検索キーワードを含む最小XML部分文書抽出のための索引手法

三木　健士† 　横田　治夫††,†

† 東京工業大学 大学院 情報理工学研究科 計算工学専攻　〒 152–8552 東京都目黒区大岡山 2–12–1
†† 東京工業大学 学術国際情報センター　〒 152–8550 東京都目黒区大岡山 2–12–1
E-mail: †takeshi@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

あらまし　近年，XMLが普及する中、XML文書の中から与えられた検索キーワードを全て含む部分文書を高速に抽出する処理が注目されている．我々は、テキスト文書検索用索引に使われるスーパーインポーズドコードの手法を，階層構造を持つXML文書に対して適用する手法を提案してきた．本論文では，XML文書用のスーパーインポーズドコード手法とキーワードのB+木を組み合わせることによって，検索キーワードを全て含む最小XML部分文書を高速に取得する方法を提案する。XML木の葉ノード毎に生成したスーパーインポーズドコードを，XML木の階層構造に従ってさらにスーパーインポーズして算出したビット列を，キーワードに対して構成したB+木の葉ノードに格納することで、フォールスドロップの無い最小部分木抽出が可能となる．提案手法の有効性を示すために実験による評価を行う．

キーワード　XML、キーワード、索引、スーパーインポーズドコード、シグニチャ

Indexing Methods to Extract MCTs from an XML Document Containing Search Keywords

Takeshi MIKI† and Haruo YOKOTA††,†

† Department of Computer Science, Graduate School of Information Science and Engineering,
Tokyo Institute of Technology　Ookayama 2–12–1, Meguro-ku, Tokyo, 152–8552 Japan
†† Global Scientific Information and Computing Center, Tokyo Institute of Technology
Ookayama 2–12–1, Meguro-ku, Tokyo, 152–8550 Japan
E-mail: †takeshi@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

Abstract　The functions for extracting XML subdocuments containing all search keywords from an XML document are paid to attention recently. We have proposed methods applying superimposed code techniques to XML documents to derive the lowest common ancestor nodes for the keywords. In this paper, we enhance methods to extract the minimum connecting trees of the XML subdocuments containing all search keywords without false drops. The proposed methods combine the superimposed code techniques with a B+tree structure containing keywords. We estimate the processing costs of the proposed methods based on structural models. We also evaluate them using Xmark benchmark. The estimation and experimental results indicate that the proposed methods are effective comparing with an ordinary method.

Key words　XML , keyword ,index ,superimposed code, signature

1. はじめに

XML文書の中から必要な部分文書をXMLの構造を考慮して抽出する要求が高まっている．そのひとつとして，与えられた複数のキーワードを全て含む部分文書を効率よく抽出する方法が研究されている [1]．例えば，シェークスピアの作品をXML化したものの中から，母，王，兄弟というキーワードを台詞に含む部分文書を検索する場合，その検索結果として一人の台詞から成る部分文書や，複数人の台詞から成る部分文書などが考えられる．

XML文書はラベル付木としてモデル化されることから，上記のような検索は，キーワードを含む葉の最低共通親 (Lowest Common Ancestors: LCAs) [2, 3] を探す問題に置き換えることができる．ただし，実際には，最小最低共通親 (Smallest

LCA) [3] や，最低最小接続木（Lowest Minimum Connecting Trees: Lowest MCTs）を探す必要がある．最低最小接続木とは，Smallest LCA を根とし，検索キーワードを含んだ葉ノードを持つ木である．[1] では，XML 木の Lowest MCT を抽出するために索引の付いた XML 木に対して入れ子ループを用いる方法やスタックを用いる方法等について提案を行っている．

我々は，複数キーワードに対して MCT を効率良く求めるために，まず MCT の親である LCA を効率よく検索する手法として，スーパーインポーズドコードを利用した手法を提案してきた [4]．一般に，スーパーインポーズドコードは，文書ファイル毎にキーワードのシグニチャファイルを用意することで，文書ファイルに対するキーワード検索を高速に行うために使われる手法である [5, 6] が，これを XML 木の LCA 検索に適用する．XML 木の葉毎に割り当てた bit 列を階層構造でスーパーインポーズすることで，複数キーワードが全て含まれる LCA を判定することができる．さらに，木のノードに Dewey Order [7] のラベルを付け，キーワードに対応する葉ノードの Dewey Order のラベル間の共通部分検出で共通親を探す方法である KBDL 手法，その手法とスーパーインポーズドコードを組み合わせた手法である KBSI 手法、スーパーインポーズドコードに Bit Index で索引付けして効率を高めた方法である SIBI 手法を提案してきた．しかし，スーパーインポーズドコードを使用した手法では，フォールスドロップを考慮していなかった．また，ノードラベルを有効利用して Lowest MCT を求める方法を明らかにしていなかった．

本稿では，まず KBSI の効率の良いフォールスドロップの解決方法を提案する．次に，複数のキーワードに対して MCT を高速に得る手法として，KBDL を用いて LCA を抽出した後，ノードラベルを有効利用し Lowest MCT を抽出する KBDLM 手法と，フォールスドロップを解決した KBSI を用いて LCA を抽出した後，Lowest MCT を抽出する KBSIM 手法を提案する．さらに，既存手法である SA(Stack Algorithm) [1] と提案手法の検索コストの見積もりと実験による比較を行う．

以下では，まず 2 節で本研究の関連する研究を紹介する．3 節は本稿で使用する XML 部分木の表記法やそれに関する用語の説明をし，そして 4 節ではこれまで提案してきた LCA を求める手法を紹介する．5 節で，XML 木から XML 部分木を効率よく抽出するための索引手法の提案をし，6 節では提案手法，SA のコストの見積もりを行う．7 節では，提案手法の有効性を調べるために実験を行い，最後に 8 節では結論と今後の課題を述べる．

2. 関 連 研 究

部分文書検索としては，検索要求においてそれに類似した文書の一部分だけを検索するというパッセージ検索がある [8]．これを XML に適用し，XML 文書検索で文書構造と文書の内容に基づく順位付けをし，あらかじめ検索対象を絞る手法が提案されている [9]．XIRQL [10] も同様に順位付けをし，検索対象を絞る．この二つの研究の違いは順位付けの計算モデルの違いにある．しかし，これらの研究は，単に XML 部分文書と問い合わせキーワード間の類似度を計算する枠組を提供しているのに過ぎず，検索対象をあらかじめシステム設計者が 決めている点で汎用性に欠ける．汎用性を持たせるために，XML 部分文章の統計量を利用し検索対象である部分文章の絞り込みを行なう方法も提案されている [11]．これらの研究は，基本的に処理コスト削減のために検索対象の部分文書となりえないものを除去してから検索するフィルタリングが中心となっている．

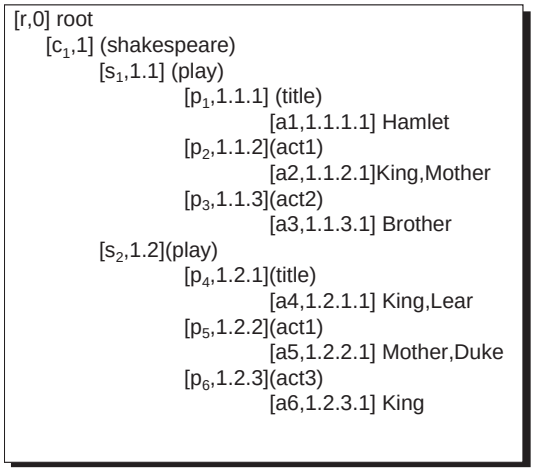

図 1　例で使用する XML 文書ラベル付

また，XML 文書の索引に関しては，XPath [12] のような XML の問合わせ言語を前提とした索引の研究が多数なされている [13–15]．しかし，MCT の検索の場合にはそのまま適用することは難しい．

一方，文献 [1] では，MCT を求めているのではなく，MCT の表記法を変えた GDMCT を用い，利用者に与える必要最小限の情報としてや Lowest GDMCT を求めている．

3. 基 本 事 項

本節では，文献 [1] で用いられている XML 部分木の表記法を紹介する．

3.1 Smallest LCA:Smallest Lowest Common Ancester

ラベル付木 T のノード $v_1,\ldots,v_m$ の Smallest LCA は，次を満たす．

（1） ノード $v_1,\ldots,v_m$ の最低共通親 (LCA) である．

（2） $v_1,\ldots,v_m$ のすべて LCA の中で，互いに親子もしくは祖先子孫関係にない．

3.2 MCT:Minimum Connecting Tree

ラベル付木 T のノード $v_1,\ldots,v_m$ の最小接続木 (MCT) は，$v_1,\ldots,v_m$ を接続する T の最小サイズの部分木 T_M である．木 T_M の根は，ノード $v_1,\ldots,v_m$ の LCA である．

例えば，Dewey Order ラベルを付した XML 文書 (図 1) において，検索キーワード (King,Mother) の MCT は (1-6) となる．

$$\begin{array}{ccccccc} c_1 & \rightarrow & s_1 & \rightarrow & p_2 & \rightarrow & a_2 \\ & \searrow & s_2 & \rightarrow & p_4 & \rightarrow & a_4 \end{array} \qquad (1)$$

$$\begin{array}{lllll} s_2 & \rightarrow & p_5 & \rightarrow & a_5 \\ & \searrow & p_4 & \rightarrow & a_4 \end{array} \quad (2)$$

$$\begin{array}{lllll} c_1 & \rightarrow & s_2 \rightarrow & p_5 & \rightarrow & a_5 \\ & \searrow & s_1 & \rightarrow & p_2 & \rightarrow & a_2 \end{array} \quad (3)$$

$$a_2 \quad (4)$$

$$\begin{array}{lllll} c_1 & \rightarrow & s_1 & \rightarrow & p_2 & \rightarrow & a_2 \\ & \searrow & s_2 & \rightarrow & p_6 & \rightarrow & a_6 \end{array} \quad (5)$$

$$\begin{array}{lllll} s_2 & \rightarrow & p_5 & \rightarrow & a_5 \\ & \searrow & p_6 & \rightarrow & a_6 \end{array} \quad (6)$$

3.3 DMCT:Distance Minimum Connecting Tree

ラベル付木 T のノード $v_1,\ldots,v_m$ を考える．ノード $v_1,\ldots,v_m$ の MCT の Distance MCT:$T_D = d(T_M)$ は，次のような最小ノードラベル，最小辺ラベルである．

（1） T_D は，ノード $v_1,\ldots,v_m$ を含む．

（2） T_D は，$v_i,v_j \in v_1,\ldots,v_m$ であるノードの組 (v_i,v_j) の LCA:$u_1,\ldots,u_k$ を含む．

（3） 辺のラベルは，ノード $n \in v_1,\ldots,v_m$ と $n' \in u_1,\ldots,u_k$ の距離とする．

ここで，例を使って説明をする．MCT(1-6) の DMCT は，(7-12) である．

$$\begin{array}{lll} c_1 & \overset{3}{\rightarrow} & a_2 \\ & \overset{3}{\searrow} & a_4 \end{array} \quad (7)$$

$$\begin{array}{lll} s_2 & \overset{2}{\rightarrow} & a_5 \\ & \overset{2}{\searrow} & a_4 \end{array} \quad (8)$$

$$\begin{array}{lll} c_1 & \overset{3}{\rightarrow} & a_5 \\ & \overset{3}{\searrow} & a_2 \end{array} \quad (9)$$

$$a_2 \quad (10)$$

$$\begin{array}{lll} c_1 & \overset{3}{\rightarrow} & a_2 \\ & \overset{3}{\searrow} & a_6 \end{array} \quad (11)$$

$$\begin{array}{lll} s_2 & \overset{2}{\rightarrow} & a_5 \\ & \overset{2}{\searrow} & a_6 \end{array} \quad (12)$$

3.4 GDMCT:Grouped Distance Minimum Connecting Tree

木 T の Grouped DMCT は，ノードのリストと数字でラベル付けされたラベル木である．DMCT:D と GDMCT:G とし，f を D のノード，辺から G のノード，辺へのマッピングとする．f は，次を満たす．

（1） n_D は D のノード，n_G は G のノード，$f(n_D) = n_G$ であるならば，n_G のラベルは，n_D の ID を含む．

（2） e_D は D の辺，e_G は G の辺，$f(e_D) = e_G$ であるならば，e_D と e_G のラベルは同じ数である．

例えば，GDMCT(13)，(14)，(15) は，DMCT(7,9,11)，(8,12)，(11) を表す．

$$\begin{array}{lll} c_1 & \overset{3}{\rightarrow} & [a_2,a_5] \\ & \overset{3}{\searrow} & [a_2,a_4,a_6] \end{array} \quad (13)$$

$$\begin{array}{lll} s_2 & \overset{2}{\rightarrow} & [a_5] \\ & \overset{2}{\searrow} & [a_4,a_6] \end{array} \quad (14)$$

$$a_2 \quad (15)$$

3.5 Lowest GDMCTs:Lowest Grouped Distance Minimum Connecting Trees

木 T の Lowest GDMCTs は，T からノード ID のリストでラベル付けされたノードと数字でラベル付けされたラベル木の集合 (n,G) である．検索キーワード $k_1,k_2,\ldots,k_m$ において，タプル (n,G) の集合は以下を満足する．

（1） n は，$k_1,k_2,\ldots,k_m$ の LCA である．

（2） G は，n を根とした GDMCT である．

（3） 求める集合のすべての LCA$\ni n,n'$ において， n が n′ の祖先ではない．

例えば，検索キーワード (King,Mother) の Lowest GDMCTs を以下のようである．

$$\begin{array}{lll} s_2 & \overset{2}{\rightarrow} & [a_5] \\ & \overset{2}{\searrow} & [a_4,a_6] \end{array} \quad (16)$$

$$a_2 \quad (17)$$

4. LCA 抽出のための索引手法

ここで本提案の前提として，[4] で提案した LCA を効率よく抽出するための索引手法を紹介する．

4.1 KBDL: Keyword B+tree with Dewey order Label

KBDL は，ノードラベル間の比較を行い単純に LCA を抽出するための索引手法である．これは，XML 木のノードに Dewey Order のラベルを付けてキーワードが出現するノードラベルをキーワード B+tree の索引に格納する．

問い合わせ時は，m 個の検索キーワード $k_1,k_2,..,k_m$ に対し，B+tree によりそれぞれのキーワードノード $KN_1,KN_2,...,KN_m$ を得る．検索キーワード k_i と $k_j : 1 \leqq i,j \leqq m$ を含む XML 木のノードの共通の親は，KN_i と KN_j の Dewey Order のラベルを比

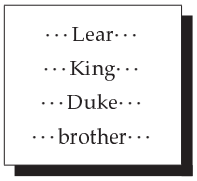

図 2　file-A

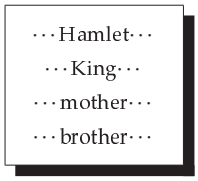

図 3　file-B

Lear	1000001
Hamlet	0100001
King	0100010
Duke	0101000
mother	0100100
brother	1100000

図 4　ワードシグニチャ

キーワード	シグニチャ
Lear	1000001
King	0100010
Duke	0101000
brother	1100000
論理和	1101011

図 5　file-A のシグニチャ

キーワード	シグニチャ
Hamlet	0100001
King	0100010
mother	0100100
brother	1100000
論理和	1100111

図 6　file-B のシグニチャ

ファイル名	シグニチャ
file-A	1101011
file-B	1100111

図 7　シグニチャファイル

較することで得ることができる．したがって，$KN_1, KN_2, ..., KN_n$ の中のキーワードの組みのラベルを順番に比較していくことで，すべてのキーワードを含む共通の親ノードを得ることができる．

4.2　スーパーインポーズドコード

SIXT と KBSI の説明の前準備として，文書検索に利用されるスーパーインポーズドコードを用いたシグニチャファイルによる索引手法の概要を述べる．

4.2.1　シグニチャファイルの構成方法

まず，シグニチャファイルを用いた索引では，全文書ファイルに出現するキーワードを抽出し，それぞれのキーワードに異る bit 列を割り当てる．これをワードシグニチャという．例として，二つのファイル（図 2，3）に出現するキーワードのワードシグニチャを示す（図 4）．

次に，対象となる文書ファイル中に含まれるキーワードのワードシグニチャの論理和を取ることでその文書ファイルに対するスーパーインポーズドコードを作成する (図 5,6)．このスーパーインポーズドコードを対応づけた表 (図 7) がシグニチャファイルである．

4.2.2　問い合わせ方法

問い合わせが与えられると，その問い合わせに含まれるキーワードの集合から問い合わせシグニチャと呼ばれる bit 列 (Q) を作成する．問い合わせシグニチャは，問い合わせ中の各キーワードのワードシグニチャの論理和を取ることで生成できる．次に，シグニチャファイル中の各ファイル (F_i) に対応するシグニチャ (S_i) との論理積を取り，その結果が問い合わせシグニチャ（Q）と等しければ，つまり以下の式を満たせば，そのファイル (F_i) は検索結果候補となる．

問い合わせキーワード	問い合わせシグニチャ	file-A	file-B
King, brother	1100010	actual drop	actual drop
King, mother	0100110	no match	actual drop
Lear, King	1100011	actual drop	false drop

図 8　問合わせとドロップの例

$$Q \wedge S_i = Q \tag{18}$$

このような候補はドロップと呼ばれる．ワードシグニチャの組合わせにより，ドロップの中には，実際には全てのキーワードが含まれないものもある．そのようなドロップをフォールスドロップと呼ぶ．これに対して，実際に全てのキーワードを含むものをアクチュアルドロップと呼ぶ．ドロップ中からフォールスドロップを見分ける処理は，フォールスドロップリゾリューションと呼ばれる．

図 7 のシグニチャファイルに対して，いくつかのキーワードの組みに対する検索結果を図 8 に示す．

4.3　SIXT: SuperImposed for XML Tree

SIXT は，XML 木のノードにスーパーインポーズドコードを割り当てて，ノードシグニチャファイルを生成した索引手法である．XML 木の葉ノードに含まれるそれぞれのキーワードに対応したワードシグニチャの論理和演算を適用することで葉ノードのスーパーインポーズドコードを生成し，XML 木の階層構造に従って論理和演算を適用することで上位ノードのスーパーインポーズドコードを生成する．これは，詳しくは次にようにして求める．まず，XML 文書のキーワード集合を K とする．葉ノード KN_i に出現するキーワード $k_j \in K (1 \leqq j \leqq |KN_i|)$ とする．葉ノードのスーパーインポーズドコード S_i は，以下の式で求める．

$$S_i = H(k_1) \wedge H(k_2) \wedge \ldots \wedge H(k_{|KN_i|}) \tag{19}$$

次に，上位ノード MN_i のスーパーインポーズドコード s_i は，MN_i の子ノード $MN_{(i-1)1}, MN_{(i-1)2}, \ldots, MN_{(i-1)m}$ のシグニチャを $s_{(i-1)1}, s_{(i-1)2}, \ldots, s_{(i-1)m}$ とし，以下の式で求める．

$$s_i = s_{(i-1)1} \wedge s_{(i-1)2} \wedge \ldots \wedge s_{(i-1)m} \tag{20}$$

検索の際，文書ファイルの時と同様にまず問い合わせシグニチャ (Q) を求め，そのシグニチャとノードシグニチャファイルのノードシグニチャ (NS_i) が以下の式を満たすか調べる．

$$Q \wedge NS_i = Q \tag{21}$$

その式が成り立つノードはその問い合わせキーワードを含むノードを子ノードとして含む．つまり，すべての検索キーワードを含む部分木の親を得ることができる．ただし，フォールスドロップリゾリューションを行っていないので，求めた LCA は，フォールスドロップを含む可能性がある．

4.4 KBSI: Keyword B+tree with SuperImposedcode

KBSIは，スーパーインポーズドコードとキーワードに対するB+木を組み合わせた索引手法である．これは，XML木のノードにスーパーインポーズドコードを付す．さらに，XMLの各ノードにDewey Orderのラベルをつけ，キーワードに対するB+木に上記のスーパーインポーズドコードと格納することで，スキャンのコストを減らす．ただし，フォールスドロップリゾリューションを行っていないので，求めたLCAは，フォールスドロップを含む可能性がある．

4.5 課　　題

上記の手法の課題は，まず，すべての手法に関してあくまでLCAを取得する手法であり，Lowest GDMCTを求めていない．次に，SIXT，KBSI手法で得られるLCAは，フォールスドロップリゾリューションを行っていないのでフォールスドロップを含む可能性がある．これは，検索キーワード $k_1,\ldots,k_m$ を含む $L \in lca_1,\ldots,lca_n$ において，k_t を含まず $k_{i1},\ldots,k_{ij}$ を含むような $lca_i \in L$ が存在する場合，そのノード lca_i において次の式を満たす場合にフォールスドロップが発生する．

$$k_t \wedge (k_{i1} \wedge \ldots \wedge k_{ij}) = k_i \qquad (22)$$

つまり，含まないキーワードのワードシグニチャが他のキーワードのワードシグニチャの論理和によって算出されたシグニチャに含まれてしまうことが原因である．

5. 提 案 手 法

本稿では，まずKBSIの課題であるフォールスドロップを解決方法を説明をする．続いて，KBDL及びフォールスドロップを解決したKBSIを用いて複数のキーワードに対してXML木のLowest GDMCTsを効率よく検索する手法として，KBDLM，KBSIM手法を提案する．

5.1 フォールスドロップを解決したKBSI

小節4.4で述べたKBSIで求めたLCAは，フォールスドロップを含む可能性がある．そこで，以下ではフォールスドロップリゾリューションの説明をする．

KBSIは，任意のキーワード k_i のキーワードノード KN_i にノードシグニチャKS_iとノードラベル NL_i の組が格納されていることから，k_i を含んでいることを保障されている．したがって，ノードシグニチャKS_iをスキャンする際，k_i 以外のキーワードのフォールスドロップが発生する可能性はあるが，キーワード k_i に対してのフォールスドロップは起こらない．

この性質を利用して次のような方法で，フォールスドロップリゾリューションを行ったLCA集合 N を得る．検索の際，検索キーワード $k_1,\ldots,k_m$ ごとのキーワードノードに格納されているノードシグニチャ集合 $NS_1,\ldots,NS_m$ を抽出する．検索キーワードから生成した問い合わせシグニチャを Q とし，検索キーワード $k_i \in k_1,\ldots,k_m$ で得られたノードシグニチャ集合 NS_i の各要素のシグニチャ s_{ij} において，以下の式を満たす場合に対応するノードラベル l_{ij} を L_i に加える．

$$Q \wedge s_{ij} = Q (0 \leqq j \leqq |NS_i|) \qquad (23)$$

```
getLCAs(k1,…,km){
Q=H(k1)∧…H(km)
For i=0,…,m do
        NSi=getNodeSigniture(ki)
        //キーワードB+木からkiのノードシグニチャ集合を得る
        Li=getResultSI(Q,NSi)
return getLCA_FalseDropLess(L1,…,Lm)
}

getResultSI(Q,NS){
L=null
//Lは空リスト
For i=0,…,|NS| do
        if(Q∧NS(i)=Q){
        L.add(NS(i))
        }
return L
}

getLCA_FolseDropLess (L1,…,Lm){
N.addAll(L1)
For  i=1,…,|L1|  do{
        For  t=2,…,m  do{
                For s=1,…,|Lt|do{
                if(L1(i)==Lt(s)){
                    break
                }
                if(t==|Li|){
                    N.remove (L1(i))
return N
}
```

図9　フォールスドロップのないLCAの取得

i を1から m まで上記の処理を繰り返し，ノードラベル集合 $L_1,\ldots,L_m$ を求める．$L_1,\ldots,L_m$ に入るノードラベルを持つノードはそれぞれ必ず $k_1,\ldots,k_m$ のキーワードを含む．それぞれのノードラベル集合に共通して現れるラベルを抽出した集合が N である．

フォールスドロップリゾリューションのアルゴリズムを，図9に示す．

5.2 Lowest GDMCTを求める手法

ここでは，Lowest GDMCTを効率よく求める手法として，KBDLMとKBSIMの提案を行う．

それぞれのLCAの抽出方法は，KBDLとフォールスドロップを解決したKBSIを利用する．Lowest GDMCTの根は，抽出したLCAのノードラベル集合 $L \ni l_1,\ldots,l_n$ において，ノードラベル比較により求めたSmallest LCAである．続いて，Lowest GDMCTの葉ノードは，検索キーワード $k_1,\ldots,k_m$ に対するキーワードノードラベル集合 $KL_1,\ldots,KL_m$ の各要素ノードラベルと根のノードラベル比較を行うことで求めるキーワードノードである．また，辺ラベルは，親と葉のノードラベル長比較により求める．

求めたLCAからLowest GDMCTs(n, G)の抽出するアルゴリズムを，図10に示す．

6. コスト見積り

ここでは，検索コスト比較として，既存手法SAと提案手法KBDLM，KBSIMの最悪の場合を見積り，比較を行う．

見積り使うXML文書のXML木を図11に表す．このときの見積りにおけるパラメタとして，以下を考える．

- m:問い合わせキーワード数
- K:全キーワード種類数

```
getLowetGDMCT(l1,…,ln, KL1,…,KLm ){
L:L はリスト
G:GをGDMCTのリスト
L=getMinimumLCA (l1,…,ln, KL1,…,KLm )
For i=0,…,|L|
      G=getG(l1,…,ln, KL1,…,KLm, L(i))
Return (L,G):Lowest GDMCT
}
getMimimumLCA( l1,…,ln, KL1,…,KLm ){
L=null //L はリスト
For i=0,…,n do
      For t=0,…,n do
        If(i!=t){
        If(lt.substring(li)){
        //ltのsubstringがliならばliはltの親である
        break
        }
        If(t==n){
        L.add(li)
        }
        }
return L //Minimum LCA
}

getG( KL1,…,KL m ,n){
G:GをGDMCTのリスト
For i=0,…,m do
      For t=0,…,|KLi| do
        If(KLi(t).substring(n)){
        G←KLi(t) : Gのi番目の葉ノードリストにKLi(t)を追加
        }
returtn G
}
```

図 10　Lowest GDMCT を得るアルゴリズム

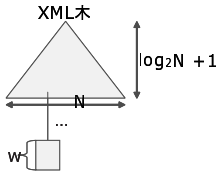

図 11　見積りにおける XML 文書木

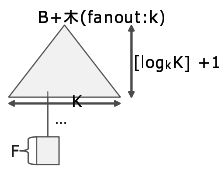

図 12　KBDLM に使用する構造

- k:B+木のファンアウト（ノードのエントリ）数
- N:XML 木の全葉ノード数
- w: 1 ノード当たりの平均キーワード種類数
- F: 1 キーワード当たりの平均ノード数
- b: 1 bit 比較コスト
- B: 1 String 比較コスト
- r:ラベルの平均 String 数
- l:シグニチャのビット長
- n:検索キーワードの平均 String 長
- d:Dewey Oeder の平均ラベル長

6.1　KBDLM

KBDLM では，LCA を求めるコストは KBDL と同じなので以下の式になる．詳細は，[4] を参照されたい．

$$m \times k \times ([log_k K] + 1) \times n \times B + F^{m-1} \times d \times B \qquad (24)$$

次は，Lowest GDMCT を求めるコストを求める．最悪の場合，Lowest GDMCT の親の数は，XML 木の最下層のノード数であるので N である．また，すべての検索キーワードのノードラベルを比較するので次式になる．

表 1　実験環境

CPU	AMD Opteron2.2G
メモリ	6G
OS	Linux version 2.6.9
database	PostgreSQL 8.1.3
DISC	DRAILD
JVM	1.5.0_07

表 2　キーワードの出現数

出現数	キーワード数
小	1-10
中	11-200
大	200-

$$N \times m \times F \qquad (25)$$

KBDLM の検索コストは，次のようになる．

$$m \times k \times ([log_k K] + 1) \times n \times B + F^{m-1} \times d \times B + N \times m \times F \qquad (26)$$

F を N と K を用いて表すと，$F = \frac{wN}{K}$ となる．これを上式に代入した上で N に関するオーダーで考えると，第 2 項，第 3 項を考慮し，

$$O((\frac{wN}{K})^{m-1} + (\frac{wN^2}{K})) = O(N^{m-1} + N^2) \qquad (27)$$

となる．$m \leqq 3$ のとき $O(N^2)$，また $m \geqq 4$ のとき $O(N^{m-1})$ ある．

6.2　KBSIM

KBSIM の LCA を求めるコストは，次の式になる．詳細は，[4] を参照されたい．

$$m \times k \times ([log_k K] + 1) \times n \times B + b \times l \times (F-1) \times m \qquad (28)$$

しかし，この LCA には，フォールスドロップを含む可能性があるのでフォールスドロップリゾリューションのコストを求める．最悪の場合，LCA 数は，N である．その中でフォールスドロップを見つけるコストは，

$$(m-1)N^2 \qquad (29)$$

である．次は，Lowest GDMCT 木を求めるコストを求める．すべてのキーワードのノードラベルを比較するので以下の式になる．

$$N \times m \times F \qquad (30)$$

以上より，検索コストは，次のようになる．

$$m \times k \times ([log_k K] + 1) \times n \times B + (m-1)N^2 + b \times l \times (F-1) \times m + N \times m \times F \qquad (31)$$

N に関するオーダーは，第 2,3,4 項となり，$F = \frac{wN}{K}$ を代入すると，$O(N^2)$ となる．

6.3　SA

[1] より，コストは次のようである．

$$O(log_2 N * F^{2m}) = O(logN * N^{2m}) \qquad (32)$$

以上より，$m > 2$ の場合，KBSIM の検索コストが一番小さいということがわかる．

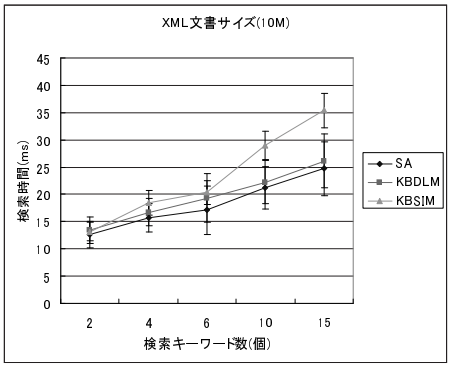

図 13　出現数：小のキーワード数を変えた比較

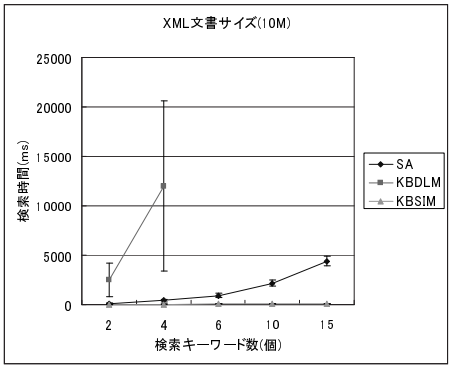

図 15　出現数：大のキーワード数を変えた比較

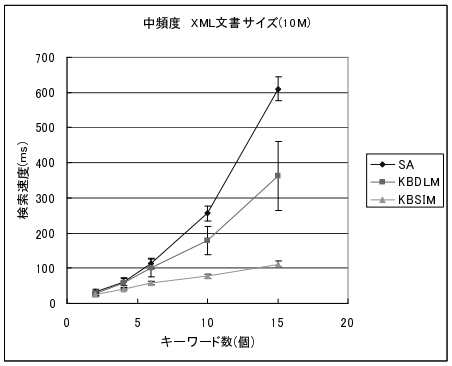

図 14　出現数：中のキーワード数を変えた比較

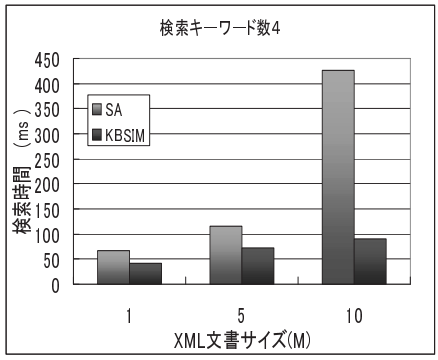

図 16　XML 文書サイズを変えた比較

7. 評価実験

本稿における評価実験の目的は，KBSIM の有効性を調べるために KBDLM と SA を様々なパラメタに基づき評価をする．実験に使用した計算機および環境を表 1 に示す．使用する XML 文書は，Xmark の XML 文書生成器 xmlgn [16] を用い規模変更因子を 0.01,0.05,0.1 として作成する．それぞれのサイズは，約 1M(1,161,652 バイト), 約 5M(5,873,341 バイト), 約 10M(11,875,066 バイト) である．予め，SA に用いる索引と KBDLM,KBSIM 手法を用いた索引を作成する．また，Lowest GDMCTs 抽出する手法のパフォーマンスは，それぞれのアルゴリズムより共通して検索キーワード数と XML 文書サイズに影響されることが予想される．したがって，パラメタを XML 文書サイズ，検索キーワード数とし，さらに [1] と詳しく比較するため検索キーワードの出現数を表 2 のように分けて実験を行う．

評価方法は，検索キーワードを与えてから Lowest GDMCTs を抽出するまでの検索時間とする．検索時間は，各出現数のキーワード群からランダムに検索キーワードを選択し，100 回の平均とする．さらに，95 %の信頼区間を使用しグラフを記した．検索キーワード数および出現数を変えた実験結果は，図 13,14,15 に示す．また，XML 文書サイズを変えた実験結果は，図 16 に示す．

7.1　フォールスドロップ

まず最初に，KBSIM のフォールスドロップリゾリューションが正常であることを確かめる．確認方法は，KBSIM の検索結果とフォールスドロップのない KBDLM，SA の検索結果との比較である．その結果，KBSIM の全ての検索結果にはフォールスドロップを含まないことが確認できた．

7.2　検索キーワード数による影響を調べる実験

検索キーワード数による影響を調べるために，XML 文書サイズを 10M に固定し検索キーワード数を変化させて検索時間を計った．併せて，検索キーワードは，各出現数において実験を行った．

まず，出現数が小である場合の実験結果を図 13 に示す．すべての手法で見られる傾向は，検索時間が検索キーワード数の増加に伴い増えている．しかし，KBSIM は他 2 手法に比べ多少遅い．次に，出現数が中である場合の実験結果を図 14 に示す．検索速度の傾向は，検索キーワード数の増加するに伴い増えている．　しかし，検索キーワード数に伴う KBSIM の検索時間の増加率は，他の手法と比べ非常に小さいことがわかる．続いて，出現数が大である場合の実験結果を図 15 に示す．ただし，KBDLM の検索時間は，他 2 手法より非常に大きいので検索キーワード数を 4 までとしている．全体の傾向としては，同様であるが，出現数が中の場合よりも顕著に特徴がはっきり現れている．

7.3　XML 文書サイズによる影響を調べる実験

次に，XML 文書サイズによる影響を調べるために，検索キーワード数を 4 に固定し XML 文書サイズを変えて実験をした (図 16)．また，KBDLM では，他 2 手法より明らかに検索時間が大きかったため図に載せていない．

図 16 から明らかなように XML 文書サイズの増加に伴い，

KBSIM は SA に比べ検索時間の増加率が小さいことがわかる．

7.4 考　　察

実験結果により検索時間は，検索キーワード数，出現数，XML 文書サイズに影響することがわかる．

まず検索キーワード数の影響について考察をする．特に出現数が中及び大の場合，検索キーワード数及び出現数の増加に伴い KBDLM と SA は大きく検索時間に影響を受けるのに対し，KBSIM は他 2 手法に比べほぼ受けていないことがわかる．これは，処理コストの見積もりで示したように KBSIM のコストのオーダは N^2 であり，検索キーワード数 m に影響しないが，SA と KBDLM は N の m 乗であるので，検索キーワード数 m に影響するためだと考えられる．なお，SA と KBDLM の検索速度を比較した場合，検索キーワード数が多いと SA のほうが KBDLM に対し早くなるが，検索キーワード数が少ない場合には，KBDLM のほうが早い．一方，出現数が小の場合の検索時間は，KBSIM が他 2 手法と比べて多少大きい．しかし，すべての手法において Lowest GDMCT を求めるためのコストが小さいため検索時間は，出現数が中及び大の場合の検索時間に比べて非常に小さいので，KBSIM の重大な欠点とは考えにくい．また，XML 文書サイズの影響に関して，KBSIM は，XML 文書サイズの増加において SA に比べ検索時間の増加率が小さいことがわかる．

以上より，KBSIM は，SA と KBDLM と比べて XML 文書サイズ，検索キーワード数，その出現数の増加に伴い効果的になると言える．

8. おわりに

本稿では，フォールスドロップを解決した KBSI と複数のキーワードに対して XML 部分木 (Lowest GDMCT) を高速に得る手法として，KBDL の手法で求めた LCA からノードラベルを有効利用し Lowest GDMCTs を求める手法である KBDLM，フォールスドロップを解決した KBSI を利用し得られた LCA からノードラベルを有効利用し Lowest GDMCT を求める手法である KBSIM を提案した．

提案手法と SA [1] に対して，いくつかのパラメタを設定し Lowest GDMCT を抽出するコストの見積もりを行った結果，検索キーワード数が 3 以上において KBSIM のコストが最小であるとわかった．提案手法の有効性を示すために，パラメタを XML 文書サイズ，検索キーワード数，およびその出現数とし，実験を行った．その結果，XML 文書サイズ，検索キーワード数，及びその出現数が増加すればするほど KBSIM 手法は，Lowest GDMCTs を効率的に抽出できることが示せた．

今後の課題として，今回は XML 部分文書を Lowest GDMCTs としたが，提案手法が様々な XML 部分文書にも有効であることを確認するため，Lowest GDMCTs 以外の XML 部分文書を求める必要がある．例えば，All GDMCTs [1] などがある．ここでは考えなかった XML 文書の更新へのコストおよび対処などが課題として挙げられる．

謝　　辞

また本研究の一部は，文部科学省科学研究費補助金特定領域研究 (18049026)，独立行政法人科学技術振興機構戦略的創造研究推進事業 CREST および東京工業大学 21 世紀 COE プログラム「大規模知識資源の体系化と活用基盤構築」の助成により行なわれた．

文　　献

[1] Vagelis Hristidis and Nick Koudas. Keyword Proximity Search in XML Trees. *IEEE Transaction on knowledge and data engineering*, No. 4, April 2006.

[2] Yunyao Li, Cong Yu, and H. V. Jagadish. Schema-free xquery. In *VLDB*, pp. 72–83, 2004.

[3] Yu Xu and Yannis Papakonstantinou. Efficient keyword search for smallest lcas in xml databases. In *SIGMOD Conference*, pp. 537–538, 2005.

[4] 三木健士, 横田治夫. スーパーインポーズドコーディングを用いた XML 文書キーワード索引. 情処学会研究会報告 DBS-140-25, 情報処理学会.

[5] C. Faloutsos and S. Christodoulakis. Descriptionand performance analysis of signature file methods for office filing. *ACM Transactions on Office Information Systems*, No. 3, pp. 237–257, July 1987.

[6] C. Faloutsos and S. Christodoulakis. Signature Files: An Access Method for Documents and Its Analytical Performance Evaluation. *ACM Trans. Inf. Syst.*, No. 4, October 1984.

[7] Igor Tatarinov, Stratis D. Viglas, Kevin Beyer, Jayavel Shanmigasundaram, Eugene Shekita, and Chun Zhang. Storing and Querying Ordered XML Using a Relational Database System. In *Proc. ACM SIGMOD 2002*, pp. 204–215, 2002.

[8] G. Salton, J. Allan, and C. Buckley. Approaches to Passage Retrieval in Full Text Information System. In *Proc. of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 49–58, June/July 1993.

[9] Y. Hayashi, J.Tomita, and G. Kikui. Searching text-rich XML documents with relevance ranking. In *Proc. of ACM SIGIR 2000 Workshop on XML and Information Retrieval*, pp. 204–215, 2000.

[10] N.Fuhr and K. FroBjohann. XIRQL: A query language for information retrieval in XML document. In *Proc. of 24th Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 172–180, 2001.

[11] 波多野賢治, 絹谷弘子, 吉川正俊, 植村俊亮. XML 文書検索システムにおける文書内容の統計亮を利用した検索対象部分文章の決定. 電子情報通信学会論文誌, No. 3, pp. 422–431, March 2006.

[12] J. Clark and S.DeRose. XML path language (Xpath) version1.0, 1999.

[13] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *Proc. of VLDB*, pp. 436–445, August 1997.

[14] Praveen R. Raw and Bongki Moon. PRIX: Indexing and querying XML using prufer sequences. In *Proc. of ICDE*, pp. 288–300, March 2004.

[15] Toshiyuki SHIMIZU and Masatoshi YOSHIKAWA. FULL-Text and Structual Indexing of XML Documents on B+-Tree. *IEICE TRANS. INF. & SYST.*, No. 1, January 2006.

[16] The xml benchmark project. http://www.xml-benchmark.org ,2006.