, \dots, I_k, CT, L$);
- 4 ELRA_G=computeELRA_G($I_{seq}, I_1, I_2, \dots, I_k, CT, L'$);
- 5 **return and output** SLCA, ELRA_P, ELRA_G;

Function computeELRA_P ($I_{seq}, I_1, I_2, \dots, I_k, CT, L$)

- 7 initial empty ELRA_P; //mapping each u to $\forall v$ s.t. $u \& v \in$ ELRA pair
- 8 **for** (each self-or-ancestors u of each node in I_{seq} in top-down order) **do**
- 9 get I_i, \dots, I_m whose keywords u does not cover;
- 10 **if** ($u \notin$ ELRA_P **and** u does not cover all keywords) **then**
- 11 $Q =$ getConnectedList(u, CT, L);
- 12 $S_u =$ computeSLCA(Q, I_i, \dots, I_m);
- 13 remove $\forall v \in S_u$ from S_u s.t. v covers all keywords;
- 14 ELRA_P.put(u, S_u);
- 15 **for** ($\forall a$ s.t. a is ancestor of u **and** $a \in$ ELRA_P) **do**
- 16 $S_a =$ ELRA_P.get(a); $S_a = S_a - S_u$; // set difference
- 17 ELRA_P.update(a, S_a);
- 18 **return** ELRA_P;

Function computeELRA_G ($I_{seq}, I_1, I_2, \dots, I_k, CT, L'$)

- 20 initial empty ELRA_G; // mapping each u to a group G s.t. u is a hub to
// connect $\forall v \in G$ as an ELRA group
- 21 **for** (each self-or-ancestors u of each node in I_{seq} in top-down order) **do**
- 22 $G =$ getELRAGroup(u);
- 23 **if** ($G \neq$ null) **then** ELRA_G.put(u, G);
- 24 **for** ($\forall a$ s.t. a is ancestor of u **and** $a \in$ ELRA_G) **do**
- 25 **if** ($G \cup \{a\} \supseteq$ ELRA_G.get(a)) **then** ELRA_G.remove(a);
- 26 $Q =$ getConnectedList(u, CT, L');
- 27 **for** (each self-or-ancestors q of each node in Q in top-down order) **do**
- 28 $G =$ getELRAGroup(q);
- 29 **if** ($G \neq$ null) **then** ELRA_G.put(q, G);
- 30 **for** ($\forall a$ s.t. a is ancestor of u **and** $a \in$ ELRA_G) **do**
- 31 **if** ($G \cup \{a\} \supseteq$ ELRA_G.get(a)) **then** ELRA_G.remove(a);
- 32 **return** ELRA_G;

Function getELRAGroup (h)

- 34 **if** (h cover all keywords) **then** **return** null;
- 35 $Q =$ getConnectedList(h, CT, L');
- 36 initial empty set G ;
- 37 **for** (each $I_i \in I_1, I_2, \dots, I_k$) **do**
- 38 $Y =$ getSLCA(I_i, Q);
- 39 remove $\forall y \in Y$ from Y s.t. y covers all keywords;
- 40 **if** (Y is empty **and** h does not cover I_i 's keyword) **then** **return** null;
- 41 $G = G \cup Y$;
- 42 **return** G ;

Function getConnectedList (u, CT, L)

- 44 return the list of lowest nodes computed by depth-limited search from CT
that have n -hop-connection ($n \leq L$) to u in document order;
-

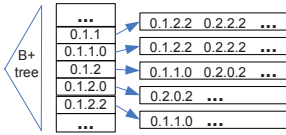


Fig. 2. Connection Table of XML tree in Figure 1

| Data | File size | Keyword inverted lists | | Connection table | |
|-------|-----------|------------------------|---------|------------------|--------|
| | | creation time | size | creation time | size |
| DBLP | 362.9MB | 321 sec | 145.7MB | 81 sec | 1.62MB |
| XMark | 113.8MB | 193 sec | 140.3MB | 234 sec | 13.7MB |

Fig. 3. Data size, index size and creation time

the Rarest-Lookup algorithm from Sequential-Lookup are at lines 2–4. Instead of getting the sort-merged list of all query keyword inverted lists, we pass the shortest (rarest) inverted list to the functions for ELRA pairs and groups.

The time complexities of each step for Rarest-Lookup algorithm are:

- $O(kd|N_{min}| \log |N_{max}|)$ for SLCA,
- $O(d|N_{min}|(|E_L| + kd|Q_L| \log |N_{max}|))$ for ELRA pairs and,
- $O(|Q_{L'}|d^2|N_{min}|(|E_{L'}| + kd|Q_{L'}| \log |N_{max}|))$ for ELRA groups,

where the variables are the same as those in Sequential-Lookup’s complexity.

5 Experimental Evaluation

5.1 Experimental Settings

Hardware and implementation. We use a normal PC with Pentium 2.6GHz CPU and 1GB memory. All codes are written in java. We set the limits of connection chain length as two for ELRA pairs and one for ELRA groups.

Datasets and indexes creation. We choose both real DBLP and synthetic XMark datasets in our experiment. Two datasets are pre-processed to create the inverted lists and connection tables. They are stored in Disk with Berkeley DB [1] B-trees. The details of file sizes and index creation of the two datasets are shown in Figure 3. Note that the connection table of DBLP is small due to the incomplete citation information of the data.

Queries and performance measures. For each dataset, we generate random queries of 2 to 5 keywords long, with 50 queries for each query size. We use these queries to compare the efficiency of 1) Sequential-lookup with Rarest-Look algorithm and 2) our algorithms with heuristic Bi-directional expansion in general digraph model.

5.2 Comparison of Search Efficiency Based on Random Queries

Sequential-lookup v.s. Rarest-lookup. In Figure 4, we show the efficiency comparisons between Sequential-lookup and Rarest-lookup in computing ELRA pairs (Seq_P and $Rarest_P$) and ELRA groups (Seq_G and $Rarest_G$). It is clear that Rarest-lookup achieves much better efficiency in both datasets. Rarest-lookup is also more scalable to queries of more keywords.

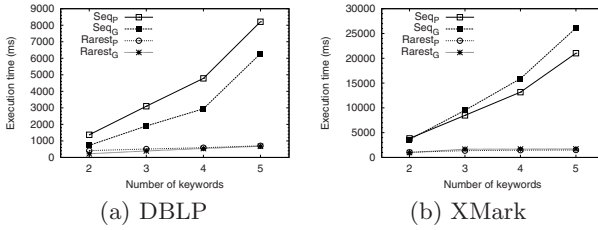


Fig. 4. Time Comparisons between Rarest-lookup and Sequential-lookup

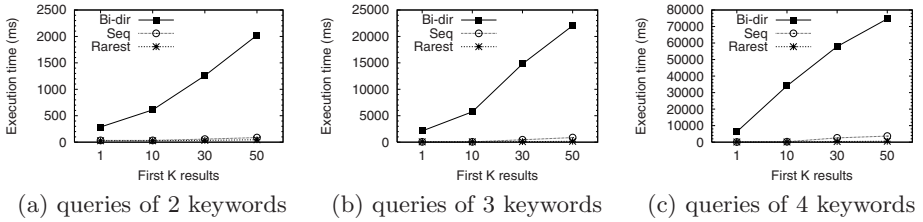


Fig. 5. Time comparisons between Bi-Directional Expansion and proposed algorithms for computing first-k results in XMark

Tree + IDREF v.s. general digraph model. Now, we compare the efficiency of our algorithms with Bi-directional expansion (Bi-dir for short) heuristics ([4]) to find first-k results. Figure 5 shows Bi-directional (Bi-dir) is significantly slower than Sequential-lookup (Seq) and Rarest-lookup (Rarest) for XMark data. For DBLP data with less amount of ID references, the result shows even more significant advantage of our approach over Bi-dir, which is omitted due to space limitation. The reasons for the inefficiency of Bi-dir are: Firstly, at each expansion, Bi-dir needs to find the best node to expand among all expandable nodes. Secondly, Bi-dir involves floating point numbers in computing and comparing the goodness of expandable nodes. Thirdly and most importantly, when Bi-dir computes or updates the goodness of a node, it has to recursively propagate the goodness to all neighbors to improve their goodness until no nodes’ goodness can be improved.

6 Conclusion

Motivated by the limitations of existing tree and general digraph model, we propose tree + IDREF model for XML keyword proximity search. In particular, we propose novel LRA pairs, ELRA pairs and groups semantics to exploit information in ID references and meanwhile leverage the efficiency benefit of tree model. Experimental evaluation shows the advantage of our methods. In the future, we would like to study relevance oriented ranking for keyword search in our model.

References

1. Berkeley DB, <http://www.sleepycat.com/>
2. Cohen, S., Kanza, Y., Kimelfeld, B., Sagiv, Y.: Interconnection semantics for keyword search in Xml. In: Proc. of CIKM Conference, pp. 389–396 (2005)
3. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: XRANK: Ranked keyword search over XML documents. In: SIGMOD, pp. 16–27 (2003)
4. Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R., Karambelkar, H.: Bidirectional expansion for keyword search on graph databases. In: Proc. of VLDB Conference, pp. 505–516 (2005)
5. Kimelfeld, B., Sagiv, Y.: Efficiently enumerating results of keyword search. In: Proc. of DBPL Conference, pp. 58–73 (2005)
6. Li, Y., Yu, C., Jagadish, H.V.: Schema-free XQuery. In: VLDB, pp. 72–83 (2004)
7. Schmidt, A., Kersten, M.L., Windhouwer, M.: Querying Xml documents made easy: Nearest concept queries. In: ICDE, pp. 321–329 (2001)
8. Xu, Y., Papakonstantinou, Y.: Efficient keyword search for smallest LCAs in XML databases. In: Proc. of SIGMOD Conference, pp. 537–538 (2005)