3F2-02

大規模データマイニングのための頻出データアイテム発見アルゴリズムとネットワークデータへの応用

Efficient Algorithms for Mining XML Data Streams

川副真治[*1] 浅井達哉[*1] 有村博紀[*1]
Shinji Kawasoe Tatsuya Asai Hiroki Arimura

[*1]九州大学 大学院システム情報科学府・研究院
Department of Informatics, Kyushu University

The *frequent item list problem* is the problem of, given a stream of N data items and a threshold $0 < \sigma < 1$ for the minimum frequency, finding all *frequent data items* that have frequency above σ, and it has many useful applications such as network monitoring, text mining, and iceberg query processing. In this paper, we propose a space-efficient two-pass algorithm called *DoubleScan* that exactly solves the frequent item list problem in $O((1/\sigma)\log N)$ space by simply running the approximation online algorithm called LossyCounting recently developed by Manku and Motowani (VLDB'02). Then, we present theoretical and empirical comparion of DoubleScan algorithm and the well known two-pass exact algorithm, called Partition, by Savasere *et al.* (VLDB'95), and show that DoubleScan is superior to Partition in space order of magnitude.

1. はじめに

多くの大規模データ解析事例において，潜在的に非常に大きな数のデータアイテム全体 $\mathcal{I}$ に対して，データアイテムの集合であるデータベース $\mathcal{D}$ から，事前に与えられたしきい値 $0 \leq \sigma \leq 1$ 以上の頻度を持つごく少数のデータアイテムを発見する問題が重要になる．このような問題を，**頻出アイテム発見問題** (The frequent item list problem) [7] とよぶ．

例えば，ネットワークログ解析では，データベースは，ネットワーク通信データを `tcpdump` 等のパケット解析ツールで解析して得られたログデータであり，ここからあらかじめ専門家が指定した複数の属性の値を組み合わせることで得られた属性の組がデータアイテムとなる．具体的には，送信ホストと，通信タイプ，サービス（ポート）の組 (`216.239.53.101, tcp, http`) や，受信ホストと送信ホスト，ポートの組 (`133.5.24.1, 216.239.53.101`) 等がデータアイテムの例であり，分析者は，一定の頻度以上出現する組に注目し，不正侵入検出やネットワーク障害の兆候を発見する．

この頻出アイテム発見問題は，代表的なデータマイニングである結合規則発見問題と比べて，基本となる少数の属性があらかじめ固定されているため，一見自明な問題にみえる．実際，図 1 の単純なアルゴリズム（以後，Naive アルゴリズムとよぶ）で計算可能である．ここで，データベース $\mathcal{D} = (d_1, \ldots, d_N) \in (\mathcal{I})^*$ は，長さ N のデータアイテム列とする．

しかし，この自明なアルゴリズムの領域計算量は $O(N)$ であり，可能なアイテム数 $|\mathcal{I}|$ とデータサイズ $N = |\mathcal{D}|$ が非常に大きいときには，きわめて非効率的である．実際に，ネットワークログデータでは，一度だけ出現するようなものも含めるとデータアイテム数は膨大なものとなり，固定された主記憶容量よりもはるかに大きな値になり得る．

そこで本稿では，**少ない領域とデータ走査数で，厳密にすべての頻出アイテムを計算する**効率良いアルゴリズムを開発することを目標として研究をおこなう．

連絡先: 有村 博紀，九州大学大学院システム情報科学研究院，〒812-8581 福岡市東区箱崎 6-10-1，TEL: 092-642-2688，FAX: 092-642-2698，E-mail: `arim@i.kyushu-u.ac.jp`

1.1 定義

形式的には，頻出アイテム発見問題を次のように定義する．アイテム（item）の集合を $\mathcal{I}$ とする．サイズ N の**データベース** (database) とは，アイテムの配列 $\mathcal{D} = d_0 \cdots d_{N-1}$ $(d_i \in \mathcal{I})$ である．$\mathcal{D}$ の第 i 番目の要素を $\mathcal{D}[i]$ で表す．$\mathcal{D}$ におけるアイテム p の出現数を次のように定義する：

$$hit_{\mathcal{D}}(p) = \Sigma_{i=0}^{N-1}[d_i = P].$$

ここに，関数 $[E]$ は式 E が真のとき 1 をとり偽のとき 0 をとる．p の**出現頻度**は $freq_{\mathcal{D}}(p) = hit_{\mathcal{D}}(p)/|\mathcal{D}|$ である．このとき，**最小頻度**とよばれる正数 $0 < \sigma \leq 1$ に対して，p が σ **頻出**であるとは，$freq_{\mathcal{D}}(p) \geq \sigma$ が成立することをいう．

頻出データアイテム発見問題
入力: 長さ N のアイテム配列 $\mathcal{D}$ と最小サポート $0 < \sigma \leq 1$.
問題: すべての σ 頻出なアイテム $p \in \mathcal{I}$ を出力せよ．

1.2 主結果

本稿では，Manku と Motowani（VLDB'02）のオンラインアルゴリズム Lossy Counting をデータ上で 2 回走らせることで，きわめて小さい記憶領域で頻出データ発見問題を厳密に解くことを示す．具体的には，図 3 のアルゴリズム DoubleScan は，ちょうど 2 回のデータ走査と領域計算量 $O(\frac{1}{\sigma}\log N)$ を用いて，頻出データアイテム発見問題を厳密に計算することを示す．

この結果の鍵は，今回示した頻度列の巡回 (rotation) に関する次の組み合わせ的結果である：

補題：要素の総和が N となる長さ N の任意の整数配列 A に対して，次の性質をみたす巡回 B が存在する：B のすべての接頭列 $B[0:i]$ $(0 \leq i < N)$ に対して，その要素の総和 $\sum_{j=0}^{i} B[j]$ が長さ i 以上になる．

これにより，Manku と Motowani のアルゴリズムが 2 回のデータ走査の間に，すべての頻出アイテムの出現を正しく検出することが保証される．

さらに，合成データと実データを用いた計算機実験で，提案のアルゴリズムを次の節で紹介する Naive と Partition の 2 つ

のアルゴリズムと比較する．結果として，提案の DoubleScan アルゴリズムは計算速度は遅いが，著しく小さな領域計算量で頻出データアイテム発見問題を解くことが分かった．また，逐次的な入出力特性により，外部記憶アルゴリズムとして用いることができる．このことから，DoubleScan アルゴリズムは限られた記憶領域の元で，非常に大きなデータ集合に対して頻出アイテムを発見する場合に有効である．

1.3 本稿の構成

2 節で，関連研究について述べる．3 節で，Manku と Motowani（VLDB'02）のオンラインアルゴリズム Lossy Counting を簡単に紹介し，われわれが提案する DoubleScan アルゴリズムを与える．ここで，上の定理 2 を証明し，これを用いて主結果を示す．4 節で，実験結果について述べ，5 節でまとめを述べる．

2. 関連研究

この問題を解く自明な方法は，図 1 に示した Naive 法である．Naive 法は，1 回のデータ走査しか行なわないが，$O(|\mathcal{I}|)$ の主記憶を必要とし，最悪時の領域計算量は $O(N)$ である．

Algorithm Naive

foreach $i = 1, \ldots, |\mathcal{D}|$ **do:**

- **if** $(d = \mathcal{D}[i]) \in \mathcal{C}$ **then** $\mathcal{C}[d] = \mathcal{C}[d] + 1$;
- **else** $\mathcal{C}[d] = 1$;

図 1: Naive アルゴリズム

これに対して，Savasere 等 [8] は，二回のデータ走査でこの問題を厳密に解く分割型アルゴリズム Partiton を与えた．これは，データを一定サイズのブロックに分割し，ブロックごとに計算した頻出アイテムを併合するアイディアを用いている．彼らは領域量の解析を与えていないが，文献 [6] では，この領域計算量を解析し，ブロック長を $B = \sqrt{N/\sigma}$ ととったときに，Partition の領域計算量が $O(\sqrt{N/\sigma})$ となることを示した．さらに，Partition が使用する領域量が少なくとも $\Omega(\sqrt{N/\sigma})$ となるようなデータベース $\mathcal{D}$ が存在することもわかった．

最近，Manku と Motowani（VLDB'02）は，すべての頻出アイテムを厳密に求めるのではなく，1 回のデータ走査で近似的に頻出アイテムを求める近似計数技法を提案した．与えられた誤差パラメータ ε に対して，彼らの近似アルゴリズム Lossy Counting は，すべての頻出アイテムを発見し，それらの頻度を真の頻度から高々ε 程度高い程度で推測可能である．反対に，非頻出アイテムを見つける可能性もあるが，$\sigma - \varepsilon$ 以下の頻度をもつアイテムは決して発見されないことが保証される．

3. 提案アルゴリズム

本節では，頻出データアイテム発見問題を 2 回のデータ走査と $O(\frac{1}{\sigma}\log N)$ 領域で解く走査型 2 パスアルゴリズム DoubleScan を提案する．さらに，その理論的解析を行なう．

3.1 アルゴリズム

図 3 に，提案のアルゴリズム DoubleScan を示す．このアルゴリズムは，Manku と Motowani[7] の 1 パス近似アルゴリズム LossyCounting を流用して構成したもので，2 回繰り返

Algorithm Partition

入力: 長さ N のアイテム配列 $\mathcal{D}$ および, 最小サポート値 $0 < \sigma \leq 1$, ブロックサイズ $0 < B \leq N$.

ステージ 1:

- アイテム配列 $\mathcal{D}$ を，$n = \lceil N/B \rceil$ 個のサイズ B のブロック $\mathcal{D} = \mathcal{B}_1 \cdots \mathcal{B}_n$ に分割する．
- 各 $1 \leq i \leq n$ に対して，ブロック $\mathcal{B}_i$ を主記憶に読み込み，$\mathcal{B}_i$ 内で出現回数 $freq_{\mathcal{B}_i}(p) \geq B\sigma$ を満たすすべてのアイテム $p \in \mathcal{B}_i$ を求め，局所的候補集合 $\mathcal{C}_i$ に格納する．（パス 1）

ステージ 2

- 大域的候補集合 $\mathcal{C} := \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_n$ を計算する．
- アイテム配列 $\mathcal{D}$ を左から右へ走査しながら，各候補アイテム $p \in \mathcal{C}$ の出現回数 $g(p)$ を計算する．（パス 2）
- $g(p)/N \geq \sigma$ を満たすすべてのアイテム $p \in \mathcal{C}$ を頻出アイテムとして出力する．

図 2: Partition アルゴリズム

してアイテム配列を走査する点以外は，LossyCounting とまったく同一のアルゴリズムである．領域計算量の解析は，パラメータの意味が変わるが, Manku と Motowani[7] の議論がそのまま成立する．

しかし，まったく同じ走査を 2 回続けてくりかえすことで，1 パス近似アルゴリズムが，2 パスの厳密計算アルゴリズムとして働くことの証明はそれほど自明ではない．この正当性の証明を本節で行なう．

アルゴリズム DoubleScan は，初めにブロック長 $= \lceil 1/\sigma \rceil$ を計算し，入力アイテム配列 $\mathcal{D}$ を n 個のサイズ B のブロック $\mathcal{D} = D_1 \cdots D_n$ に区切る．このブロック長 B の選択がアルゴリズムの鍵である．その後，アイテム配列 $\mathcal{D}$ 全体を左から右にちょうど 2 回走査しながら，各ブロックを順に主記憶に読み込み，出現したアイテムを候補集合 $\mathcal{C}$ に追加し，その後，出現するたびにその計数を更新する．

候補集合 $\mathcal{C}$ 中の各候補アイテム $p \in \mathcal{C}$ は，次の情報を属性としてもつ:

- 候補集合に挿入されたときの時刻 $p.s \geq 0$.
- 出現回数のカウンタ $p.f \geq 0$.

この候補集合 $\mathcal{C}$ の管理の方針は，つぎのとおりである．

- 初めて出現したアイテムは，$\mathcal{C}$ に追加する．
- $\mathcal{C}$ に追加されたアイテムは，条件 $p.s + p.f - 1 \geq current$ が成立する間は $\mathcal{C}$ に保持しつづける．

3.2 解析

このアルゴリズム DoubleScan は，次のようなアイディアに基づいている．今，ある頻出アイテムがデータ系列 $\mathcal{D}$ に $M \geq \sigma$ 回出現していると仮定する．すると，今 $\mathcal{D}$ を長さ $B = \lceil 1/\sigma \rceil$ の等長ブロックに分割したとする．すると，p は頻出であるから，平均して各ブロックに 1 回以上出現すると期待できる．そこで，アイテム p に対して，それが隣接したブロックに連続

Algorithm DoubleScan

入力: 長さ N のアイテム配列 $\mathcal{D}$ と最小サポート値 $0 \leq \sigma \leq 1$.

- $B := \lceil 1/\sigma \rceil$; /* ブロック長の設定 */
- $\mathcal{D}$ を，$n = \lceil N/B \rceil$ 個のサイズ B のブロック $\mathcal{D} = D_0 \cdots D_{n-1}$ に分割する．
- $\mathcal{C} := \emptyset$; $\mathcal{F} := \emptyset$ /* 初期化 */
- 以下の操作を, $current = 0, \ldots, 2n-1$ に対して繰り返す:
  - $k := current \bmod n \ (0 \leq k \leq n-1)$.
  - k 番目のブロック D_k を主記憶に読み込み，各アイテム $p \in D_k$ に対して，もし $p \in \mathcal{C}$ ならば，$p.f := p.f + 1$ とカウンタを増やす．それ以外のとき，p を $\mathcal{C}$ に加えて，$p.f := 1$ と $p.s := current$ で初期化する．
  - 条件 $p_j.f < current - p_j.s + 1$ を満たすすべての候補 $p \in \mathcal{C}$ を $\mathcal{C}$ から削除する．
  - 条件 $current - p.s + 1 = n$ を満たすすべての候補 $p \in \mathcal{C}$ を頻出アイテムとして $\mathcal{A}$ に加え，同時に $\mathcal{C}$ から削除する．
- **Return** $\mathcal{F}$;

図 3: DoubleScan アルゴリズム

して 1 回以上出現する限り，p を候補集合 $\mathcal{C}$ に保持する戦略をとる．こうすると，頻度が σ 以上のパターンは長期にわたって $\mathcal{C}$ にとどまりつづけ，頻度が σ 未満のパターンは $\mathcal{C}$ から除去されると期待できる．しかしアイテムの出現は任意であるから，実際はアイテムの出現は疎なところと，密なところが存在する．そこで，候補 $p \in \mathcal{C}$ が $\mathcal{C}$ に追加された後の累積出現数 $p.f$ を用いて, p の管理を行なうという戦略を採用している．

Manku と Motowani[7] による LossyCounting の領域計算量の解析で，誤差パラメータ ε を最小サポート σ で置きかえることで，直ちに次の補題が成立する．

補題 1 (Manku と Motowani[7]) 図 *3* のアルゴリズム DoubleScan は，ちょうど *2* 回のデータ走査と領域計算量 $O(\frac{1}{\sigma} \log N)$ を用いて，頻出データアイテム発見問題を厳密に計算する．

入力として，$\mathcal{D}$ 上のブロック系列 $(D_i)_{i=1,\ldots,2n-1}$ を仮定しよう．データ系列を 2 回走査して得られるブロック列を $\mathcal{S} = (D_i)_{i=0,\ldots,2n-1}$ とし，任意のアイテムを $p \in \mathcal{I}$ とおく．このとき，p に関する $\mathcal{S}$ 上の**長さ ℓ の成功連** (successful run) とは，連続した $\mathcal{S}$ の添え字の列 $0 \leq i_0, \ldots, i_{\ell-1} \leq 2n-1$ で次の条件をみたすものをいう：

- 添え字は連続している．すなわち，任意の j に対して，$i_{j+1} = i_j + 1$ が成立する．
- p が候補となってから現在までの累積出現数から，候補となってから現在までのブロック数 (時間) を引いたものが非負である．すなわち，任意の j に対して，$p_j.f \geq j - p_j.s + 1$ が成立する．

DoubleScan の正当性は，任意の頻出アイテム p が長さ n の成功連をもつかどうかにかかっている．この鍵は，今回示した頻度列の巡回 (rotation) に関する次の組み合わせ的性質である．詳細については，文献 [5] を参照されたい．

表 1: 実験に用いたデータ

名称	サイズ N	異なりアイテム数 I	生成方法
Ohsumed	12MB	740,000 個	英文テキスト
Uniq	1MB	10,000 個	一様分布乱数
Zipf	1MB	10,000 個	Zipf 分布乱数

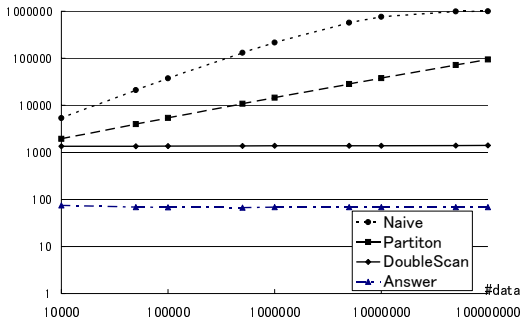

図 4: 実験 1．データサイズと領域計算量

補題 2 要素の総和が N となる長さ N の任意の整数配列 A に対して，次の性質をみたす巡回 B が存在する：B のすべての接頭列 $B[0:i]$ $(0 \leq i < N)$ に対して，その要素の総和 $\sum_{j=0}^{i} B[j]$ が長さ i 以上になる．

次の補題がただちに成り立つ．

補題 3 アルゴリズム DoubleScan を $\mathcal{D}$ 上のブロック系列 $(D_i)_{i=1,\ldots,2n-1}$ 上で走らせたと仮定する．このとき，アイテム p が $\mathcal{D}$ 上で σ 頻出であることと，p に関する長さ n の成功連が存在することは同値である．

補題 2 と補題 3 から，本稿の主結果が導かれる．

定理 1 図 *3* のアルゴリズム DoubleScan は，ちょうど *2* 回のデータ走査と領域計算量 $O(\frac{1}{\sigma} \log N)$ を用いて，頻出データアイテム発見問題を厳密に解く．

4. 実験

開発したアルゴリズム DoubleScan の性能を評価するために，人工データと実データを用いて，頻出アイテム発見の計算機実験を行った．

4.1 データ

実験には，表 1 に示すデータを用いた．Ohsumed は，英文医療文献データ MEDLINE の記事 86MB からなるデータ OHSUMED [*1] で，各異なり単語をアイテムとしたものである．Unif と Zipf は，それぞれ，一様分布乱数 $P(x) = 1/I$ と Zipf 分布 $P(x) = x^{-\alpha}$ を用いて生成した合成データである．ここに $\alpha = 1$ であり，出現頻度の上位 5000 個が全体の約 70% を占める．Ohsumed データを，Zipf 分布で近似すると，$\alpha = 1.188$ になる．

*1 http://ftp.ics.uci.edu/pub/machine-learning-databases/ohsumed/

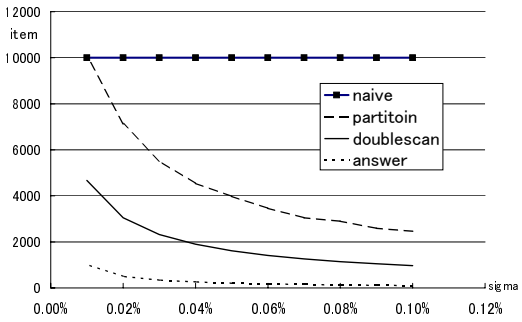

図 5: 実験２．最小サポート値と領域計算量

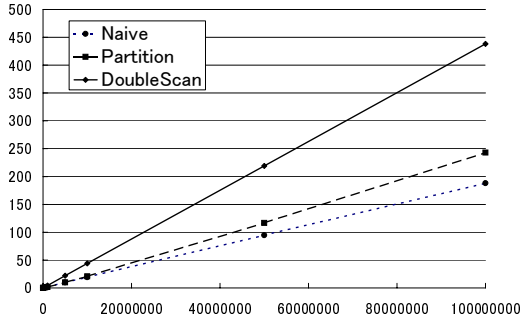

図 6: 実験３．データサイズと計算時間

4.2 実験方法

次の 3 つのアルゴリズムを，C++言語と STL(C++ Standerd Template Library) を用いて実装した．

* NAIVE: 2 節のアルゴリズム Naive を，候補集合 $\mathcal{C}$ に STL の拡張可能なハッシュ辞書を用いて実装したもの．
* PARTITION: 2 節で示した Savasere 等の分割型 2 パスアルゴリズム Partition(図 2) を実装したもの．定理??にしたがい，ブロックサイズを $B = \sqrt{N/\sigma}$ とした．
* DOUBLESCAN: 3 節で提案した操作型 2 パスアルゴリズム (図 3) を実装したもの．

すべての実験は PC（Pentium4, 2.5GHz, RAM 1GB, WindowsXP）上で行った．

4.3 結果

実験１:図 4 では，$\sigma = 0.1(\%)$ と固定し，サイズを N =10KB ～100MB で変化させた．このデータでは，解の異なり数は一定にしていることに注意．プロットは，上から Naive, Partition, DoubleScan アルゴリズム，正解数（頻出パターン数）の順である．参考として表 2 に，データサイズ $N = 5 \times 10^6$(item) のときの各アルゴリズムの領域量を示す．

表 2: アルゴリズムによる領域量の比較

	Naive	Partiton	DoubleScan	Answer
領域量	572,069	28,460	1,388	69
相対比	×412	×21	×1	×0.04

これより，正解数と比較して，DoubleScan の領域量は，その × 10 程度であり，Partition の × 100 や Naive の × 1000 前後に比べて圧倒的に領域効率が良い．

ここでは示せなかったが，ohsumed データや Zipf データ等でも同様の傾向が示された．さらに，実データに多いこのような分布が偏ったデータに対しては，DoubleScan は一様分布より小さな領域量しか使わなかった．

実験２：図 5 に最小サポート値と領域計算量に関する結果を示す．一般に最小サポート σ が小さいほど（図の左側），正解数と領域量が急速に大きくなる．図から，DoubleScan と Partition が，サポート値に自動的に適応して，領域量を節約していることと，DoubleScan の領域量が特に小さいことがわかる．注目すべき点として，サポート値が低い値（$\sigma = 0.01\%$）で，Partition は Naive と同程度に領域効率が悪いのに対して，DoubleScan はその 2 倍程度領域効率が良い．

実験３：図 6 にデータサイズを変化させた場合の，計算時間を示す．図の上から，DoubleScan, Partition, Naive の順でプロットしており，DoubleScan は Naive の 2 倍程度の計算時間を必要とすることがわかる．また，どのアルゴリズムもデータサイズに関して線形時間であり，規模耐性は良い．さらに，図 4 から，もし最大領域量が定数 (例えば 10K 個のアイテム分) に限定されたとき，Naive は，$N = 10^4$ 個超で，Partition は $N = 10^6$ 個弱でメモリ不足になるのに対して，DoubleScan の領域量は実験的には正解数の定数倍であり，長期に動きつづけると予想される．

実験４：最後に，実際のネットワークデータ上で，Naive と DoubleScan の性能比較実験をおこなった．データには，KDD Cup1999 IDS Data [*2]を用いた．これは，ネットワーク不正侵入検出のためのテストデータであり，このデータセットから，1 節でふれたような (サービス, 発信元，送信先) の 3 つの属性値の組をとりだし，その一部をデータベースとした．データベース $\mathcal{D}$ のサイズは，$|\mathcal{D}|$ =3,013,862 (アイテム) であり，64,636 個の異なるアイテムを含む．最小頻度は $\sigma = 0.1\%$ (3,014 個) であり，頻出アイテム数は 24 個とデータサイズに比べて小さい．

本来の不正侵入検出は，分類規則による予測を行なったり，外れ値検出を行なう等の機械学習手法を用いることが多いが (e.g., [9]), ここでは，単にアルゴリズムの領域効率の評価のために頻出パターン発見を行なった．

表 3: ネットワークデータでの性能比較

Algorithm	Time (sec)	Space (item)	実メモリ
Naive	60.84	64,636	8.8MB
DoubleScan	100.67	2,893	1.6MB

表 3 に実験結果を示す．候補集合のサイズでは，DoubleScan は，Naive の 30 倍程度効率が良い．逆に，計算時間に関しては，Naive の 1.8 倍程度 DoubleScan が大きい．この表から実際のネットワークデータにおいても，DoubleScan は計算時間はやや遅いが，領域効率は良いことがわかる．

5. まとめ

本稿では，頻出データアイテム発見問題に対する領域効率の良いアルゴリズム DoubleScan を提案した．これは，データ走査に DoubleScan は，長さ N のデータに対して，２回の走査と，N の線形時間および領域 $O(\frac{1}{\sigma}\log N)$ 領域で，すべての頻出アイテムを厳密にすべて見つけることを示した．さらに，人工データとテキストデータ上で領域計算量を，自明なアルゴリズム Naive および分割型 2 パスアルゴリズム Partition と比較した．

*2 http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

実験結果をまとめると，DoubleScan は，Partition に比べて計算速度では 2 倍程度劣るが，領域効率は 10 倍程度と非常によく，また適応性に優れている．したがって，非常に大量のデータを対象とする応用に，きわめて有効であると期待される．ネットワークデータの解析や，テキストマイニングなど，実際のデータマイニング問題への適用が今後の課題である．

最近になって，Karp, Papadimitriou, Shenker 等 [4] が，頻出アイテム問題を 2 回のデータ走査と領域計算量 $O(1/\sigma)$ で厳密に解くアルゴリズムを提案している．彼等のアルゴリズムと DoubleScan を実データ上で比較し，実際の解の個数に対する領域効率を比較することも今後の課題である．

参考文献

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules, In Proc. VLDB'94, 487–499, 1994.

[2] S. Brin, R. Motwani, C. Silverstein. Beyond Market Baskets: Generalizing Association Rules to Correlations, In Proc. SIGMOD Conference 1997, 265–276, 1997.

[3] C. Estan and G. Verghiese. New directions in traffic measurement and acconting, In Proc. ACM SIGCOMM Internet Measuerment Workshop, November 2001.

[4] R. M. Karp, C. H. Papadimitriou, S. Shenker, A simple algorithm for finding frequent elements in streams and bags, Manuscript, 2002. (Available at the authors home page)

[5] 川副真治，データベースから頻出アイテムを求める領域効率の良い 2 パスアルゴリズム，九州大学大学院システム情報科学府，情報理学専攻，修士論文，平成 15 年 2 月．

[6] 川副真治，有村博紀，領域効率の良い頻出データアイテム発見アルゴリズム, Proc. DEWS 2003, 電子情報通信学会，データ工学研究会，March 2003.

[7] G. S. Manku, R. Motwani, Approximate Frequency Counts over Data Streams, In Proc. VLDB'02, 2002.

[8] A. Savasere, E. Omiecinski, S. Navathe, An Efficient Algorithm for Mining Association Rules in Large Databases, In Proc. VLDB'95, 1995.

[9] K. Yamanishi, J. Takeuchi, A Unifying Framework for Detecting Outliers and Change Points from Non-Stationary Time Series Data, In *Proc. SIGKDD-2002*, ACM, 2002.