バーストを伴うデータストリームのマイニングその他について

岩沼宏治 （山梨大学）
共同研究者： 伊藤秀志，山本泰生 （山梨大学）

論理と推論の理論，実装，応用に関する合同セミナー 2013/07/25,
北海道大学工学部 ERATOセミナ室

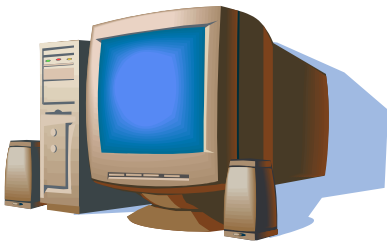
発表構成

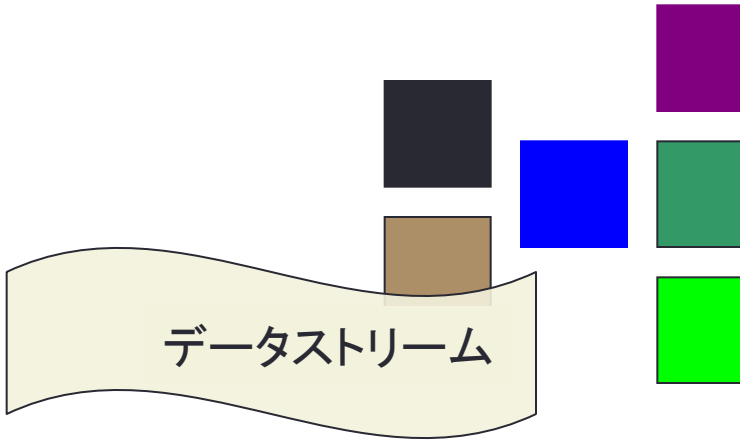
- 研究背景
  - データストリームマイニングとは？
  - 頻出アイテムのマイニングの簡単
    - Counting手法: Frequent, Lossy Counting, Space Saving, FPDM-1
    - Hashing手法： Count Sketch, Count-Min
    - Sampling: Sticky Sampling, Probabilistic - Inplace
  - 頻出アイテム集合のマイニング
- 研究内容
  - データのバースト出現をもつストリームのマイニング
  - 提案手法とその評価
- まとめと今後の課題

データストリームマイニングとは

- 常にデータを受け取りながら即座に頻出データを抽出する技術

オンライン型

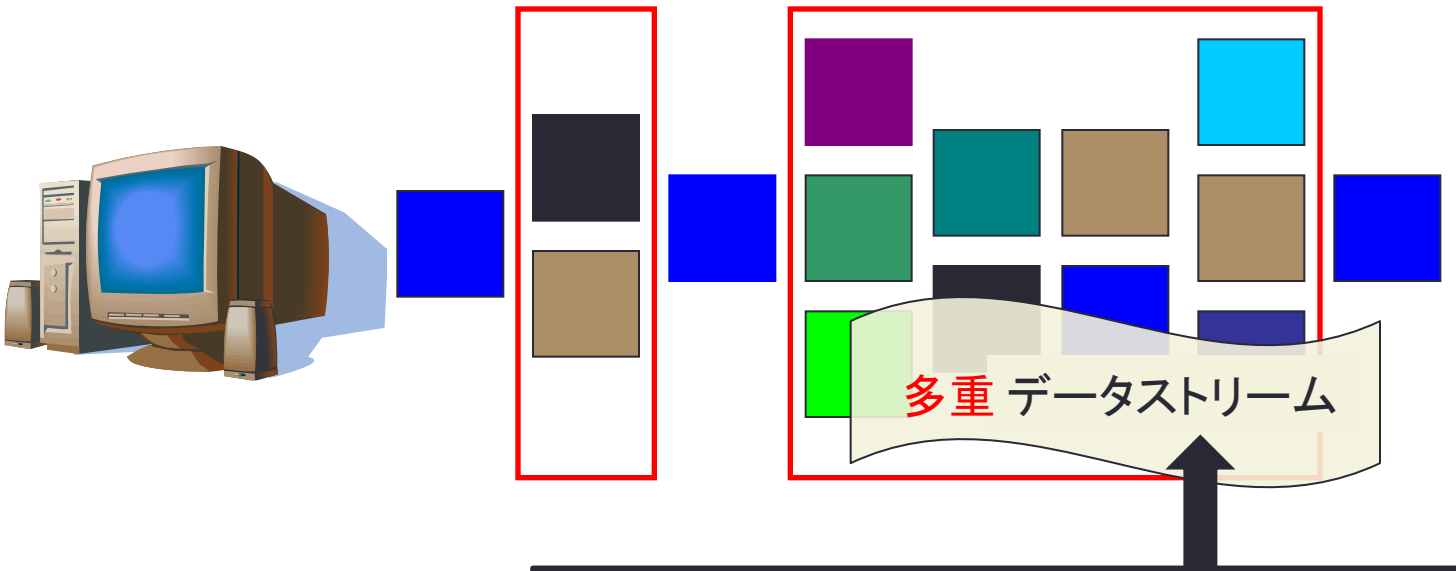




データストリームマイニングとは

- 常にデータを受け取りながら即座に頻出データを抽出する技術

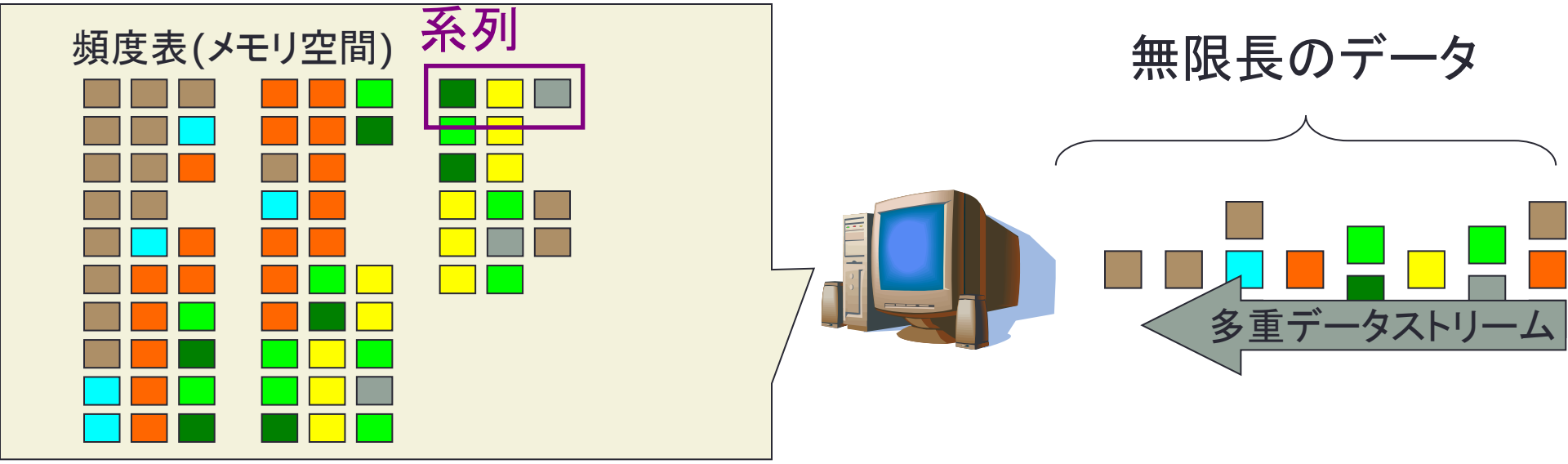
オンライン型



データストリームマイニングにおけるメモリ使用量の問題

- 無限長のデータを扱うことを仮定
  - 正確な計算を行うには原理的にメモリが足りない
  - メモリに残すデータを取捨選択し，メモリ節約を行うことが重要

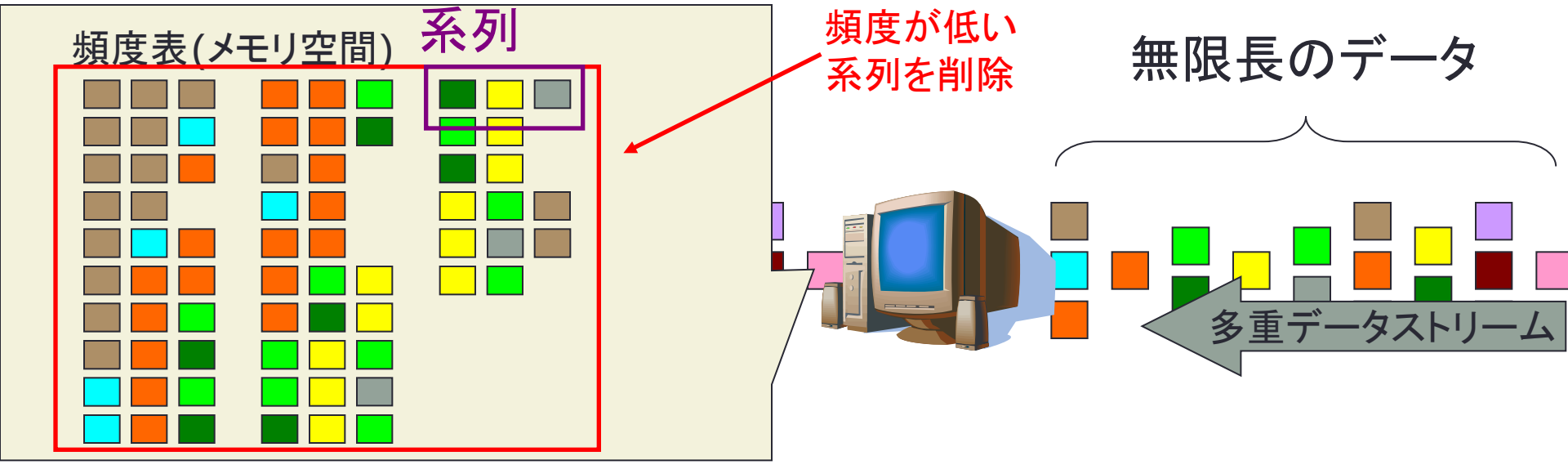
計算に**誤差**を許すことで，メモリ節約を行う



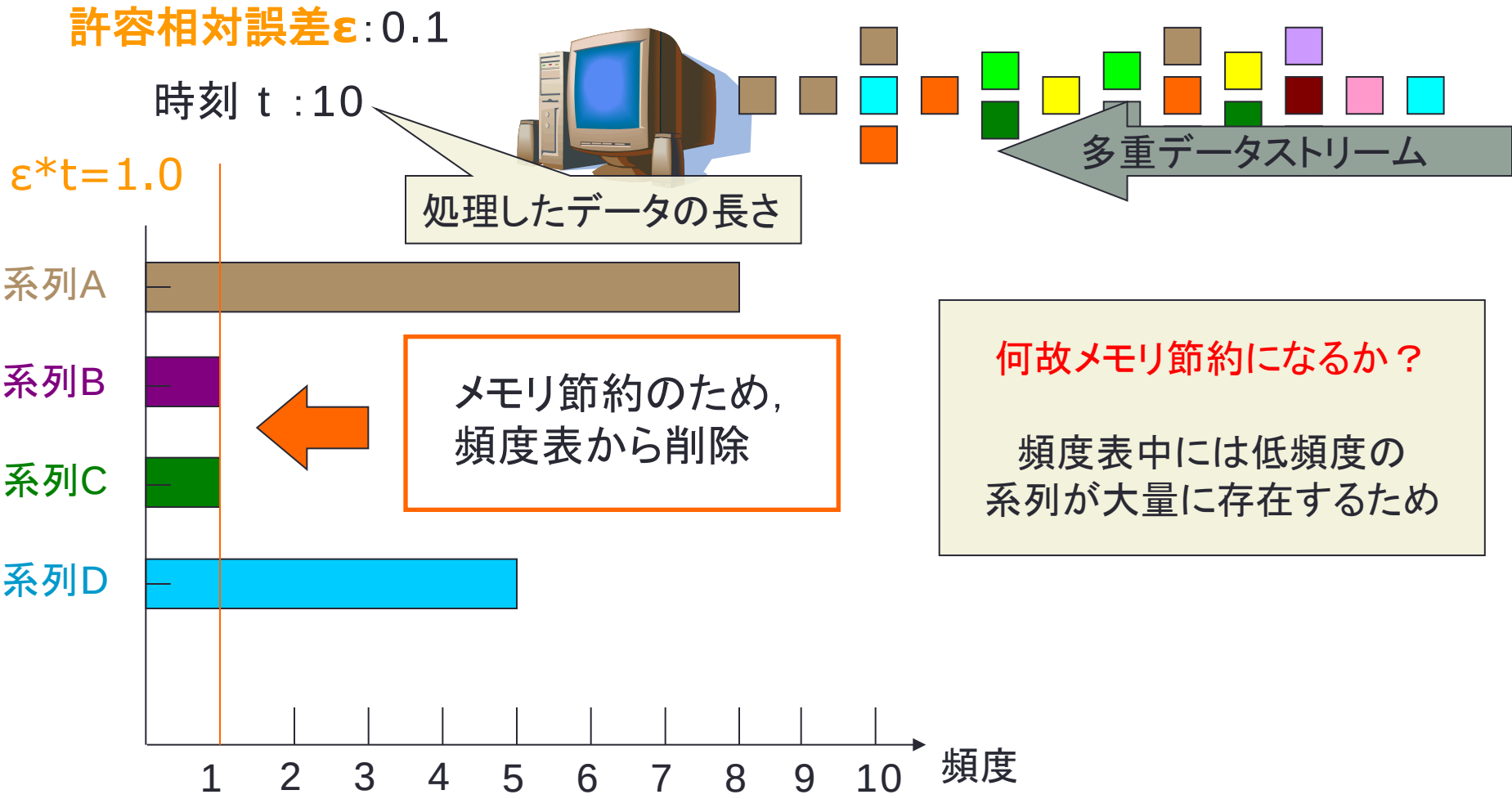
データストリームマイニングにおけるメモリ使用量の問題

- 無限長のデータを扱うことを仮定
  - 正確な計算を行うには原理的にメモリが足りない
  - メモリに残すデータを取捨選択し，メモリ節約を行うことが重要

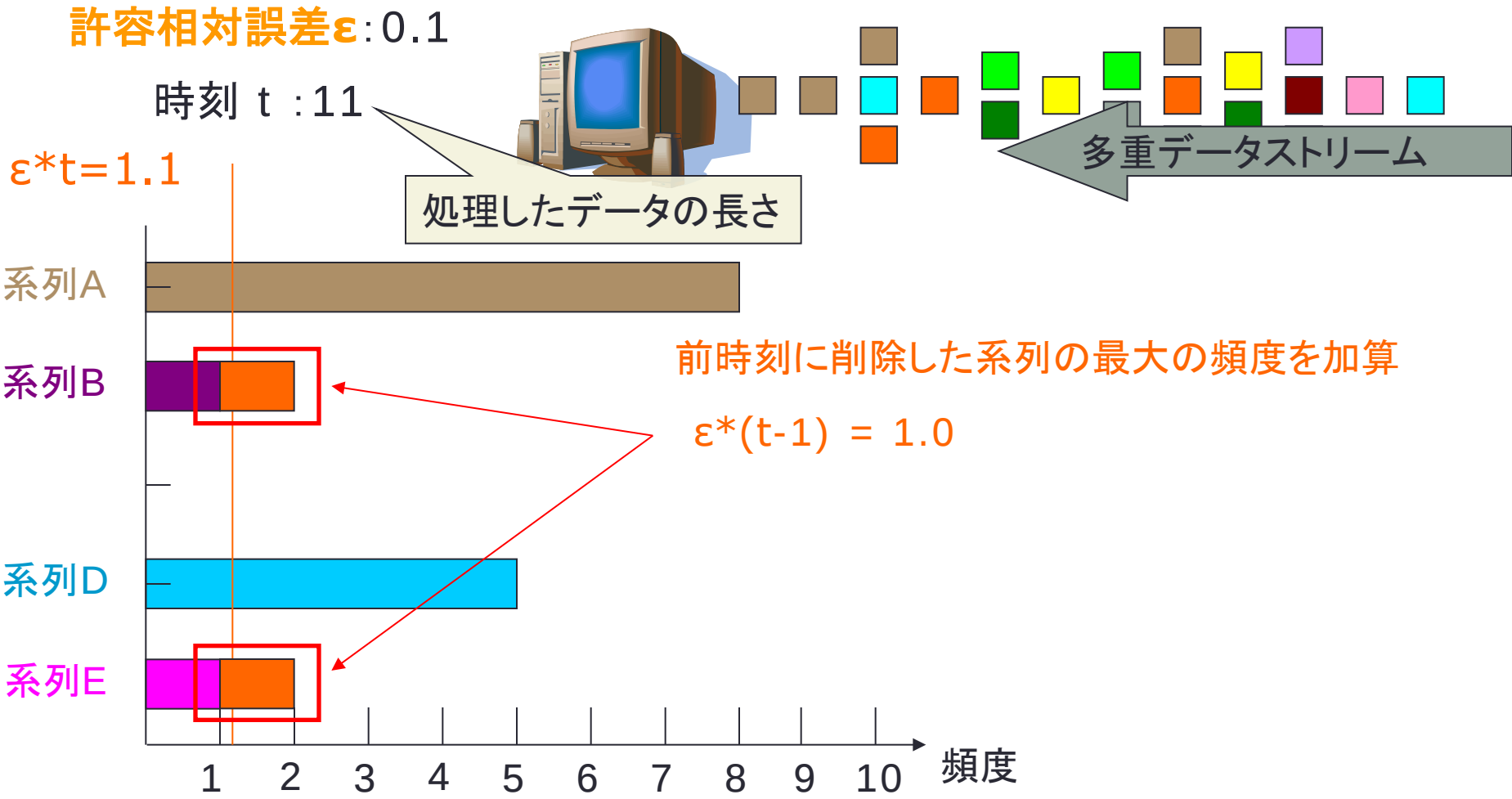
計算に**誤差**を許すことで，メモリ節約を行う



例　Lossy Counting法のメモリ節約手法



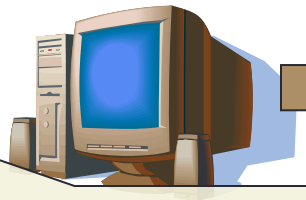
例　Lossy Counting法のメモリ節約手法



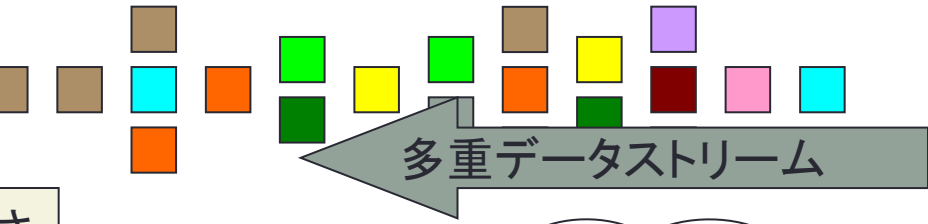
例　Lossy Counting法のメモリ節約手法

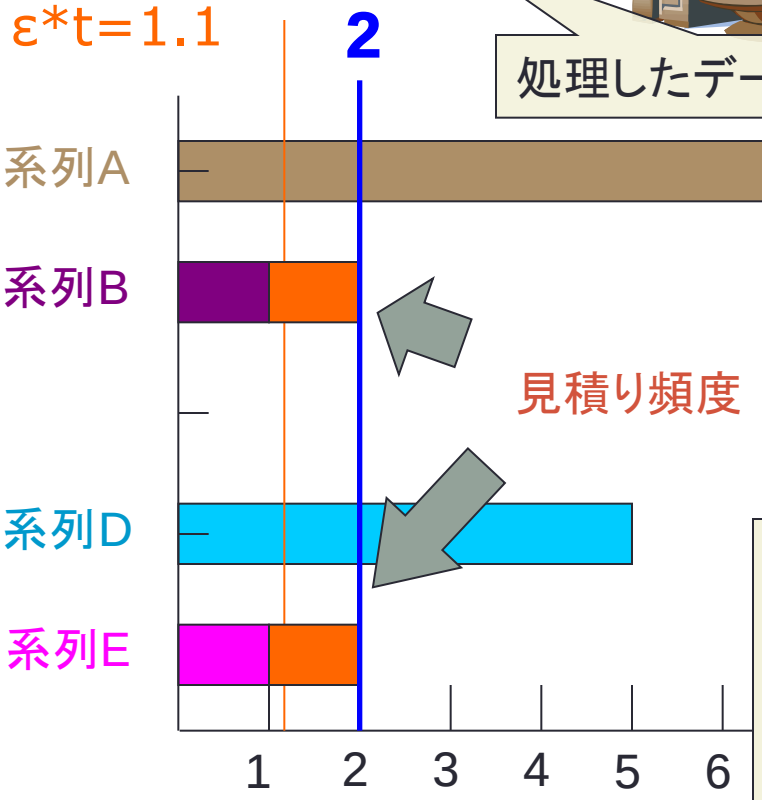
許容相対誤差ε:0.1

時刻 t :11



処理したデータの長さ





見積り頻度 ≧ 実頻度

頻度が2以上の系列
を抽出したい

系列Eは誤検出される
系列Bは正しく検出される

完全性　:全ての頻出系列を抽出できる
誤差保証:　頻度値の計数誤差は最大でも εt

メモリを節約しつつ，頻出系列の近似計算が可能

先行研究の簡単なサーベイ

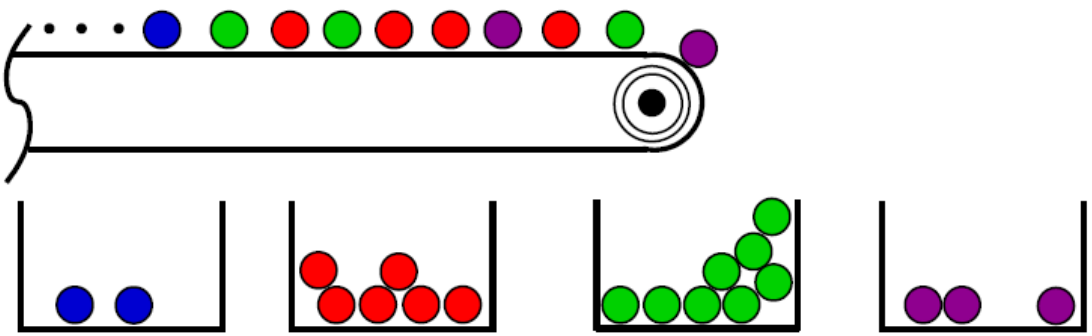
- **頻出アイテムのマイニング：** 非常によく研究され，各種の近似手法が体系化され，時間計算量，空間計算量その他もよく解析されている。

  - Counting手法: Frequent, Lossy Counting, Space Saving,
  - FPDM-1
  - Hashing手法： Count Sketch，Count-Min
  - Sampling: Sticky Sampling，Pobablistic – Inplace

- **頻出アイテム集合のマイニング：** 空間計算量などは自明な上界 $O(2^n)$ (但しnはアイテム種類数）以外のものは知られておらず，研究はあまり進んでいない。

データストリームからの頻出アイテム抽出

例 [Cormode et al. 2010]

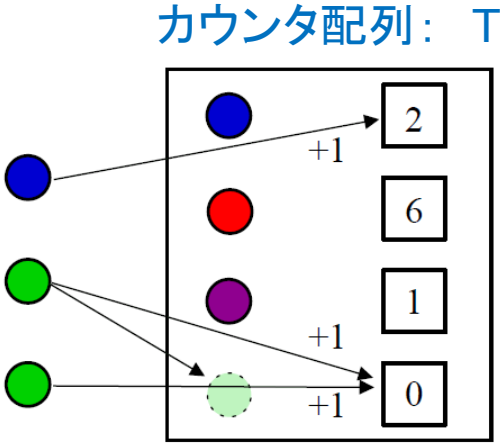


閾値を20%とすると，総計19アイテムなので，頻度3.8以上の緑と赤のアイテムが頻出となる。

上記の問題を正確に解くアルゴリズムの空間計算量の**下界はΩ(N)** (但し，Nはアイテム種類数)となる(これは集合のメンバシップ判定問題が上記問題へ帰着できることから簡単に導ける)。よって，全てのアイテムに対して(最終的に出現数の報告が必要なくても)出現数を数えるカウンタを用意する必要が生じる。

最大空間計算量をO(N)より下げるには，**カウント誤差**などを許容する必要がある。

Counting法その1：Frequent [Demaine et al. 2002, Karp et al.2003]



- もともとはtop-k アイテム (頻度 n/(k+1) 以上のアイテム)を抽出するもの
- 相対許容誤差をεとするとき，k= 1/εとすれば，頻出アイテム抽出が可能。
- Tに空きが無くなったら，T中のカウンタ全ての値を -1 するので，計数誤差が大きく出る傾向がある(in 実験的評価
- 抽出アイテムごとの個別の誤差は判別できない

Algorithm 1: FREQUENT(k)

```
1  n ← 0;
2  T ← ∅;
3  foreach i do
4      n ← n + 1;
5      if i ∈ T then
6          c_i ← c_i + 1;
7      else if |T| < k − 1 then
8          T ← T ∪ {i};
9          c_i ← 1;
10     else forall j ∈ T do
11         c_j ← c_j − 1;
12         if c_j = 0 then T ← T\{j};
```

図の出展 [Cormode et al. 2010]

Counting 法その2：
Lossy Counting [Manku et al. 2002]

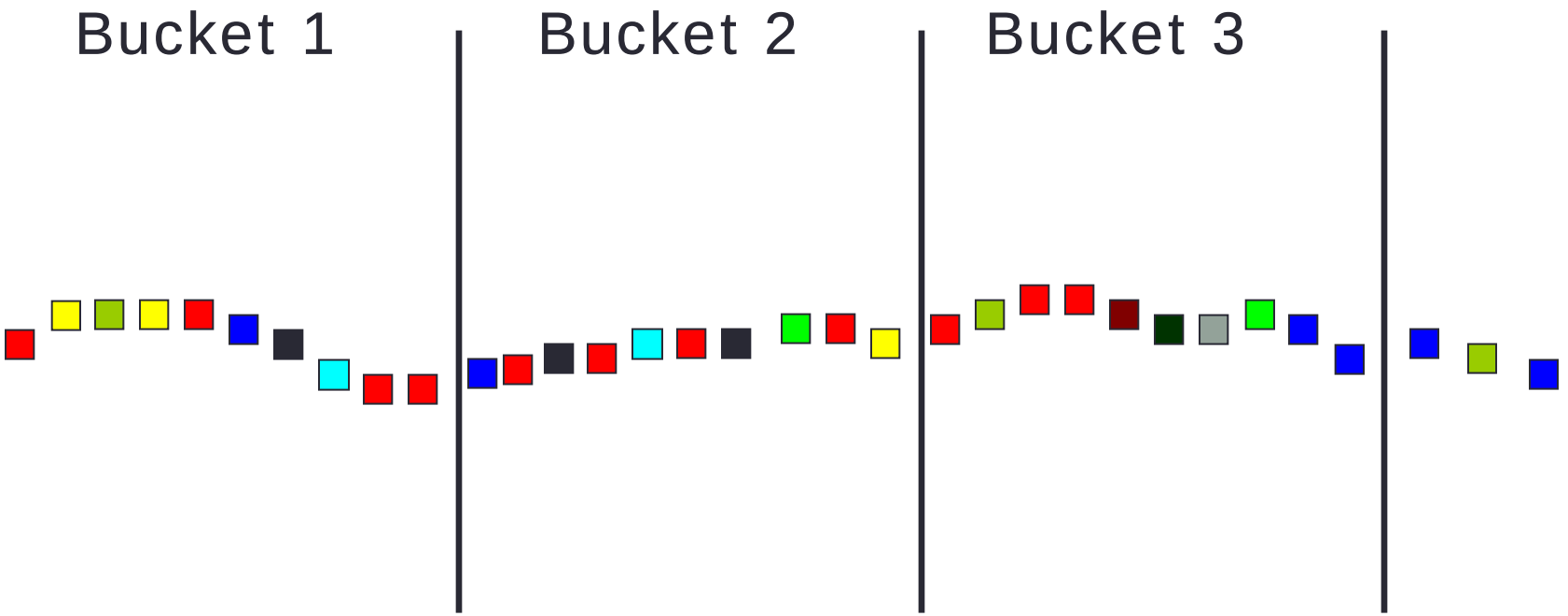
Algorithm 2: LOSSYCOUNTING(k)

```
1  n ← 0; Δ ← 0; T ← ∅;
2  foreach i do
3      n ← n + 1;
4      if i ∈ T then c_i ← c_i + 1;
5      else
6          T ← T ∪ {i};
7          c_j ← 1 + Δ;
8      if ⌊n/k⌋ ≠ Δ then
9          Δ ← ⌊n/k⌋;
10         forall j ∈ T do
11             if c_j < Δ then T ← T\{j};
```

図の出展 [Cormode et al. 2010]

- ⊿は計数誤差
- n/k 個のアイテムを受け取るごとに，積極的にTの圧縮を行う。これは単位時間あたりの処理量（＝スループット）の低下を招きがちである.
- 相対許容誤差をεとするとき，k= 1/ε
- 空間計算量は O((1/ε)･log n) であり，n の因子が残っている

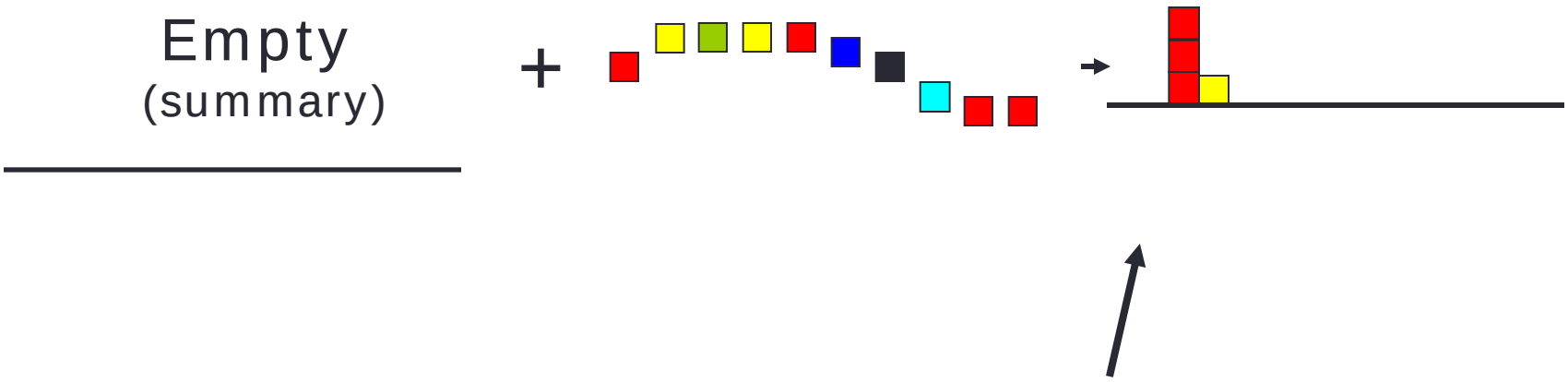
計数誤差の保証の仕組み [Han et al. 2006]



ストリームを Buckets へ分割
(bucket サイズは許容誤差の逆数 1/ ε = 1000)

計数誤差の保証の仕組み [Han et al. 2006]

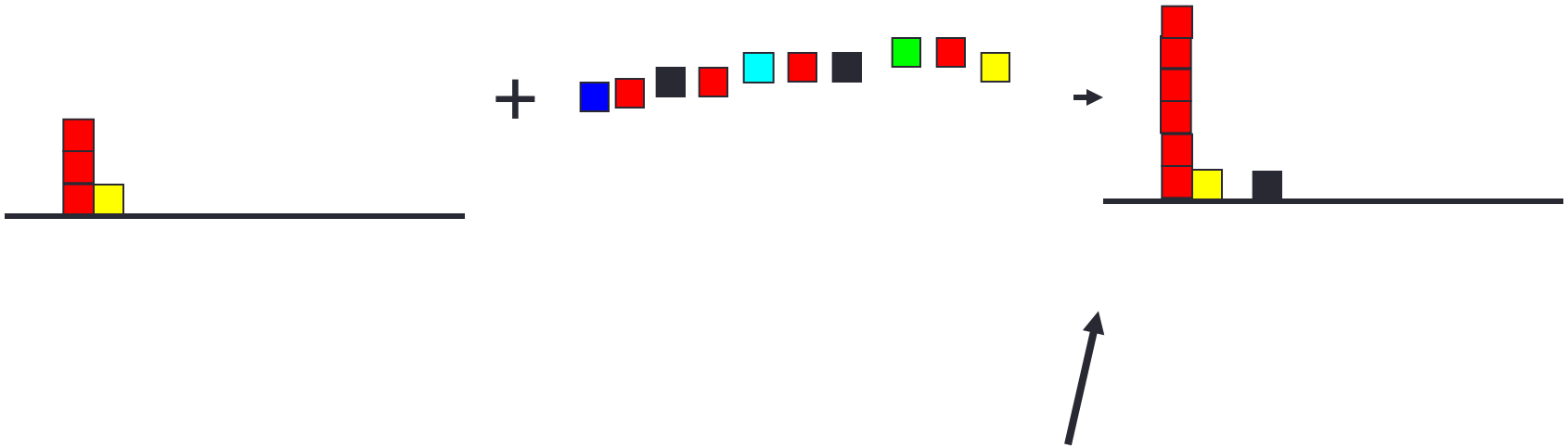
ー Lossy Counting と Frequent の混合形を例として ー



バケットの最後で，全てアイテムの頻度カウンタを -1 する．
カウンタが0になったアイテムはsummaryから削除する．

計数誤差の保証の仕組み [Han et al. 2006]

ー Lossy Counting と Frequent の混合形を例として ー



バケットの最後で，全てアイテムの頻度カウンタを -1 する．
カウンタが0になったアイテムはsummaryから削除する．

計数誤差の保証の仕組み [Han et al. 2006]

ー Lossy Counting と Frequent の混合形を例として ー

- 入力 (1) 最少相対サポート: σ, (2) 許容相対誤差: ε, (3) データストリーム長 N
- 出力: 計数頻度が(σ – ε) Nを超えるアイテム全て

- このときの計数誤差の見積もり?

  Bucketサイズが1/εで，データストリーム長さがNなので，バケット総数は εN しかない。よって各アイテムの計数誤差の最大値は **εN**

- 頻度近似の誤差保証
  - false negative なアイテムは無い （＝頻出アイテムは全て出力）
  - false positive なアイテムはあり得るが，その真の頻度は (σ–ε)N 以上
  - 各アイテムの計数頻度の誤差は最大で **εN**

Counting 法その3:
Space Saving [Metwally et al. 2005]

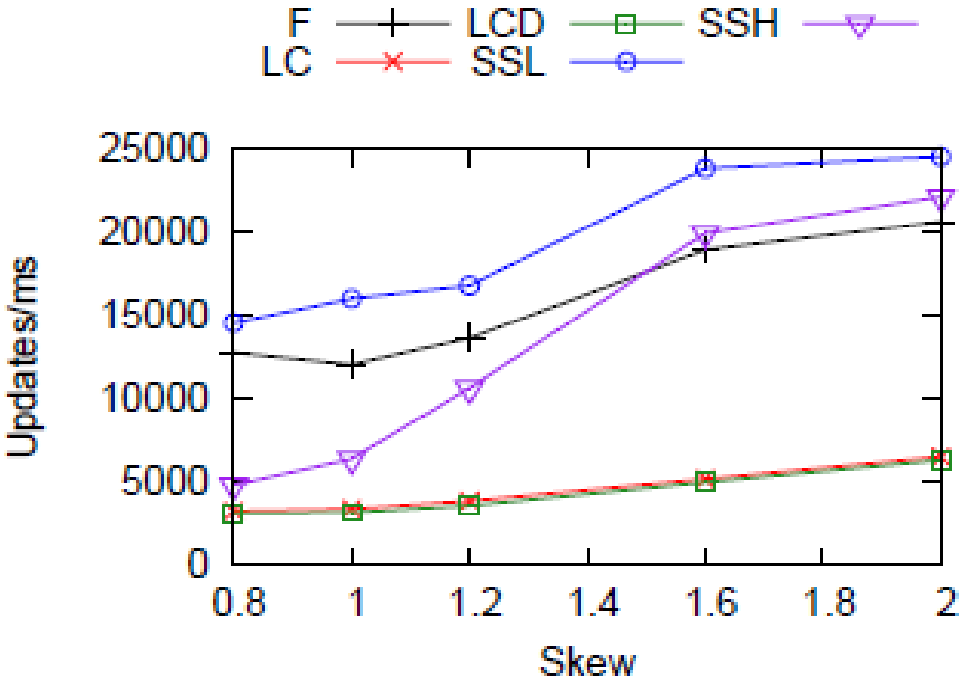
Algorithm 3: SPACESAVING(k)

```
1  n ← 0;
2  T ← ∅;
3  foreach i do
4      n ← n + 1;
5      if i ∈ T then c_i ← c_i + 1;
6      else if |T| < k then
7          T ← T ∪ {i};
8          c_i ← 1;
9      else
10         j ← arg min_{j∈T} c_j;
11         c_i ← c_j + 1;
12         T ← T ∪ {i} \ {j};
```

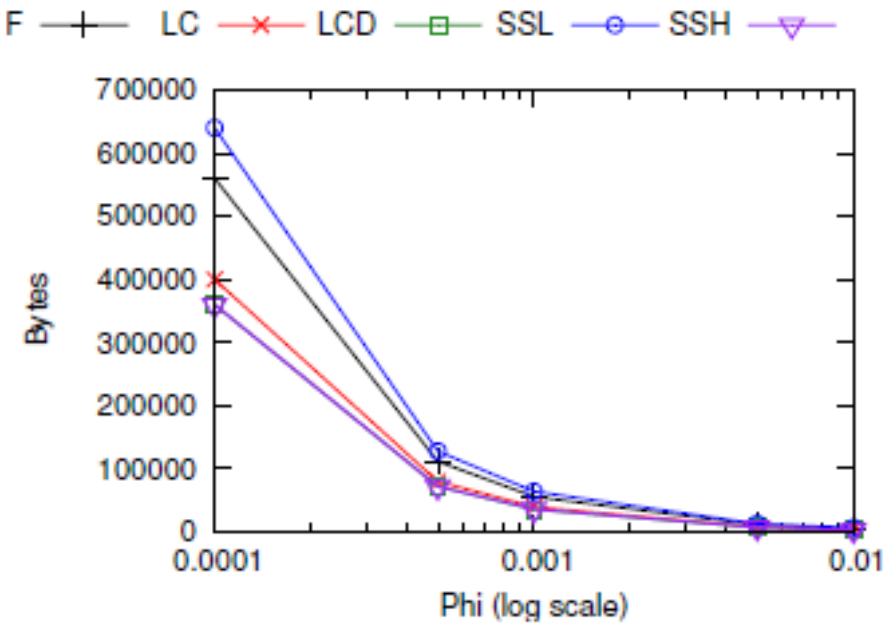
図の出展 [Cormode et al. 2010]

- カウンタ配列Tに空きが無くなったら，頻度最少のアイテムと入れ替える。スループットを上げられると同時に，廃棄するアイテムが最小限度になるので，計数誤差が少なくできる
- 削除されたアイテムjの頻度c_jが計数誤差になる
- 空間計算量は O(1/ε)

Counting 法の実験的評価の例 [Cormode et al. 2010]
ー スループットとメモリ使用量 ー

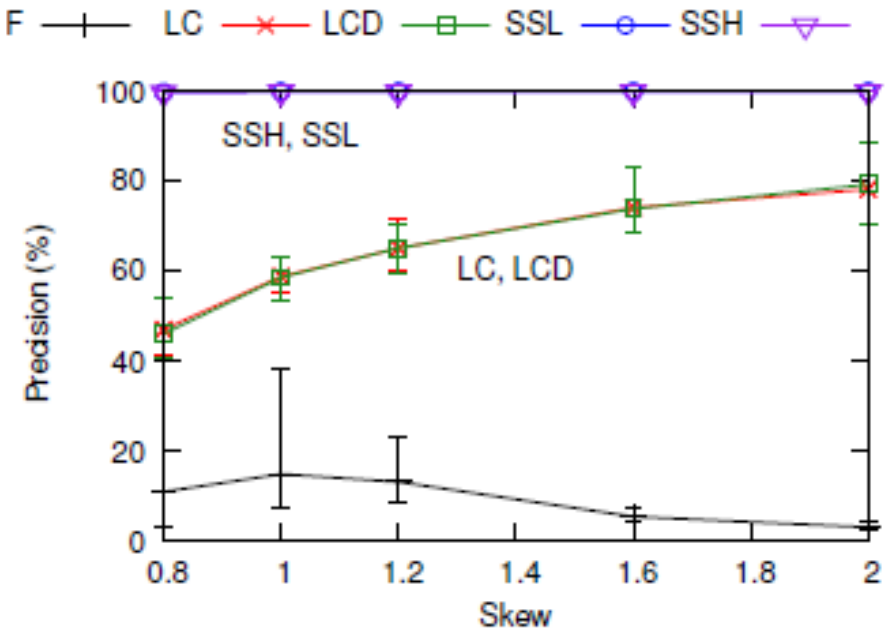


(a) Zipf: Speed vs. Skew.

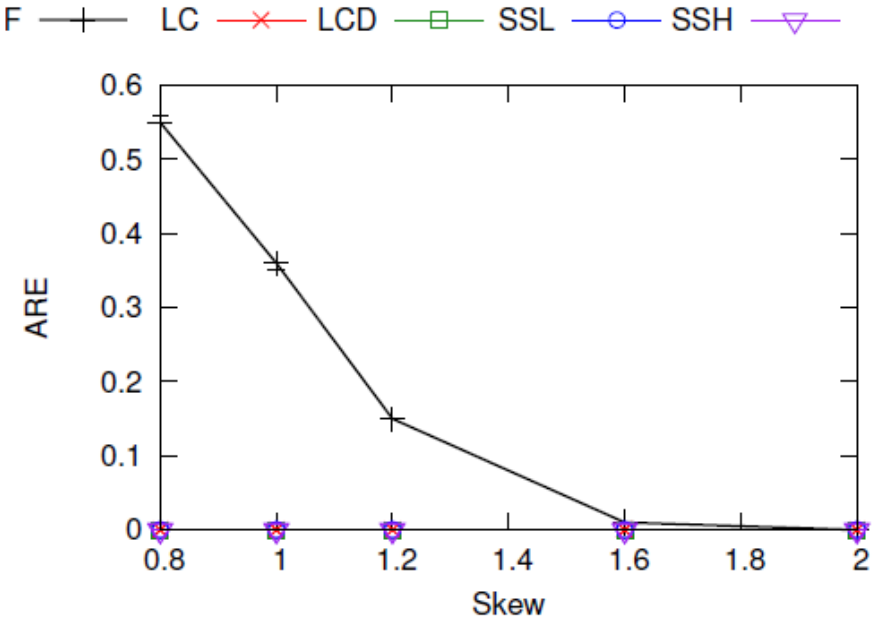


(c) Zipf: Size vs. ϕ.

Counting 法の実験的評価の例 [Cormode et al. 2010]
ー 正解率と平均誤差 ー



(f) Zipf: Precision vs. Skew.



(h) Zipf: ARE vs. Skew (frequent items).

Counting 手法その4： FPDM-1 [Yu et al. 2004]
ー アイテム出現の統計的独立性の仮定 ー

$\overline{sup}(X)$ ：アイテムXの現在までの出現数

アイテムX が *potentially infrequent* とは

$$\overline{sup}(X)/n < s - \epsilon_n$$

但し

$$\epsilon_n = \sqrt{\frac{2s\ln(2/\delta)}{n}}$$

注： ε_n は nが大きくなると 0 に収束

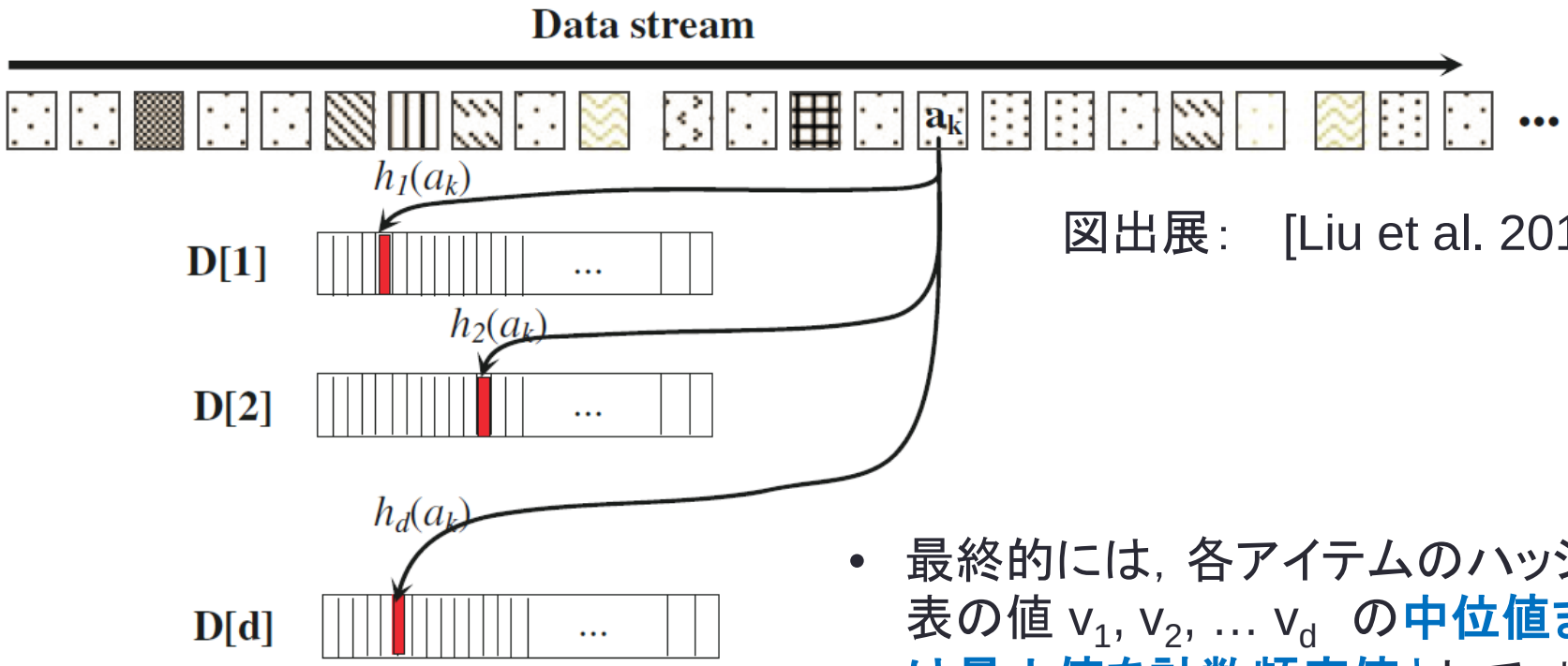
- 決定性アルゴリズムの確率的性能保証
- 特徴：No false positiveアルゴリズム
- 実頻度 εn のアイテムを確率 1- δ以上で出力
- 空間計算量は

Eq.(3): $n_0 = \frac{2 + 2\ln(2/\delta)}{s}$

Algorithm 1 FDPM-1(s, δ)

```
1:  let n_0 be the required number of observations (Eq. (3));
2:  n ← 0, P ← ∅;
3:  while a new transaction t arrives do
4:    if t ∈ P then
5:      increase t's count by 1;
6:    else
7:      if |P| > n_0 then
8:        calculate the running ε_n for the n observations;
9:        delete all entries in P that are potentially infrequent;
10:     end if
11:     insert t with an initial count 1 into P;
12:   end if
13:   n ← n + 1;
14:   output P on demand;
15: end while
```

Hashに基づくストリームマイニング：
計測頻度の確率的な誤差保証



図出展：　[Liu et al. 2011]

- ハッシュの数d は信頼性に影響し，各ハッシュ表の大きさは頻度の計数誤差に影響を及ぼす．

- 最終的には，各アイテムのハッシュ表の値 $v_1, v_2, \ldots v_d$ の**中位値または最小値を計数頻度値**として，頻出か否かの判断に用いる．
- アイテムに負の重み値をつけても，頻度値の産出が可能

頻出アイテムのストリームマイニング一覧表 [Liu et al. 2011]

Category	Name	Stream model	Query	Space bound	Update time (per item)	Guarantee
Sampling	*Concise Samples* [33]	*Cash Register*	top-k	No clear bound	$O(1)$ amortized	(−, probabilistic)
	Counting Samples [33]	*strict Turnstile*	top-k	No clear bound	$O(1)$ amortized[a]	(+, probabilistic)
	Sticky Sampling [56]	*strict Turnstile*	iceberg	$O\left(\frac{1}{\varepsilon}\log\left(\frac{1}{\varphi\delta}\right)\right)$	$O(1)$ amortized	$(-\varepsilon n, \delta)$
	Probabilistic- Inplace [25]	*Cash Register*	top-k	$O(s)$[b]	$O(1)$[c]	(−, probabilistic)
Counting	*Frequent* [25]	*Cash Register*	top-k	$\Theta(k)$	$O(1)$	$(-n/(k+1), 0)$[d]
	Lossy Counting [56]	*Cash Register*	iceberg	$O\left(\frac{1}{\varepsilon}\log(\varepsilon n)\right)$	$O(\log(\varepsilon n))$ amortized[e]	$(+\varepsilon n, 0)$
	Space Saving [57]	*Cash Register*	iceberg	$O\left(\frac{1}{\varepsilon}\right)$	$O(1)$	$(+\varepsilon n, 0)$
			top-k	$O(\frac{n}{\varepsilon F_k})$	$O(1)$	$(+\varepsilon F_k, 0)$
	FPDM-1 [68]	*Cash Register*	iceberg	$O\left(\frac{2+2\log(2/\delta)}{\varphi}\right)$	$O(1)$ amortized	$(\pm 0, \delta)$[f]
Hashing	*CountSketch* [13]	*Cash Register*	top-k	$O\left(\frac{k}{\varepsilon^2}\log\frac{n}{\delta}+k\right)^g$	$O(\log(\frac{n}{\delta}))$	$(+\varepsilon F_k, 0)$
	GroupTesting [18]	*strict Turnstile*	top-k	$O(k\log\left(\frac{k}{\delta}\right)\cdot\log m)$	$O(\log(\frac{k}{\delta})\cdot\log m)$	$(\pm, \delta)$[h]
	Count-Min [19]	*Cash Register*	iceberg	$O\left(\frac{1}{\varepsilon}\log\left(\frac{n}{\delta}\right)\right)$	$O\left(\log\left(\frac{n}{\delta}\right)\right)$	$(+\varepsilon, \delta)$
		strict Turnstile	iceberg	$O\left(\frac{1}{\varepsilon}\log m\cdot\log\left(\frac{2\log m}{\varphi\delta}\right)\right)$	$O\left(\log m\cdot\log\left(\frac{2\log m}{\varphi\delta}\right)\right)$	$(+\varepsilon, \delta)$[i]
	hCount [42][j]	*strict Turnstile*	iceberg	$O\left(\frac{1}{\varepsilon}\log\left(\frac{-m}{\log\delta}\right)\right)$	$O\left(\log\left(\frac{-m}{\log\delta}\right)\right)$	$(+\varepsilon, \delta)$

参考文献

[Cormode et al. 2010] G. Cormode and M. Hadjieleftherion: Methods for finding frequent items in data streams, *VLDB Journal*, 19(1), pp.3-20

[Demaine et al. 2002] ED. Demaine, A. L'opes-Oritz and JI. Munro: Frequency estimation of Internet packet streams with limited space, *Proc. of 10th European symp. On algorithms (ESA 2002)*, pp348-360

[Karp et al. 2003] R. Karp, C. Papadimitriou and s. Shenker: A simple algorithm for finding frequent elements in sets and bags, *ACM trans. on database syst.* 28(1), pp.51-55

[Liu et al. 2011] H. Liu, Y. Lin and J. Han: Methods for mining frequent items in data streams: an overview, *Knowl Inf Syst*, 26, pp.1-30

[Han et al. 2006] J. Han and M. Kamber: *Data mining: Concepts and Techniques* (Morgan Kaufmann 2006)

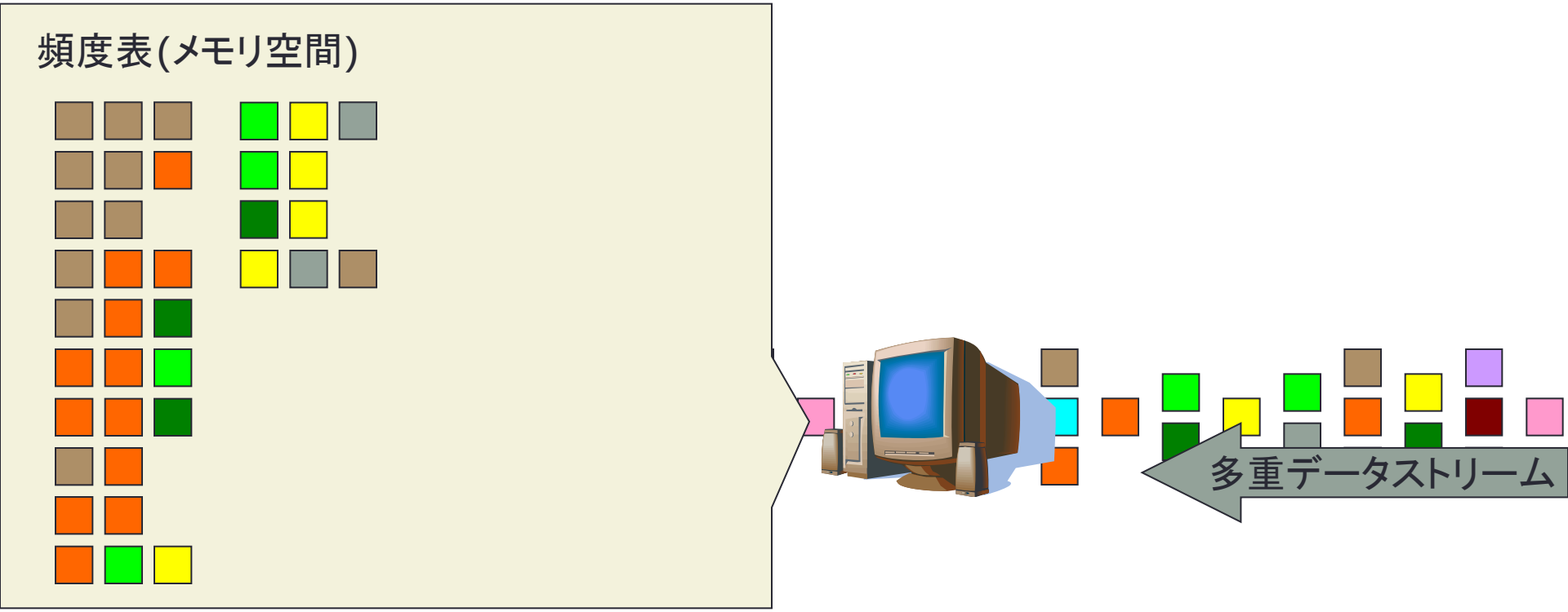
[Manku et al. 2002] G. Manku and R. Motwani: Approximate frequency counts over data streams, *Proc. of 31st Int'l Conf. on Very Large Databases (VLDB'05)* , pp.346-357

[Metwally et al. 2005] A. Metwally, D. Agrawal and AE. Abbadi: Efficient computation of frequent and Top-k elements in data streams, *Proc. of 10th Int'l Conf. on Database Theory (ICDT'05)*, pp.398-412

[Yu et al. 2004] JX. Yu, ZH. Chong, HJ. Lu and AY. Zhou: False positive or false negative: mining frequent itemsets from high speed transactional data streams, *Proc. of 30th Int'l Conf. on Very Large Databases (VLDB'04)* , pp.204-215

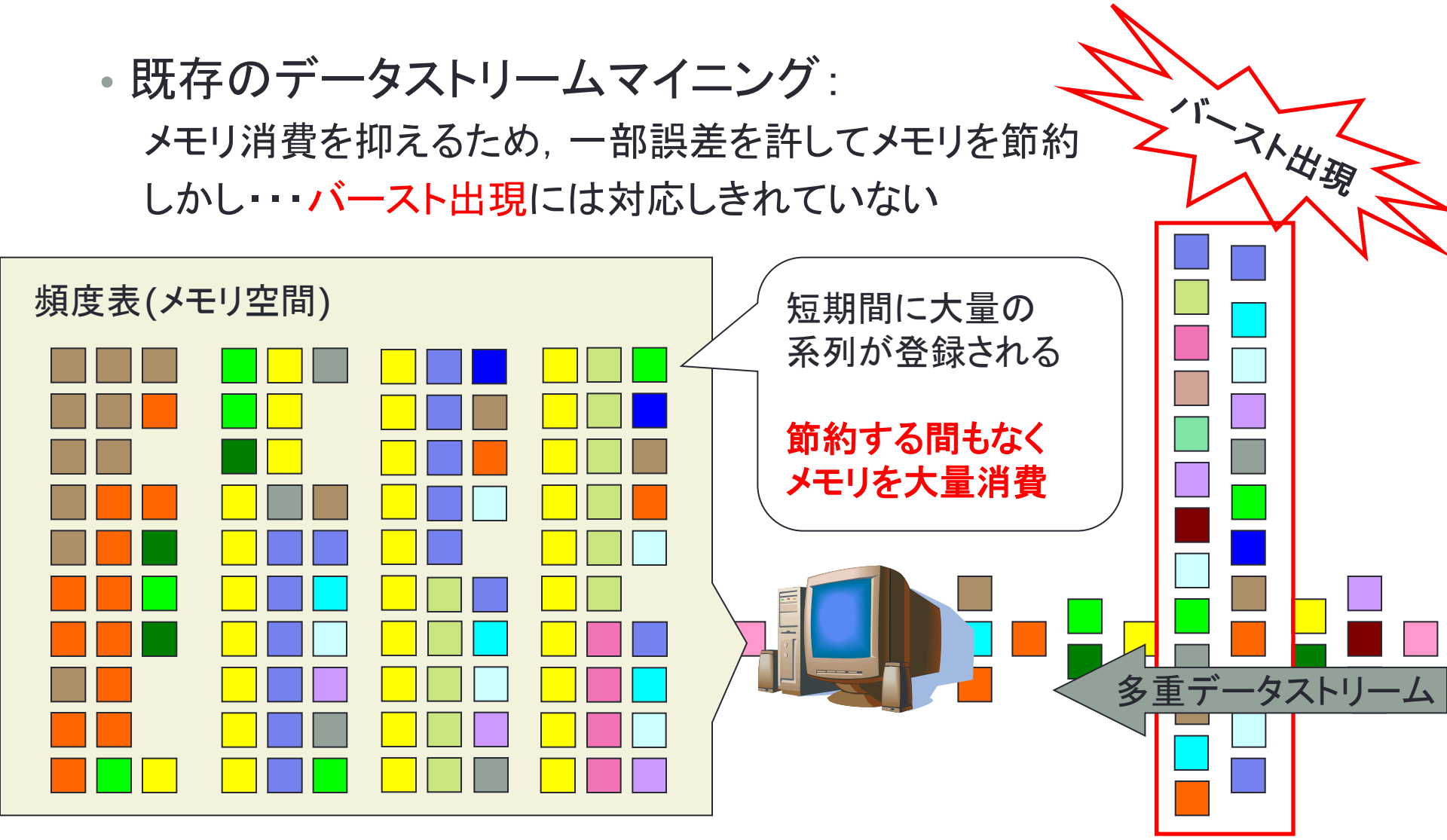
バースト出現を伴うデータストリーム

- 既存のデータストリームマイニング：
  メモリ消費を抑えるため，一部誤差を許してメモリを節約
  しかし・・・バースト出現には対応しきれていない



バースト出現を伴うデータストリーム

- 既存のデータストリームマイニング：
  メモリ消費を抑えるため，一部誤差を許してメモリを節約
  しかし・・・バースト出現には対応しきれていない



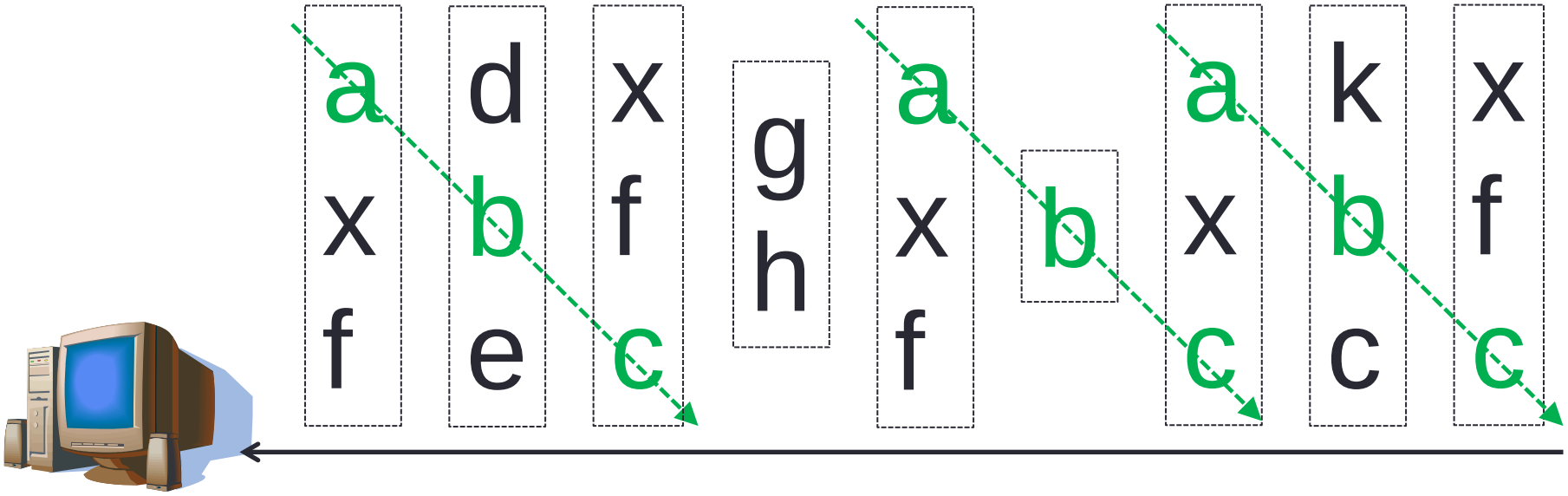
本研究

データの突発的な大量出現に対応可能な
データストリームマイニング

- メモリ使用量を一定値に固定したマイニング手法を提案
- Space Saving と Lossy Counting の混合形
- FPDM-1 に似た infrequent データの棄却

頻出部分系列マイニング

トランザクション型のデータストリーム中に頻出する部分系列のマイニング問題に取り組む。



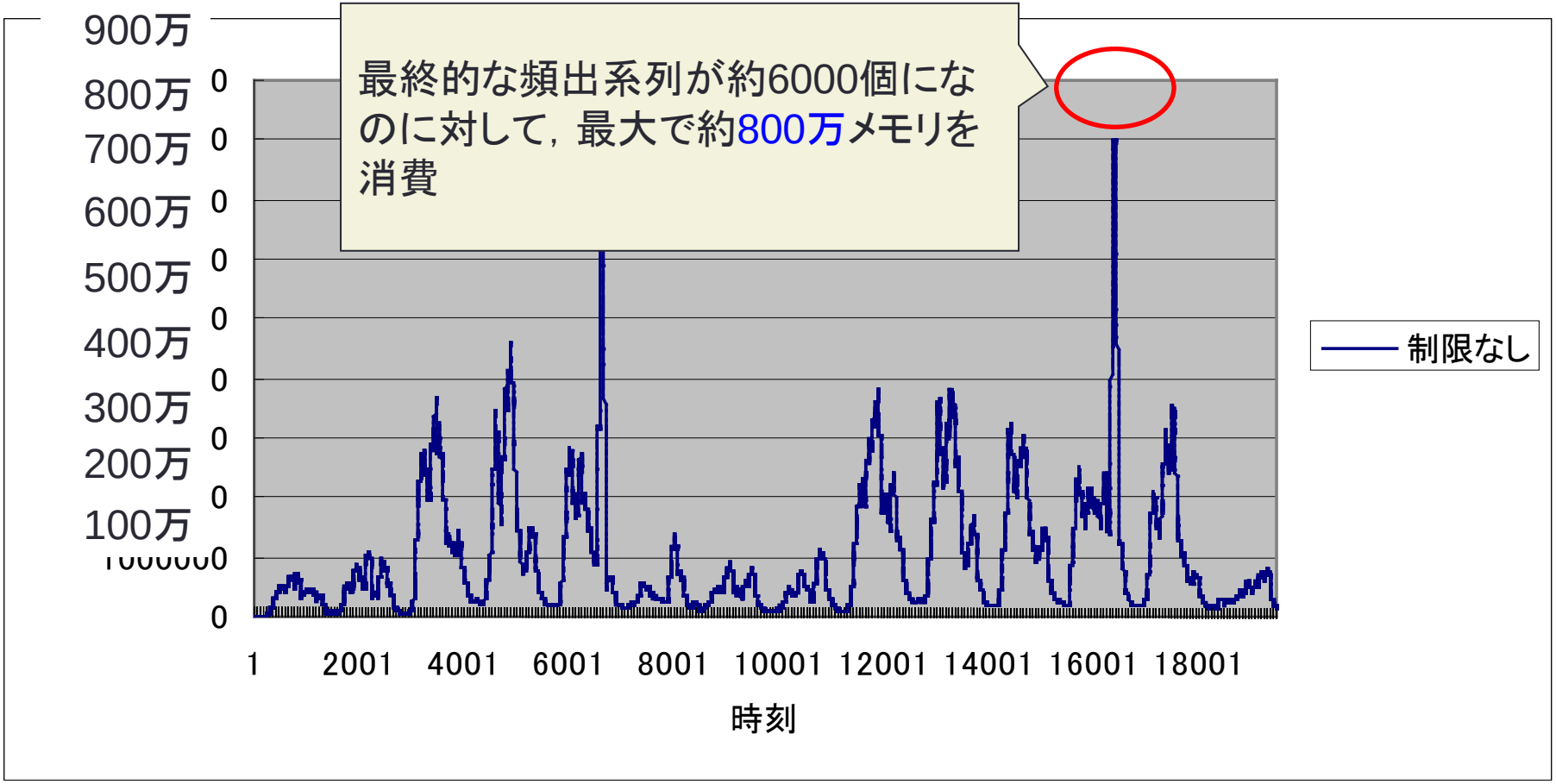
注意： 部分系列の出現の数え方はいろいろと考えられる。ここでは省略する。
詳細については以下を参照のこと

岩沼宏治： テキスト系列マイニングにおける有用性尺度について
人工知能学会誌，Vol.27, No.2 pp.136-145, 2012

Lossy Counting法による頻度表サイズの時間推移

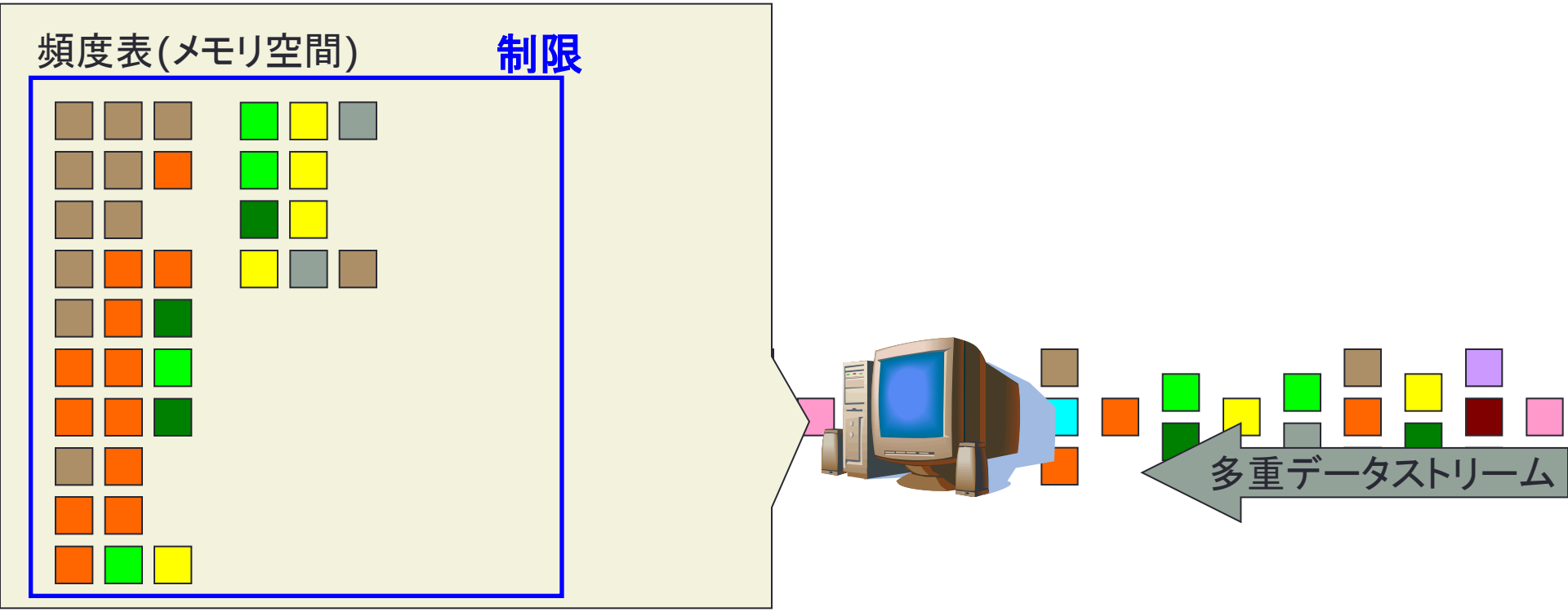
山梨大学HPの2週間のアクセス
データ長:19,466基本単位

部分系列の最大長:3,
最小相対頻度:0.1，許容誤差:0.01



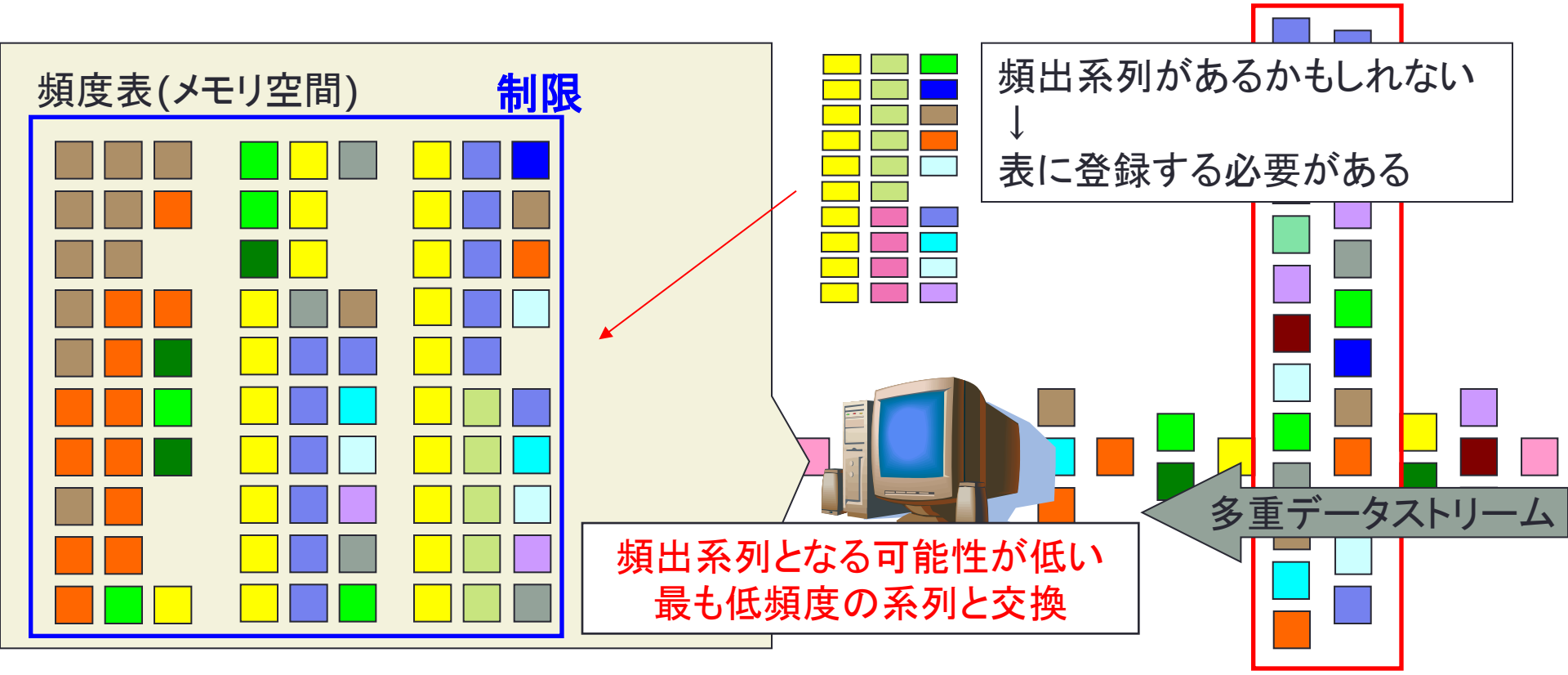
メモリ制限を導入した多重データストリームのマイニング

- バーストを扱う上で，システム停止を防ぐためメモリ使用を制限せざるをえない



メモリ制限を導入した多重データストリームのマイニング

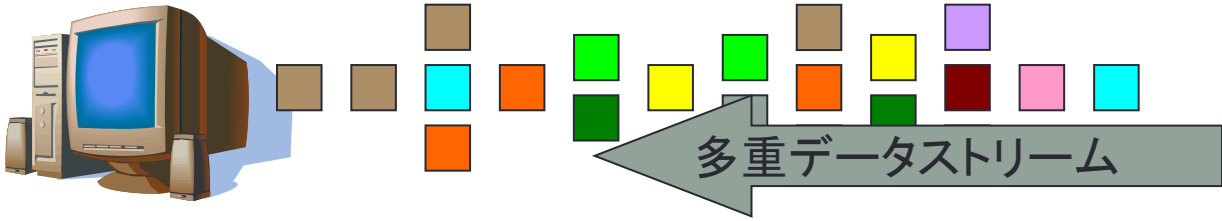
- バーストを扱う上で，システム停止を防ぐためメモリ使用を制限せざるをえない
- 対策：制限により登録できない系列は表内の系列と交換する



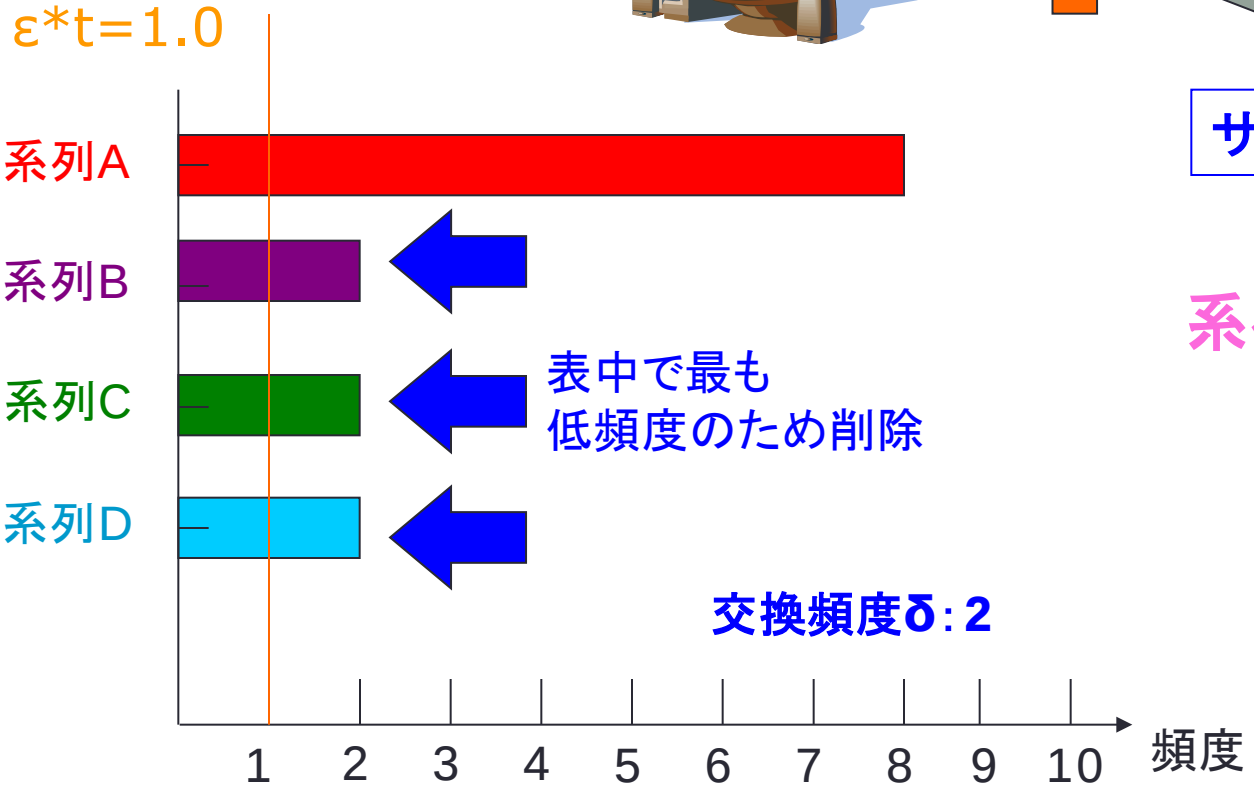
交換の仕組み

許容誤差ε：0.1

時刻 t ：10



サイズ制限：4

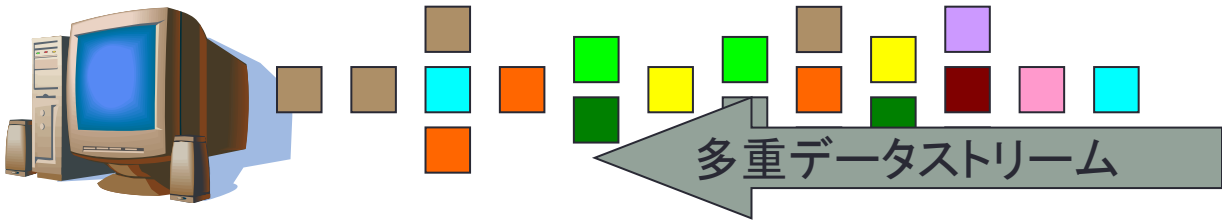


系列E

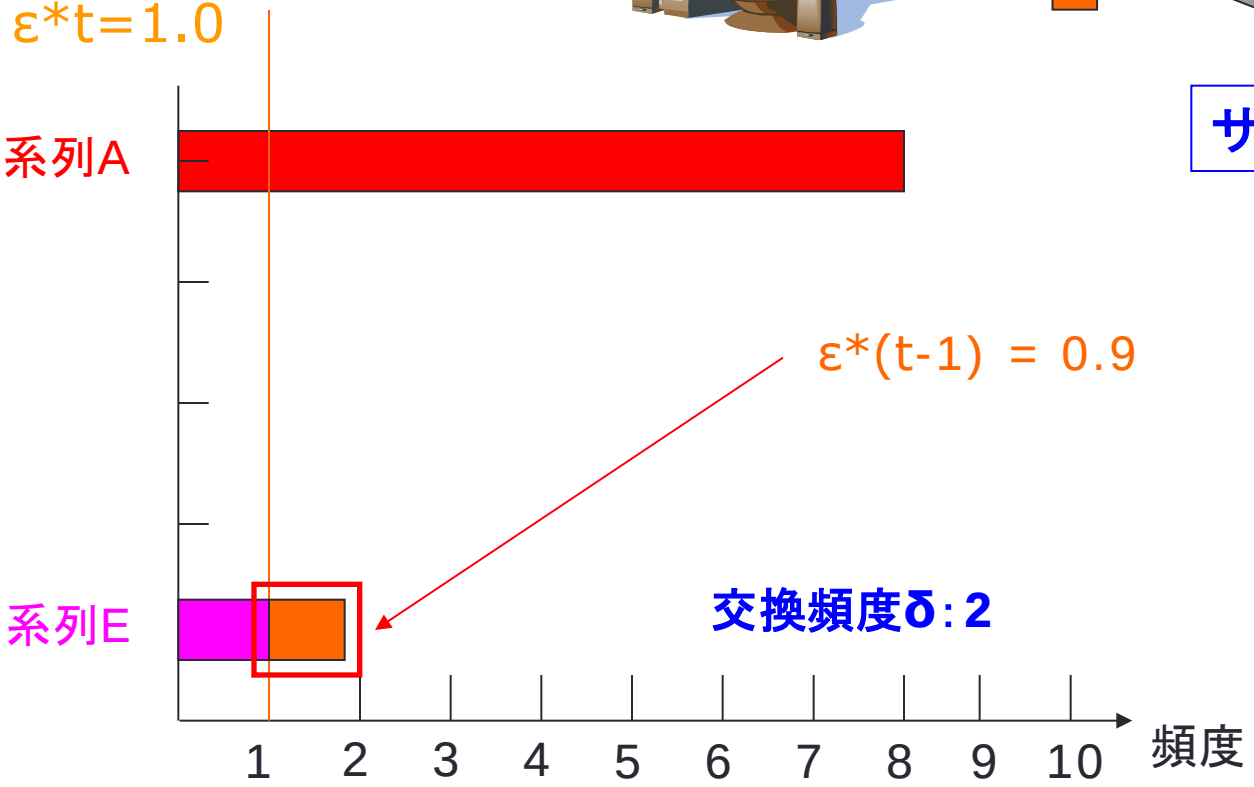
交換の仕組み

許容誤差ε：0.1

時刻 t ：10



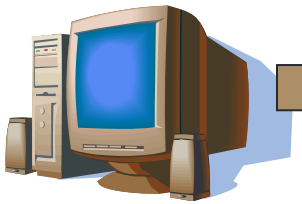
サイズ制限：4

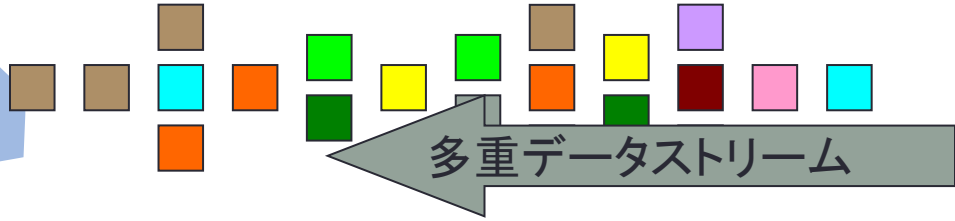


交換の仕組み

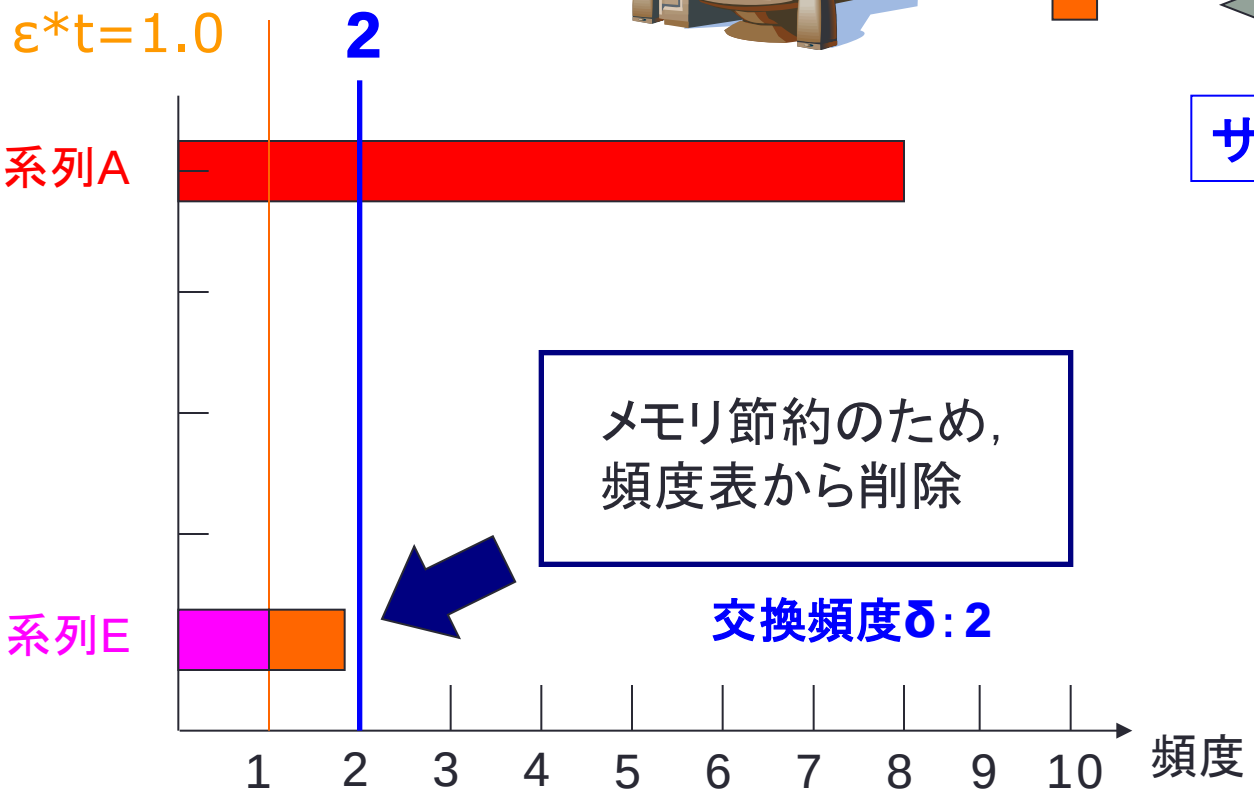
許容誤差ε：0.1

時刻 t ：10





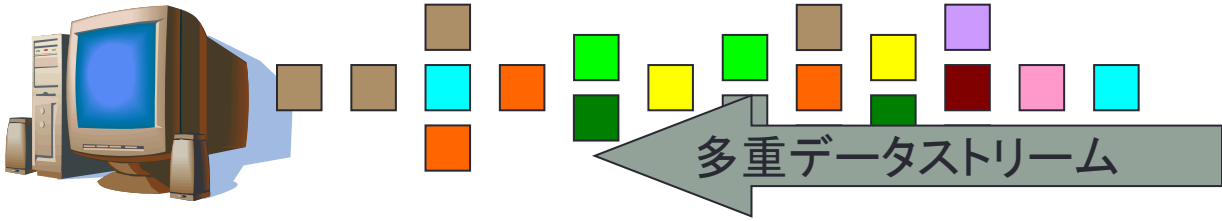
サイズ制限：4



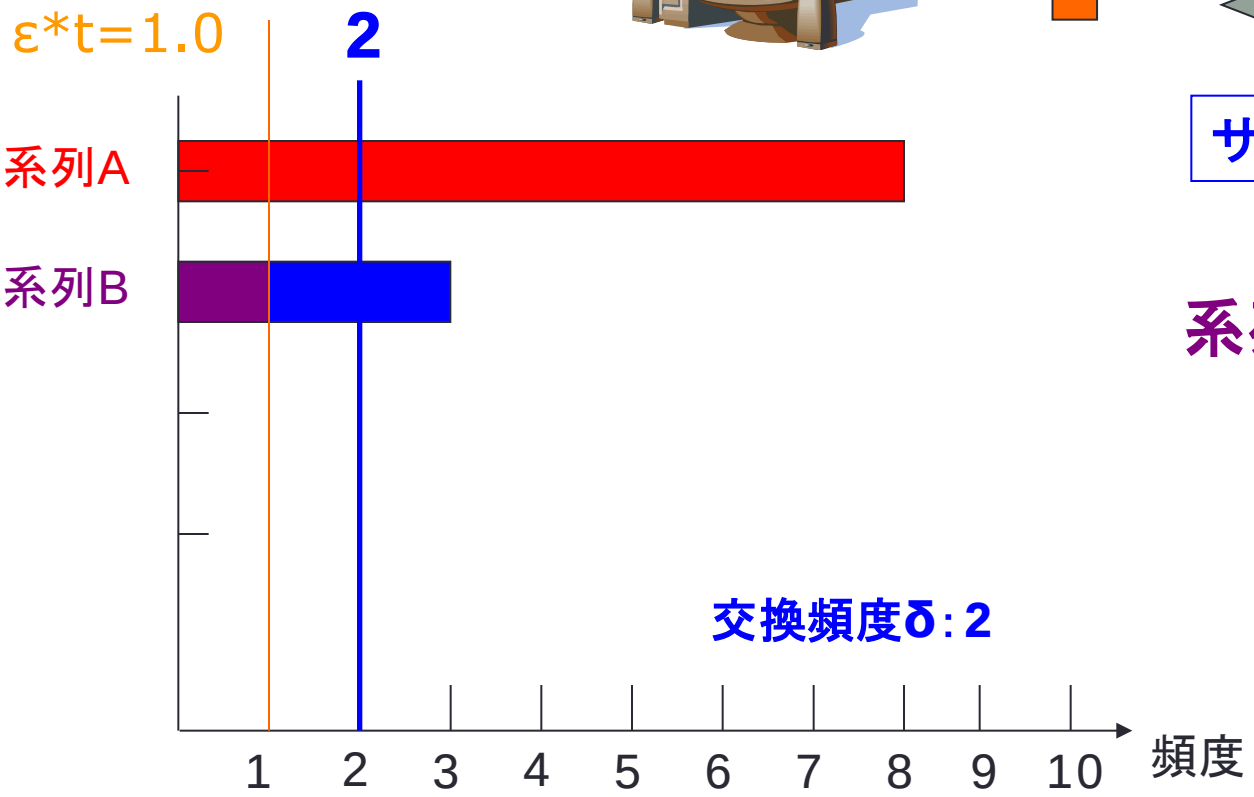
交換の仕組み

許容誤差ε：0.1

時刻 t ：10



サイズ制限：4

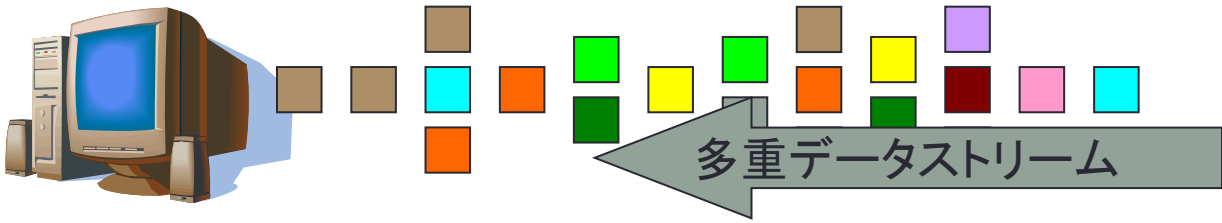


系列B

交換の仕組み

許容誤差ε:0.1

時刻 t :10



サイズ制限:4

ε*t=1.0

系列A

系列B

max(ε(t-1), δ) ＝2

登録までに最大で
どれだけ出現していたかを考える

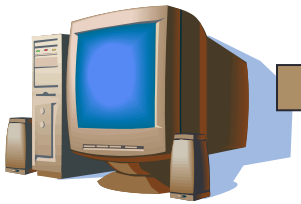
交換頻度δ:2

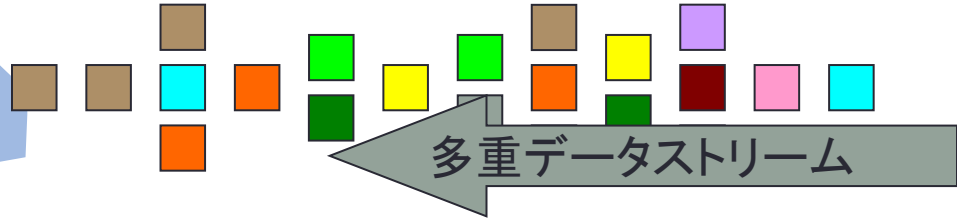
1 2 3 4 5 6 7 8 9 10 頻度

抽出結果の完全性保証

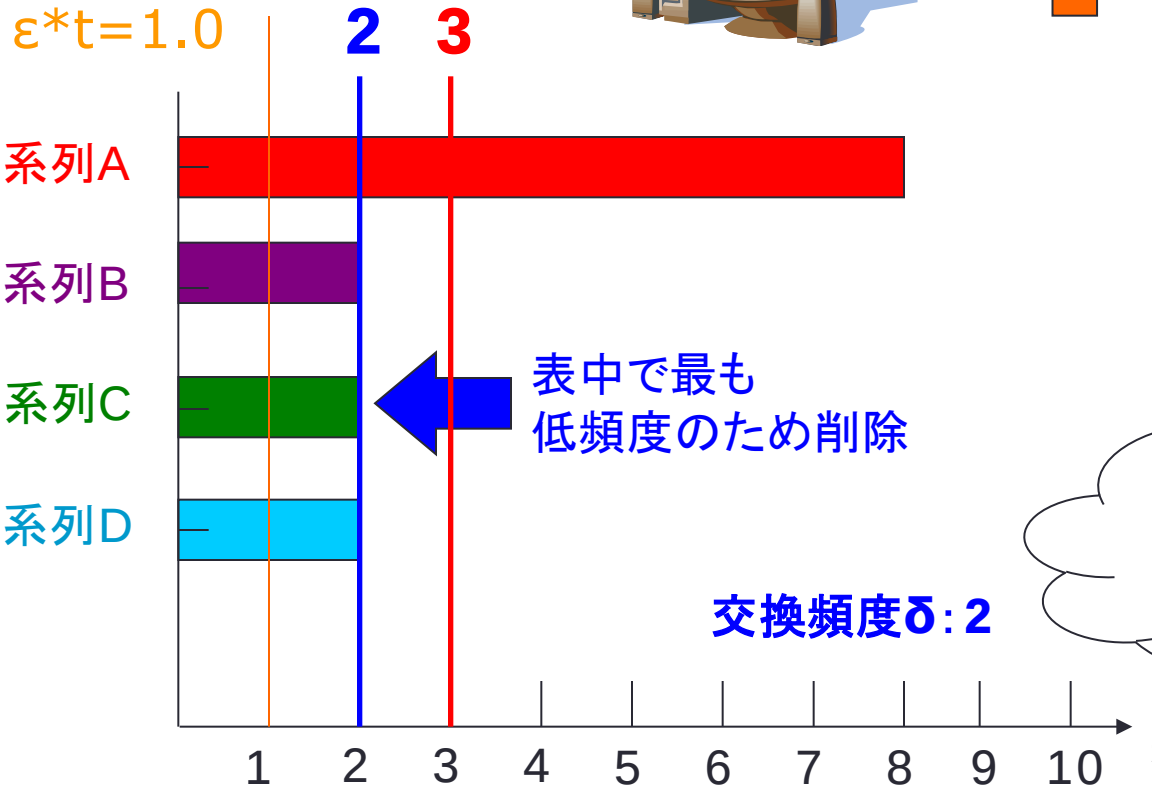
許容誤差ε：0.1

時刻 t ：10





サイズ制限：4



最小相対頻度：σ=0.3

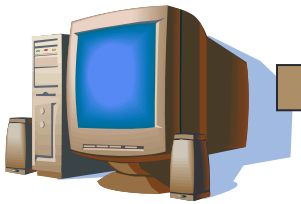
系列E

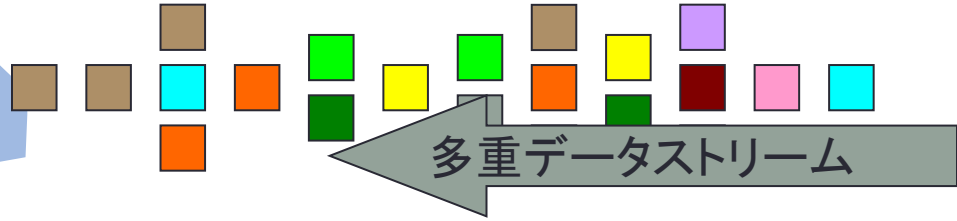
今までの頻度が3割以上
の系列を抽出したい

抽出結果の完全性保証

許容誤差ε：0.1

時刻 t ：10





サイズ制限：4

ε*t=1.0

2 3

系列A

抽出したい頻度 ≧ 交換頻度
ならば，頻出系列は表から削除されない

完全性(全ての頻出系列を抽出できること)が保証できる

最小相対頻度：σ=0.3

系列E

今までの頻度が3割以上の系列を抽出したい

交換頻度δ：2

1 2 3 4 5 6 7 8 9 10 頻度

性能評価実験1

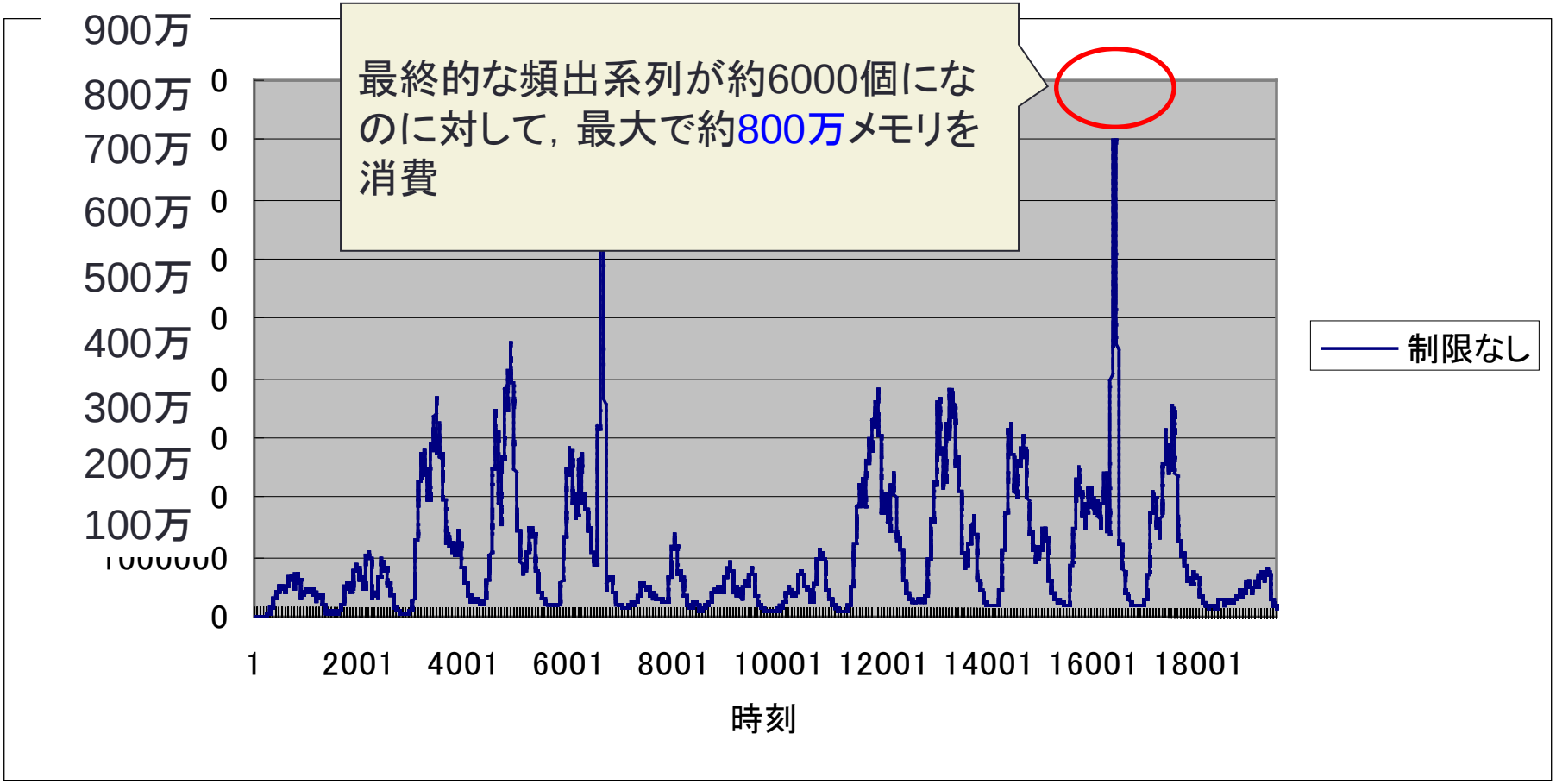
- パラメータ：
  - 最小相対頻度σ：0.1，許容誤差ε:0.01，ウィンドウ幅w:3
- 対象データ：山梨大学アクセスログ
  - ユーザが要求したURLをアイテムとし，要求時間順に並べたアイテム集合系列（約2週間分）
  - 1分間に要求されたアイテムを1つのアイテム集合とする
  - データ長：19,466
  - 総出現系列数：93,440,189個
  - 頻出系列数：5,962個

- 先行研究の最大メモリ消費量：約8,000,000
- 提案手法の最大メモリ消費量：約　120,000

Lossy Counting法による頻度表サイズの時間推移

山梨大学HPの2週間のアクセス
データ長:19,466基本単位

部分系列の最大長:3,
最小相対頻度:0.1，許容誤差:0.01



実験結果

ウィンドウ幅:3
最小相対頻度:0.1
許容誤差:0.01

データ長:19,466
最小頻度:1,946.6

頻出系列をどれだけ抽出できたか

頻度表制限値	抽出系列数	最終頻度表サイズ (瞬間最大値)	最終交換頻度	再現率(%)	適合率(%)
—	5,976	171,341(8,038,515)	—	100	99.8
180,000	5,981	159,687(180,000)	1,290. 4	100	99.7
150,000	5,983	148,206(150,000)	1,565. 1	100	99.6
130,000	6,087	102,070(130,000)	1,820.1	100	97.8
120,000	116,044	116,044(120,000)	1,979. 1	100	5.1
110,000	99,280	99,280(110,000)	2,167. 9	99.9	6.0
100,000	80,950	20,640(100,000)	2,393. 1	98.9	7.2

実験結果

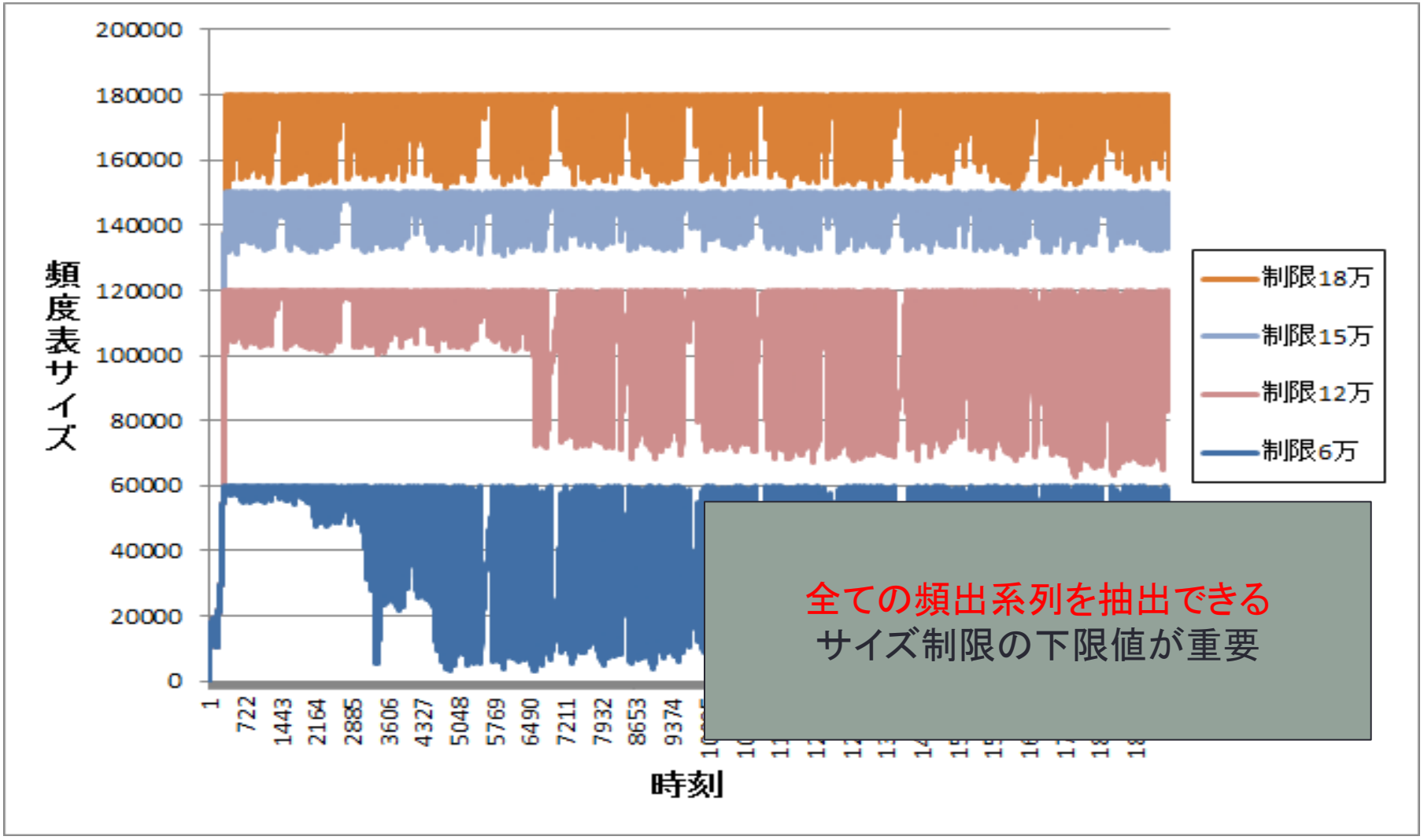
ウィンドウ幅:3
最小相対頻度:0.1
許容誤差:0.01

データ長:19,466
最小頻度:1,946.6

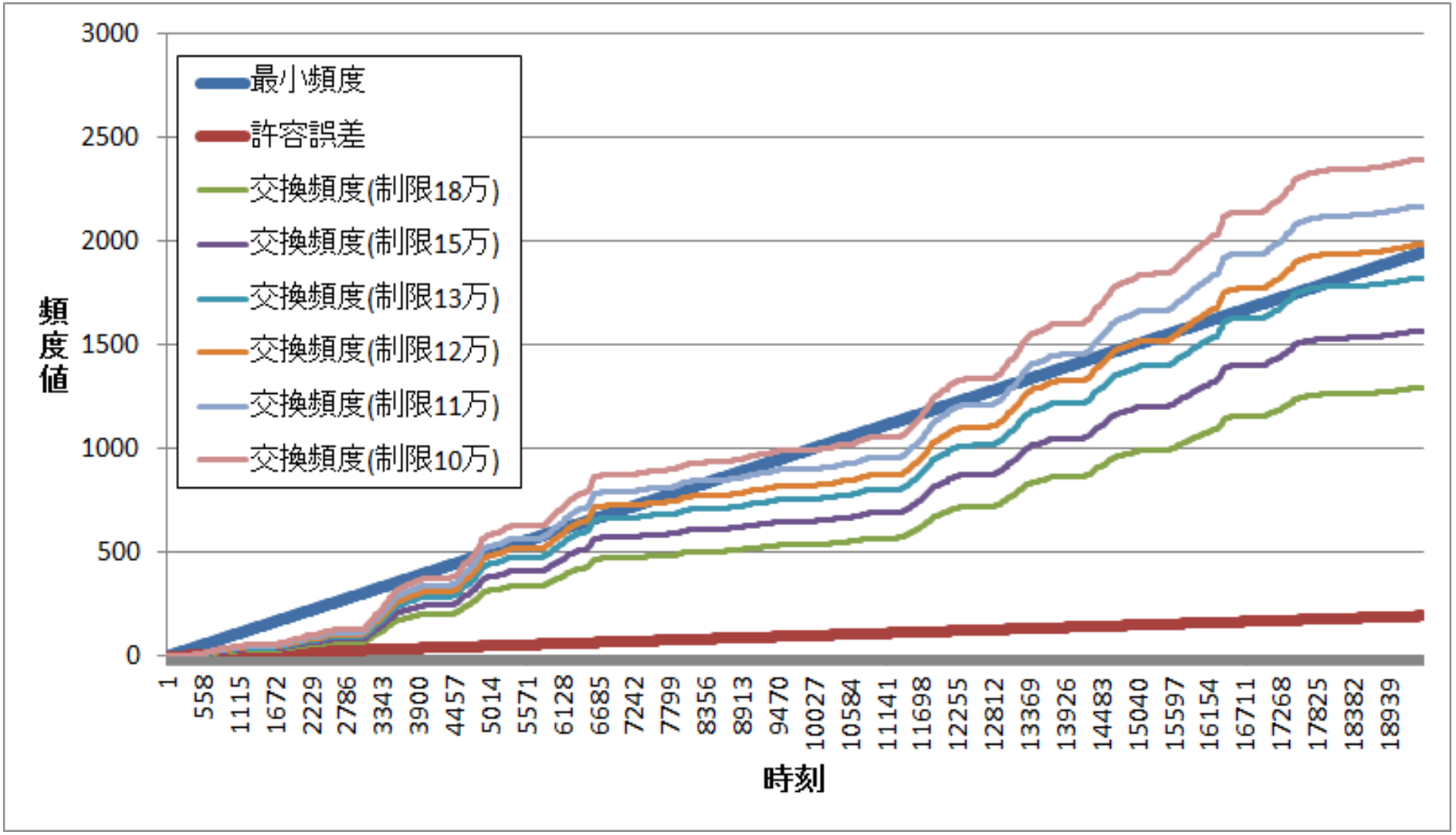
抽出結果がどれだけ正解に近いか

頻度表制限値	抽出系列数	最終頻度表サイズ（瞬間最大値）	最終交換頻度	再現率(%)	適合率(%)
—	5,976	171,341(8,038,515)	—	100	99.8
180,000	5,981	159,687(180,000)	1,290. 4	100	99.7
150,000	5,983	148,206(150,000)	1,565. 1	100	99.6
130,000	6,087	102,070(130,000)	1,820.1	100	97.8
120,000	116,044	116,044(120,000)	1,979. 1	100	5.1
110,000	99,280	99,280(110,000)	2,167. 9	99.9	6.0
100,000	80,950	20,640(100,000)	2,393. 1	98.9	7.2

提案手法の頻度表サイズ時刻推移



交換頻度の時刻推移

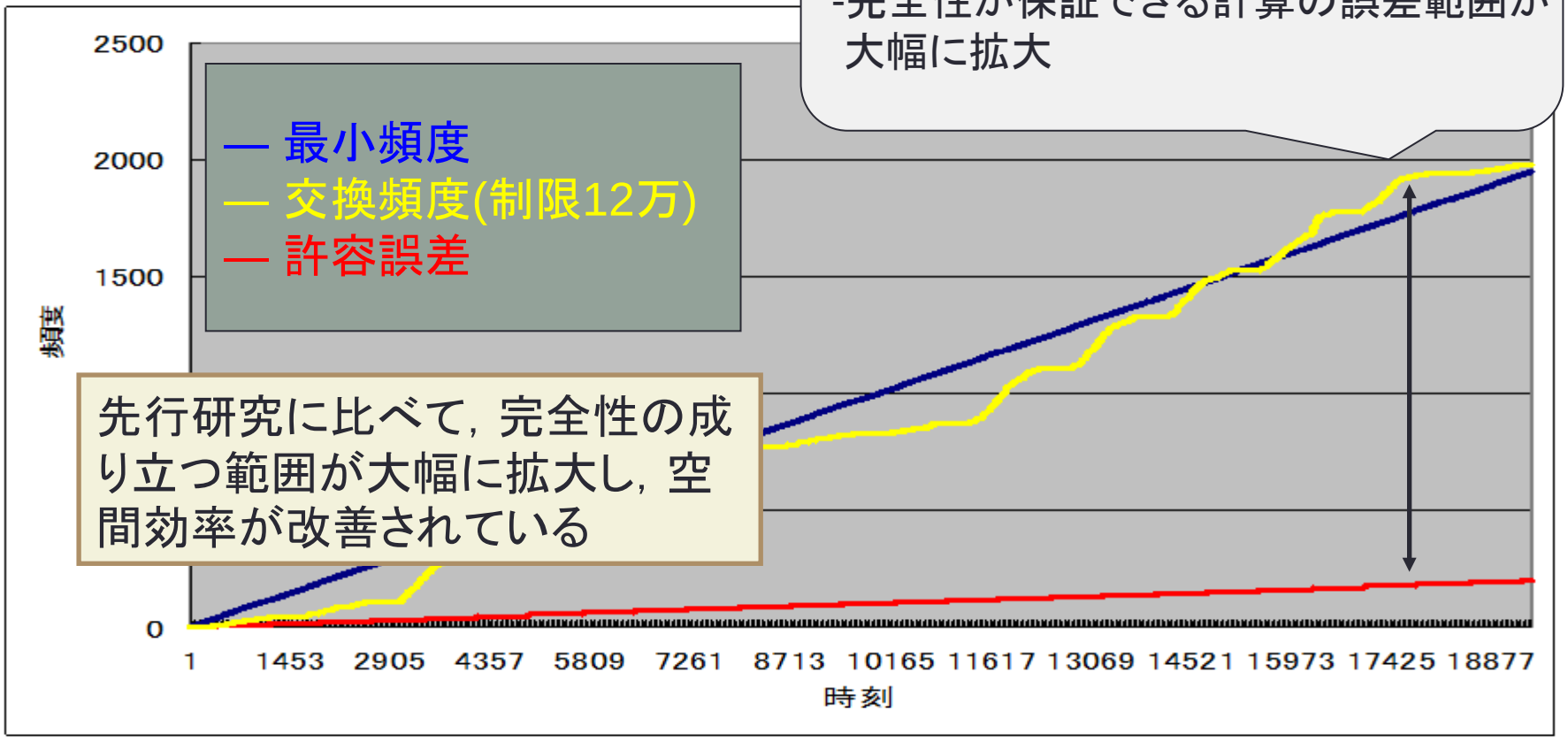

3000
2500
2000
1500
1000
500
0
頻度値
最小頻度
許容誤差
交換頻度(制限18万)
交換頻度(制限15万)
交換頻度(制限13万)
交換頻度(制限12万)
交換頻度(制限11万)
交換頻度(制限10万)
1
558
1115
1672
2229
2786
3343
3900
4457
5014
5571
6128
6685
7242
7799
8356
8913
9470
10027
10584
11141
11698
12255
12812
13369
13926
14483
15040
15597
16154
16711
17268
17825
18382
18939
時刻

空間効率

頻度表制限値12万
再現率:100%

ウィンドウ幅:3
最小相対頻度:0.1
許容誤差:0.01

-先行研究の挙動では，交換頻度は許容誤差にあたる
-完全性が保証できる計算の誤差範囲が大幅に拡大



先行研究に比べて，完全性の成り立つ範囲が大幅に拡大し，空間効率が改善されている

性能評価実験2

- パラメータ
  - 最小相対頻度0.01，許容誤差0.001，ウィンドウ幅3，
- 対象データ：地震発生時系列ログ
  - 1982年～2011年に発生した地震の記録(気象庁より引用)
  - 1アイテムは発生日時，震源地，マグニーチュード(0~9)の組み合わせ
  - 半日に起こった地震群を1アイテム集合とする
  - データ長：15,488
  - 平均アイテム集合サイズ：2.4
  - 総出現系列数　　：1,063,985
  - 頻出系列数　　：　　78

- 先行研究の最大メモリ消費量：約850,000
- 提案手法の最大メモリ消費量：約　14,000

地震発生系列データ

（検索期間 2011/02/01 00:00 – 2012/02/01 24:00）

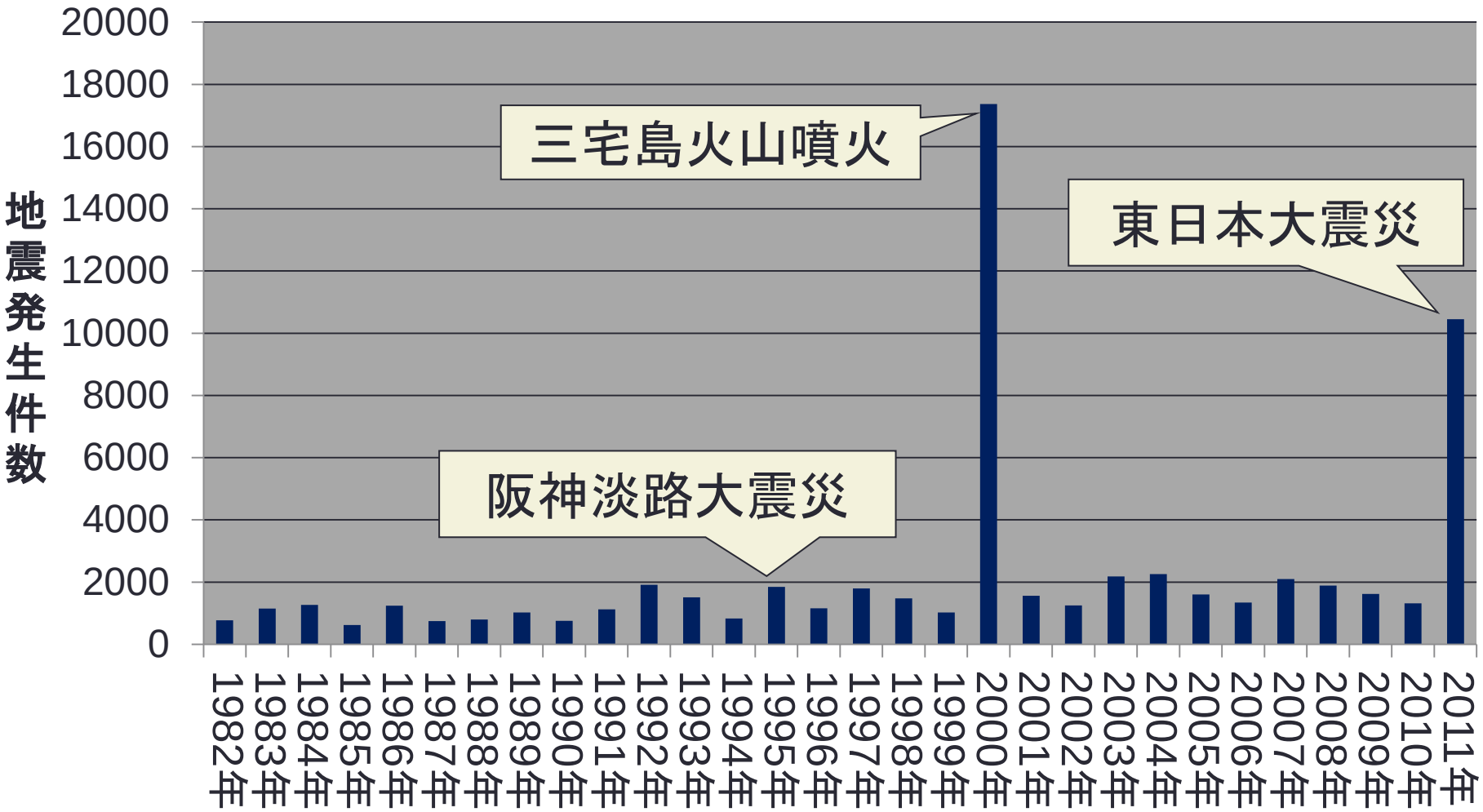
No.1	2011年2月1日02:32:26.1 31°50.0'N 132°10.8'E 22km M:4.1 日向灘 震度分布図	
宮崎県	1	川南町川南＊、宮崎市松橋＊
No.2	2011年2月1日04:07:9.0 36°27.6'N 137°21.3'E 12km M:2.8 富山県東部 震度分布図	
岐阜県	1	高山市上宝町本郷(旧)＊、飛騨市神岡町殿、飛騨市神岡町東町＊
No.3	2011年2月1日05:37:2.7 34°53.5'N 133°1.5'E 9km M:2.6 広島県北部 震度分布図	
広島県	1	庄原市西城町大佐＊、神石高原町油木＊
No.4	2011年2月1日15:42:34.4 34°43.7'N 135°16.2'E 6km M:2.2 兵庫県南東部 震度分布図	
兵庫県	1	神戸灘区神ノ木、芦屋市精道町＊
No.5	2011年2月1日18:21:43.6 42°34.8'N 144°13.0'E 63km M:3.7 釧路沖 震度分布図	
北海道	1	十勝大樹町生花＊

変換

気象庁-震度データベースより

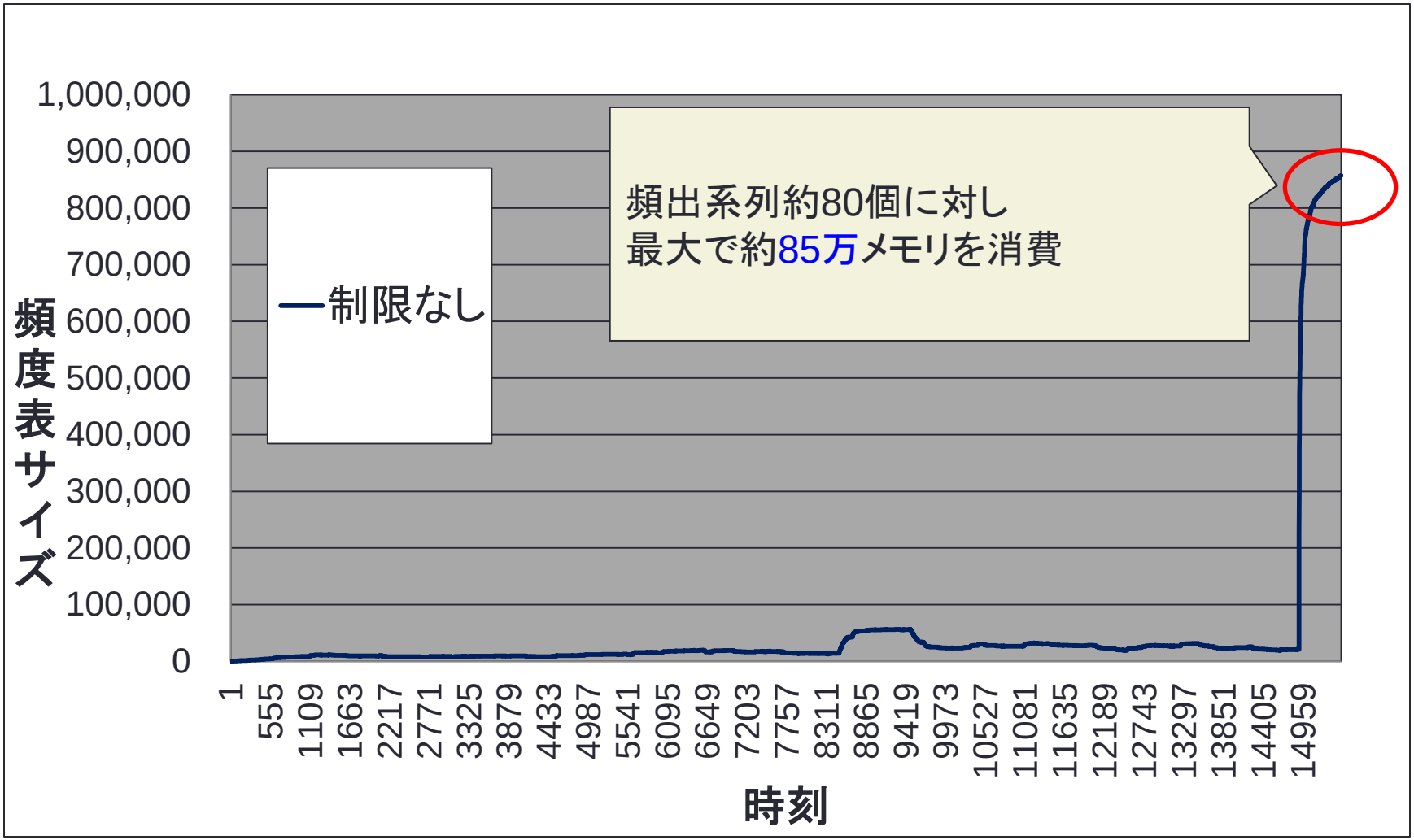
2011年2月1日,02:32:26.1,31°50.0'N,132°10.8'E,22km,M:4.1,日向灘,1
2011年2月1日,04:07:9.0,36°27.6'N,137°21.3'E,12km,M:2.8,富山県東部,1
2011年2月1日,05:37:2.7,34°53.5'N,133°1.5'E,9km,M:2.6,広島県北部,1
2011年2月1日,15:42:34.4,34°43.7'N,135°16.2'E,6km,M:2.2,兵庫県南東部,1
2011年2月1日,18:21:43.6,42°34.8'N,144°13.0'E,63km,M:3.7,釧路沖,1

参考：各年の地震発生件数



先行研究の手法による頻度表サイズの時間推移2

ウィンドウ幅:3
最小相対頻度:0.01
許容誤差:0.001
データ長:15,487



実験結果2

ウィンドウ幅:3
最小相対頻度:0.01
許容誤差:0.001

データ長:15,487
最小頻度:154.89

頻出系列をどれだけ抽出できたか

頻度表制限値	抽出系列数	最終頻度表サイズ (瞬間最大)	交換頻度	再現率(%)	適合率(%)
制限なし	85	857,236(857,246)	-	100	91.8
800,000	85	718,242(800,000)	15.9	100	91.8
400,000	85	392,084(400,000)	15.9	100	91.8
100,000	89	72,476(100,000)	30.9	100	87.6
80,000	90	50,333(80,000)	35.9	100	86.7
60,000	94	44,306(60,000)	43.9	100	82.3
40,000	103	39,814(40,000)	60.25	100	75.7
20,000	217	19,786(20,000)	112.44	100	35.9
15,000	816	7,449(15,000)	148.41	100	9.6
14,000	6,631	6,631(14,000)	158.12	100	1.1
13,000	2,115	2.115(13,000)	169.61	97.4	3.6
12,000	7,676	7,676(12,000)	180.93	96.2	0.9

実験結果2

ウィンドウ幅:3
最小相対頻度:0.01
許容誤差:0.001

データ長:15,487
最小頻度:154.89

抽出結果がどれだけ正解に近いか

頻度表制限値	抽出系列数	最終頻度表サイズ (瞬間最大)	交換頻度	再現率(%)	適合率(%)
制限なし	85	857,236(857,246)	-	100	91.8
800,000	85	718,242(800,000)	15.9	100	91.8
400,000	85	392,084(400,000)	15.9	100	91.8
100,000	89	72,476(100,000)	30.9	100	87.6
80,000	90	50,333(80,000)	35.9	100	86.7
60,000	94	44,306(60,000)	43.9	100	82.3
40,000	103	39,814(40,000)	60.25	100	75.7
20,000	217	19,786(20,000)	112.44	100	35.9
15,000	816	7,449(15,000)	148.41	100	9.6
14,000	6,631	6,631(14,000)	158.12	100	1.1
13,000	2,115	2.115(13,000)	169.61	97.4	3.6
12,000	7,676	7,676(12,000)	180.93	96.2	0.9

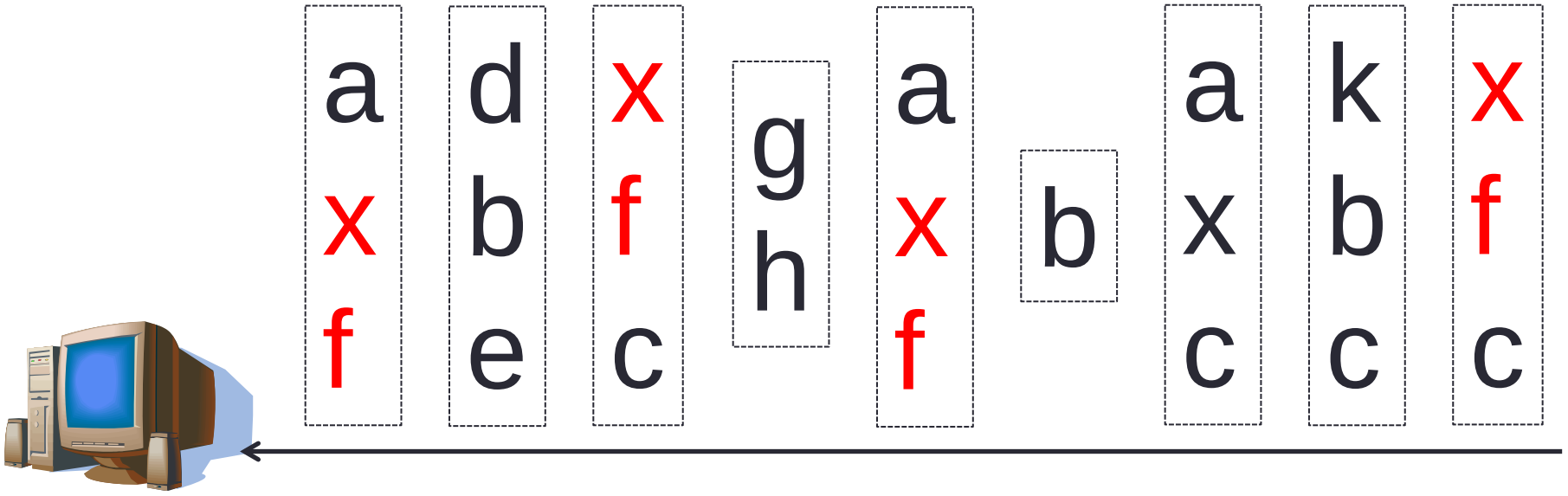
まとめと今後の課題

- 新しいオンライン型系列マイニング手法を提案
  - データの突発的な大量出現に対応可能
  - 更にいくつか改良手法を提案
  - どの手法も評価実験により有用性を示した

- 課題
  - 頻出アイテム集合の抽出
  - メモリ制限値の動的緩和についての考察

頻出集合マイニングへの適応

提案手法は頻出アイテム系列マイニング手法である
これを，頻出アイテム集合マイニングへ適応させる



頻出アイテム系列マイニング（提案手法）

頻出アイテム集合マイニング

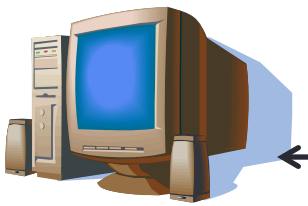
頻出アイテム集合マイニングの難しさ

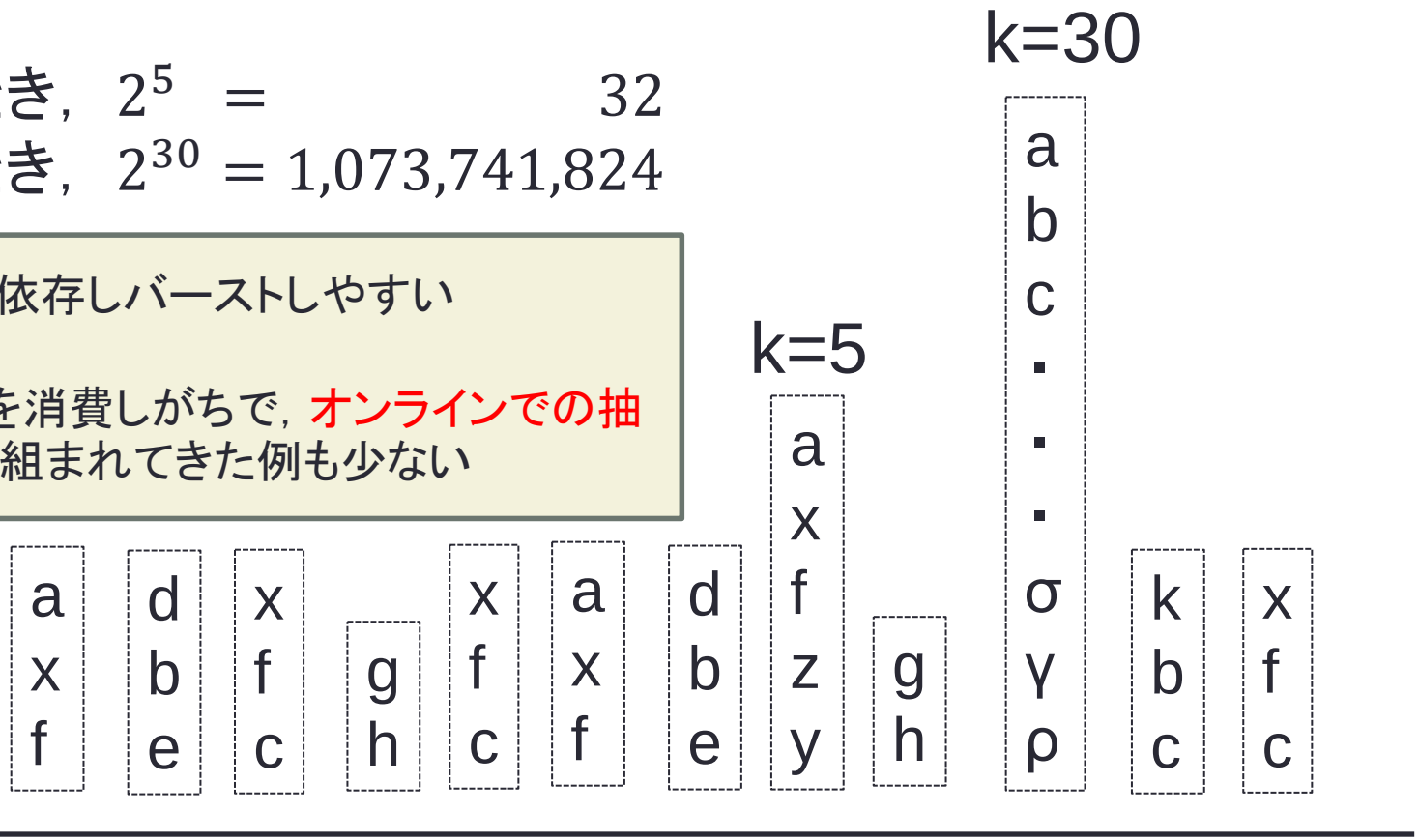
- 1トランザクション中にk個のアイテムがある場合，生成される部分系列は 2^k 個

$k = 5$ のとき，$2^5 = 32$
$k = 30$ のとき，$2^{30} = 1{,}073{,}741{,}824$

アイテム数 k に依存しバーストしやすい

どうしてもメモリを消費しがちで，オンラインでの抽出は難しく，取り組まれてきた例も少ない





頻出アイテム集合マイニングの難しさ

- 頻出集合の抽出は先行研究の基となった

  Lossy Counting法[Mankuら ‘02]でも複数のバケットによる頻出アイテム集合候補の絞り手法が提案されているが，あまり有効ではなく，特に，バースト的な出現への対策として機能しない