Webログの LCS 解析におけるスケーラビリティ向上手法の評価

Evaluation of the Scalability Improvement for Web Log Mining with LCS

戸田 誠二♡ 横田 治夫♢

Seiji TODA Haruo YOKOTA

近年，Web サイトによる情報発信は重要な位置を占めている．このため，Web サイトへのアクセス状況を的確に把握することが管理者に求められる．そこで，本論文では特に LCS (Longest Common Subsequences) を用いたアクセスログ解析を行うことで，頻出アクセスシーケンスを発見する手法について検討する．我々は以前の研究においてこの手法を提案しているが，計算量の増加等により，スケーラビリティに問題があった．本論文ではハッシュを用いたフィルタリング，同一シーケンスの統合を行うことで，アクセス状況の把握が特に求められる大規模サイトの解析にもこの手法を適用できるようその性能改善を図る．実際に本手法をアクセスログ解析に適用し，その性能に与える影響を考察する．

Nowadays, information distribution via websites is one of the most important issues. Therefore, website administrators are required to understand access trends of the websites properly. We investigate a method proposed in our previous work for mining access logs with LCS (Longest Common Subsequences) to extract frequent access sequences. However, the method still has a problem of the scalability. The execution time increases for large websites. In this paper, we propose two approaches of filtering sequences with a hash function and eliminating duplication of the same sequences to improve the performance for analyzing large websites. We evaluate the efficiency of the approaches using access logs of a real website and artificial data.

1. はじめに

現在，Web サイトは広報・広告・マーケティングの手段として極めて重要な位置を占めている．Web サイトを開設し企業情報・商品情報を発信することはもはや企業の発展にとって必要不可欠な要件である．現在は情報が氾濫しており，ユーザは無数にある Web サイトから自分の望む情報を効率良く獲得しようとする．もし，サイトの構成が悪くユーザに必要な情報が不透明であればユーザは即座に情報探索を諦め，次のサイトへ移るだろう．

このような状況の中で，Web サイトへのアクセス状況を的確に把握し発信したい情報がユーザに確実に届いているかどうかを見極めることは Web サイト管理者の最も重要な仕事の一つである．その手法の一つとして Web サイトのアクセスログ解析がある．管理者はアクセスログの解析結果から取り出す情報を意思決定材料として，ユーザのニーズにあったサイト構築を行う必要がある．

Web サイト管理者は，IP アドレス，URL，アクセス日時や，Cookie 情報等をアクセスログとして記録することができる．一般的に広く行われているアクセスログ解析のアプローチにはページ別アクセス頻度，時間別，期間別アクセス頻度，訪問者の使用している OS，ブラウザの種類の集計，Web サイトの参照元の集計などが存在する．この解析手法によってサイト管理者はページ別，時間帯別のアクセス状況，自サイトへの参照元などを知ることができる．これらの情報も管理者にとって重要な要素の一つであるが，それだけでは訪問者がどのようにサイトを巡回する傾向にあるのか判断できない．すなわちサイトの構成上の問題点については分からないのである．

我々は以前の研究でアクセスパターンを解析し，ユーザの高頻度な巡回パターンを抽出するため，LCS (Longest Common Subsequences) を用いたアクセスログ解析を行った [1]．これは目的のサイト内での全セッションの URL の推移をシーケンスとして抽出し，各シーケンスについて総当たりで LCS を求め，頻出アクセスパターンを発見するものである．また，集計された LCS をサイトの再構成に活用する手法についても考察を行った．

しかし，以前の研究では単純に各シーケンスに関して総当たりで LCS を計算していたため，その計算量がアクセスログのセッション数の二乗オーダで増加するという問題があった．一般的に，アクセスログ解析は特に商用の大規模サイトでの需要が高く，高いスケーラビリティが要求される．このため本論文では，LCS を抽出し頻出シーケンスを発見する過程において，計算量を小さくすることで性能の改善を図る．具体的には 2 つのシーケンスから LCS を計算する際，それぞれのシーケンスをハッシュによってフィルタリングすることと，完全に同一のシーケンスを統合することでこれを実現する．さらに上記手法を利用して，我々の研究室で公開している Web サイトのアクセスログを解析することで本手法の有効性を検証する．

2. 関連研究

近年，データマイニング手法等を利用したアクセスログ解析の研究は，盛んに行われてきている．

特に，Web サイトの再構成を目的とした研究としては Srikant らによるユーザのバックトラックポイントの減少を目的とした解析手法が挙げられる [2]．しかし，ユーザ操作はランダム性に富み，またネットワークの状態はユーザごとに異なるため，アクセスログのみからバックトラックポイントと目的のページを区別することは困難である．

また，アクセス解析を目的としたアクセス状況の可視化の研究も盛んである．その中でもユーザのクリック操作の可視化を目的としたものが多く，Web サイトのアクセス状況を示し，解析，改善を支援する様々なツールが提案されている [3, 4]．構造解析のためには可視化が重要であるが，それらのアプローチは個々のページのアクセス頻度や推移に主眼を置いており，ユーザのアクセスシーケンスの傾向を見いだすことはできない．

我々は以前の研究において，LCS(Longest Common Subsequences) を用いた手法を提案している [1]．これは目的のサイト内での全てのセッションについて URL の推移をシーケンスとして抽出し，各シーケンスについて総当たりで LCS を求め，頻出アクセスパターンを発見するものである．このように抽出された頻出アクセスパターンはより効率的なサイトの再構成に役立つ．また，頻出パターンを可視化したり，他の解析手法と併用することによってさらなる効果が見込める．しかしながら，この手法は計算量が大きく，スケーラビリティに問題があった．

この LCS を求める問題は DNA や蛋白質の類似性の比較に用

♡ 学生会員 東京工業大学 大学院 情報理工学研究科 計算工学専攻
toda@de.cs.titech.ac.jp

♢ 正会員 東京工業大学 学術国際情報センター
yokota@cs.titech.ac.jp

いられるため，バイオインフォマティクスの分野でも議論されている．[5] ではこの問題を解く多数のアルゴリズムについて述べられている．これらのアルゴリズムでは平均の時間計算量を小さくできることを示しているが，比較する 2 つのシーケンスの長さをそれぞれ M，N とした場合，最悪の時間計算量は単純な動的計画法で達成される $O(MN)$ 以上に改善していない．平均の計算量の改善を図る場合，Web ログの解析においては解析対象のデータの特徴に適した新たな LCS 抽出手法を用いる必要がある．

また Web ログの解析は DNA や蛋白質の類似性の比較と異なり，個別のシーケンス長は短いものの，多くの LCS 抽出問題を解く必要があるという特徴がある．このため，本論文では個別の LCS 抽出だけでなく，解析全体の効率化も図った手法を提案する．

3. アクセスシーケンス解析

アクセスシーケンス解析において，まず生のログに対し前処理を行い，ユーザのセッションを抽出する必要がある．次にマイニングアルゴリズムを適用することで様々なパターンを生成する．これをパターン分析することで有用なパターンを得ることができる [6]．以降，本論文におけるマイニングの流れを説明する．

3.1 前処理

アクセスシーケンス解析を行う際，蓄積されている生のアクセスログを精錬してマイニングに必要なデータのみを取りだし，セッション毎に抽出する必要がある．以下，その手法について説明する．

まず [7] にある標準的な前処理により，Web ページ間の移動情報に関係のない HTML ファイル以外のファイルへのリクエストを取り除く．検索エンジンのロボットのアクセス情報等も除去する．これにより，Web ログファイルは $\frac{1}{4}$ から $\frac{1}{10}$ 程度に減少する．

また，訪問者が対象サイト内の 1 ページのみを見てセッションを終了した場合，目的とするサイト内での移動情報を得ることができない．このため Web サイトの再構成に利用するのが難しいと判断し，このような情報も取り除くこととする．

さらにアクセスログからセッションを抽出する．本論文では，Cookie を用いて各セッションに一意なセッション ID を割り振る手法を用いる．これにより，プライベート IP で管理される複数ユーザが同一 IP からアクセスした場合等も区別することができる．

3.2 LCS の抽出と頻度集計

まず LCS とは何かについて簡単に説明し，次に LCS の抽出手法と頻度集計について述べる．

3.2.1 Longest Common Subsequences

リスト x の部分列 x_a とリスト y の部分列 y_b の中で両方のリストに含まれるものを共通部分列という．共通部分列の中で最も長いものを Longest Common Subsequences と呼び，頭文字をとって LCS と略する．

二つのリストの中に同じ要素が同じ順序で出現したものが共通部分列なので，LCS が長いということは二つのリストの類似性が高いことを表す．

以下にリスト X，Y の LCS を抽出した例を示す．

$$\begin{aligned} X &= (\underline{A}, F, \underline{B}, \underline{D}, E) \\ Y &= (\underline{A}, \underline{B}, C, \underline{D}, E) \\ LCS(X,Y) &= (\underline{A, B, D}) \end{aligned}$$

これを URL シーケンスに適用することで，URL シーケンス間の類似性を発見することができると考えられる．

3.2.2 LCS の抽出

Wu らは LCS を求める問題と等価である SED (Shortest Edit Distance) を求める問題に関して，効率のいい手法を提案している [8]．この研究ではエディットグラフにおける最遠点方向の探索を優先することで，かなり小さい平均の計算量 $O(NP)$ で LCS を求めることを実現している．二つの文字列 A，B の長さがそれぞれ $N, M (N \geq M)$ であり，D を二つの文字列の差異としたとき，

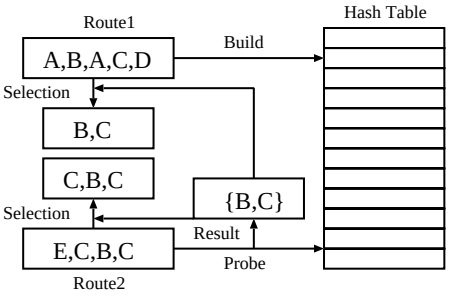

図 1: ハッシュによる LCS 対象の削減
Fig.1 Reduction of objects for LCS with hashing

$P = \frac{D}{2} - \frac{N-M}{2}$ とする．ここから二つの文字列の差異が小さいほど，必要とする時間計算量が小さいことがわかる．

我々はさらに時間計算量を小さくするため，LCS を求める文字列 A，B をハッシュテーブルによりフィルタリングするアプローチ，同一シーケンスの統合を行い冗長な計算を除去するアプローチをそれぞれ提案する．これに関しては 4. 節にて詳細に説明する．

3.2.3 LCS の頻度解析

Web アクセスログから得られたアクセスシーケンスに関して，前節で示した手法により解析を行う．アクセスされた URL の列を，URL を各要素に持つシーケンスと見なす．各シーケンスについて総当たりで任意の 2 つのシーケンスから LCS を抽出する．ここから各 LCS の集計を行い，並べ替えを行うことで，高頻度な LCS パターンを発見する．これに伴い，抽出されるセッションの総数を S とすると，セッション数の二乗に LCS 計算の平均計算量をかけた $O(S^2NP)$ の計算量が必要となる．

4. 解析速度の向上

前節のアクセスシーケンス解析において，LCS の計算に大きな計算量がかかるということを述べた．特にセッション数 S が大きくなるとき，その二乗に比例して時間計算量が増加するため，スケーラビリティに問題がある．

スケーラビリティ改善のアプローチとして，LCS 生成の対象となるセッションの組み合わせと，LCS 算出時の要素数を削減するフィルタリングの手法が考えられる．我々はセッションの組み合わせと要素数の両方を同時にフィルタリングする手法として，ハッシュを用いた手法を提案する．これまで関係データベースの結合演算等において，対象の組み合わせを削減するためにソートやハッシュを用いた手法が提案されている．しかしソートを用いる手法は，リスト内部の部分列を対象とする場合は効果が望めない．ハッシュを用いることで比較の必要がない組み合わせを省略することができる．我々はさらに比較するシーケンス要素の絞り込みにも適用することで LCS の比較回数と LCS の平均計算時間を共に削減する．

これに加えて本論文では解析対象のアクセスシーケンスのうち，まったく同一のものを統合して計算することでさらに実行時間の改善を図る．

これらのアプローチでは最悪の計算量は変わらないが，各シーケンスの平均差異が大きい場合に計算量を下げることができる．フィルタリングは対象とするシーケンスが多様であるほど有効であり，同一シーケンスの統合は完全に同一なシーケンスが多いほど有効であるため，併用することで様々な傾向をもつログに対して効果があると期待できる．

4.1 アクセスシーケンスのフィルタリング

本論文では図 1 に示すような方法でアクセスシーケンスのフィルタリングを行う．以下にその方法を解説する．

1. Route1 が要素に持つ URL をハッシュテーブルにマッピングする (Build).

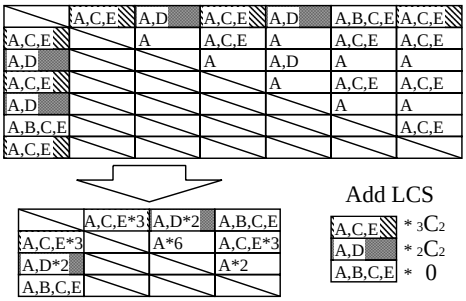

図 2: 同一シーケンスの統合
Fig.2 Duplication elimination of the same sequences

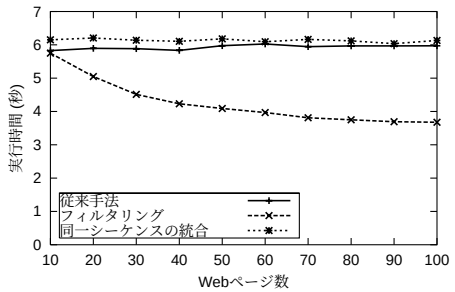

図 3: [予備実験]Web サイトの規模と性能向上の関係
Fig.3 Website size influence on performance

2. Route2 が要素に持つ URL とハッシュテーブルに含まれる URL を比較する (Probe).
3. 2 の結果を用いて Route1，Route2 をそれぞれフィルタリングする (Selection).

Route1，Route2 の長さをそれぞれ M，N として時間計算量について見てみると，1 では $O(M)$，2 では $O(N)$，3 では $O(M+N)$ の計算量が必要になる．つまり，フィルタリング全体で $O(M+N)$ の時間計算量がかかることになる．総当たりで LCS を計算する今回の場合，1 で作ったハッシュテーブルを再利用することで，計算量を削減することができる．また空間計算量に関しては，ハッシュテーブルを記憶するために $O(M)$ の計算量が必要である．

このフィルタリングを行うことで，LCS の計算を行う際に，共通する部分のみを抽出した結果を計算することができる．当然のことながら，二つのシーケンスに共通して現れない要素を取り除いても LCS の計算結果に影響はない．また 3.2.2 節で説明したように，$O(NP)$ アルゴリズムでは二つの文字列の差異が小さいほど計算量が小さくなるという特徴がある．このため，一方のシーケンスのみにしか現れない要素を取り除くことで，効率的に計算が行うことができると期待できる．本手法では非共通要素をフィルタリングするため，一般にアクセスパターンが多岐にわたるほど効果的であるとも言える．

4.2 同一シーケンスの統合

解析を行うアクセスシーケンスの中には完全に同一のものが複数含まれていることがあり，これらの LCS 計算をそれぞれ行うことは冗長である．このため，同一のシーケンスを 1 つにまとめて計算回数を小さく抑えるアプローチをとる．本論文で行う同一シーケンスの統合を図 2 に示す．以下，その方法について解説する．

1. 解析対象の全てのアクセスシーケンスをスキャンし，同一のものを 1 つに統合する．
2. 統合したシーケンスの数に応じて重み付けを行う．
3. 従来手法と同様に全てのシーケンスについて総当たりで任意の 2 つのシーケンスから LCS を抽出する．この際，得られる LCS を各シーケンスの重みの積の数加える．
4. 統合したシーケンス同士の LCS として，n 個のシーケンスを統合した場合 ${}_nC_2$ 通りの組み合わせがあるので，この数のシーケンスを LCS として加える必要がある．

上記の手法により統合を行わない場合と同じ結果を効率的に出力することができる．ただし，この手法も 4.1 節で述べた手法と同様，最悪の計算量を減らすことはできない．

統合に必要な計算量は 1，2 においてシーケンス全体をスキャンする $O(S)$，4 で統合したシーケンス自体を LCS に加える $O(S')$ の計算量であり，全体として $O(S)$ の計算量で効率よく計算が行えると思われる．ここで S，S' はそれぞれ統合前後のセッション数を表す ($S > S'$)．本手法では同一のシーケンスの冗長な計算を除去しているため，一般にアクセス偏りが大きい場合に特に有効であると言える．実際の効果については次節で示す．

表 1: 実験環境
Table 1 The experimental environment

CPU	Intel Xeon 2.40GHz
メモリ	PC2100 SDRAM 3328MB
HDD	IBM Ultrastar 18GB 15000rpm
OS	Linux 2.4.20
Java 環境	Sun JDK 1.4.1_01
Perl 環境	Perl 5.6.1

表 2: 対象 Web サイト諸元
Table 2 The website information

URL	http://yokota-www.cs.titech.ac.jp
主要ページ数	約 50
サーバプログラム	Apache 1.3.27
平均リクエスト数	12063 requests/week
言語	日本語/英語

4.3 計算時間の測定

実際のアクセスログの解析を行う前に人工的なアクセスログデータを生成して表 1 の実験環境で予備実験を行った．これを図 3 に示す．

ここでは Web ページ数を変数として，乱数でアクセスされた URL を生成し，適当なセッション ID を付与することで，セッション数 686，セッション長 5 に固定した人工的なアクセスログデータを生成した．現れる URL の数を増加させることで Web サイトの規模が大きくなったことを表現し，その解析時間を計測している．ただし，ここでは完全にランダムアクセスを行うことを前提としている．

従来手法を適用して解析した場合はほぼ一定の実行時間であるのに対し，ハッシュを用いたフィルタリングを行った場合，現れる URL の数の増加に伴い実行時間が減少しているのが分かる．また，同一シーケンスの統合では従来手法より平均 0.2 秒ほど実行時間が増加している．

これはアクセス偏りがないため，フィルタリングでは性能改善が認められるものの，完全に同一のシーケンスが現れることは極めて少なく，同一シーケンスの統合に関してはオーバヘッドの分，実行時間が増加したと考察される．

次に本論文で提案する手法の有効性を確認するため，我々の研究室で公開している Web サイトのアクセスログの解析に本手法を適用した．対象とした Web サイトの特徴は表 2 の通りである．

前処理を Perl で，アクセスログ解析を Java で実装した実験システムで性能測定を行った．実験環境に関しては予備実験と同様に表 1 に示す通りである．

2003 年 5 月 12 日から 7 月 27 日までの Web サーバへのリクエストに対するアクセスログを使って解析を行った．前処理を行うことで，20.4MB の生のログから 3.64MB の精錬されたログが得られた．また，前処理を行った後のセッション数は 1781，1 セッション当たりの平均 URL シーケンス長は 6.65 ページであった．

アクセスログのセッション数を増加させた時の実行時間の変化

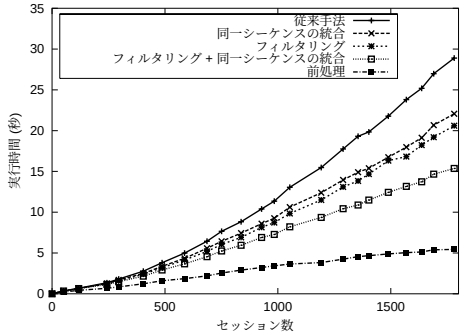

図 4: アクセスシーケンスの LCS 解析の実行時間
Fig.4 Execution time for Web log mining with LCS

を測定したものを図 4 に示す．図 4 では従来手法，ハッシュによるフィルタリングを加えた手法，同一シーケンスの統合を加えた手法，これら 2 つを加えた手法の実行時間をそれぞれ示す．また，各手法の実行時間に共通して用いる前処理の時間は別に示しているが，各手法の実行時間にも含まれている．

図 4 から分かるように前処理にかかる計算時間はほぼ線形に増加するのに対し，従来の LCS の抽出にかかる計算時間は線形以上のペースで増加している．大規模な Web ログファイルの解析を行うことを考えた場合，LCS を抽出する段階での時間計算量を小さくすることが効果的であると思われる．

アクセスシーケンスが多様である場合，ハッシュによるフィルタリングの効果が大きくなると思われる．また，共通して現れるシーケンスが多い場合に，同一シーケンスの統合により計算効率が向上できる．このことから 2 つの手法を組み合わせることで，様々な特徴を持つ Web ログ解析の計算量を抑えることができると考えられる．実際に本実験では従来手法に比べ 2 つの提案手法でそれぞれ実効時間を短縮でき，これらは組み合わせることでさらに効果があがっていることが図 4 から見て取れる．また，解析対象のアクセスログの増加に伴い実行速度の改善率が向上していることもわかり，スケーラビリティの向上が達成できたと言える．

解析対象の Web ログが増大することで，当然結果として出力される LCS は増大するが，これに伴うディスク I/O にも大きな時間が必要となる．しかし，実際に Web サイトの解析を行う場合，ユーザの設定する閾値以上の頻度で現れる LCS を出力することが考えられる．このため本手法を実際に利用する際には，Web ログファイルの増加に伴う計算時間の増大は図 4 に見られるよりも小さいものであると思われる．

5. おわりに

本論文では LCS を用いたアクセスシーケンス解析において，ハッシュを用いたフィルタリング，同一シーケンスの統合を行う手法をそれぞれ提案した．また，実際に実験システムを用いてその有効性を検証した．

本論文では Web サイトが大規模になるに伴い，ユーザのアクセスパターンが多様化することに着目し，ハッシュを用いたフィルタリングを用いることで，性能の向上を図った．これにより，LCS の比較回数と LCS の平均計算時間を共に減少させることができた．さらに，完全に同一のシーケンスについてそれぞれ LCS の抽出を行うことは，冗長な計算を行っていることであると考えられる．これらを統合して LCS の抽出を行うことでこのような冗長な計算を除去することができた．そして 2 つの手法を組み合わせることで実行時間をより短縮することができ，その改善率はアクセスログの増加に伴い向上することを確認した．これはアクセスログ解析のスケーラビリティが向上できたことを示している．

本論文では我々の研究室が公開している Web サイトのアクセスログを解析することで，提案手法の有効性を示した．実験結果からさらに大きな Web サイトのアクセスログでも大きな効果が期待できるため，実際により大きな Web サイトのアクセスログに本手法を適用し，その有効性を検証することは今後の課題である．また，大規模サイトを対象とした解析のさらなる性能向上のため，計算の並列化を考えることも必要である．

また本論文で提案したアプローチは最悪の場合の計算量を小さくするものではなく，解析対象のアクセスログの特徴により効果が異なる．4.3 節でも触れたように，ユーザのアクセスパターンが多岐にわたるような場合にはフィルタリングの効果が大きく，逆にアクセスパターンが集中するような場合には同一シーケンスの統合の効果が大きい．このように様々な特徴を持つアクセスログに対して効果的な手法であると言えるが，最悪の計算量を小さくする計算効率の向上に関しては今後の課題である．

さらに，アクセスログの解析結果から得られる頻出アクセスパターンの可視化と，利用性の向上が必要である．特に大規模なサイトに関して，マイニングアルゴリズムを適用することで得られたパターンを分析し，Web サイトの再構成に活用する方法についても今後の課題である．

[謝辞]

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 (15017233) の助成により行われた．

[文献]

[1] 宇根田純治 , 横田治夫. Web ログの共通シーケンス解析. In **信学技法**, *DE2002-2*. 電子情報通信学会, 2002.

[2] Ramakrishnan Srikant and Yinghui Yang. Mining web logs to improve website organization. In *Proc. of World Wide Web Conf.*, pages 430–437, 2001.

[3] J. Pitkow and K. Bharat. Webviz: A tool for world wide web access log analysis. In *Advance Proceedings First International World-Wide Web Conference*, pages 271–277, May 1994.

[4] Myra Spiliopoulou and Lukas C. Faulstich. WUM: a Web Utilization Miner. In *Workshop on the Web and Data Bases (WebDB98)*, pages 109–115, 1998.

[5] A. Apostolico. String editing and longest common subsequences. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 2 Linear Modeling: Background and Application, pages 361–398. Springer-Verlag, Berlin, 1997.

[6] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.

[7] A. Banerjee and J. Ghosh. Concept-based clustering of clickstream data. In *Proc. 3rd Intl. Conf. on Information Technology*, pages 145–150, Dec 2000.

[8] Sun Wu , Udi Manber , Gene Myers , and Webb Miller. An o(np) sequence comparison algorithm. Information Processing Letters , 35 : 317–323, 1990.

戸田 誠二 Seiji TODA
平 14 東工大・工・電電卒．同大大学院・情報理工・計算工・修士課程在学中．HDD 制御，アクセスログ解析等，データ工学の研究に従事．日本データベース学会学生会員．

横田 治夫 Haruo YOKOTA
昭 55 東工大・工・電物卒．昭 57 同大大学院・情報・修士課程了．同年富士通 (株) 入社．同年 6 月 (財) 新世代コンピュータ技術開発機構研究所．昭 61(株) 富士通研究所勤務．平 4 北陸先端大・情報・助教授．平 10 東工大・情報理工・助教授．平 13 東工大・学術国際情報センター・教授．工博．主としてデータベース，データ工学向けの並列アーキティクチャ等に関する研究に従事．日本データベース学会，電子情報通信学会，情報処理学会，人工知能学会，IEEE，ACM 各会員．