時間と共に変化する多重集合に対するmin-hashの高速計算

電気通信大学 情報理工学研究科 情報・ネットワーク工学専攻

古賀研究室 三原寛寿

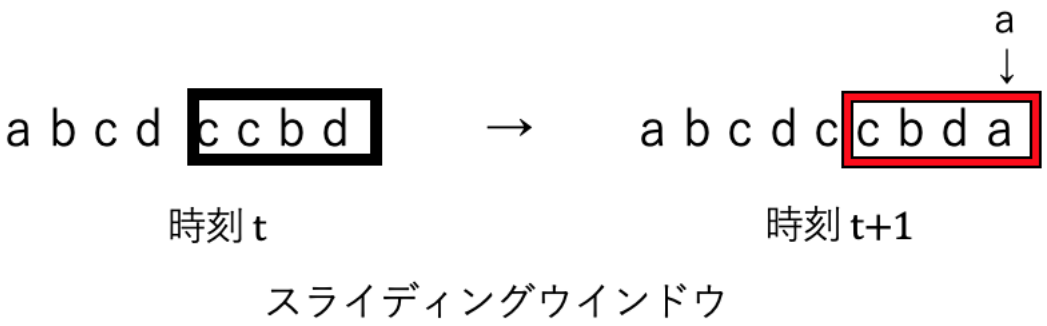
2022/2/28

概要

- 近年，IoTやSNSの発展に伴いストリームデータが取り扱われる機会が増加

→ストリームデータの類似検索の重要性も増加

- ストリームデータ：時間と共に変化するデータ
  - ➢スライディングウインドウモデル



- ストリームデータの類似検索とは、ストリームデータを**要素が変化する集合**と捉えて、**集合間類似検索**に帰着できる

集合間類似検索の類似度

- Jaccard係数
  - ２つの集合に含まれている要素のうち共通要素が占める割合
  - 計算するオーバーヘッドが大きい

$$\mathrm{sim}(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

- ハッシュを用いた集合間類似検索の高速化：**Min-Hash**
  - 集合に対するハッシュ関数
  - 2つの集合のハッシュ値が一致する確率はJaccard係数と等しい

$$P\big(mh(A) = mh(B)\big) = \frac{|A \cap B|}{|A \cup B|}$$

Min-Hashの計算

- 要素にランダムな値を割り当てる
  - ➢多重集合の場合
    - ■同一要素を複数持つことがある
    - ■同じアルファベットに異なる割り当て値を与える

	a	b	c	d
1個目	3	4	7	6
2個目	5	1	8	2

- 集合Sのハッシュ値$h(S)$
  - ➢Sの要素への割り当て値の最小値
  - ➢例：$S = \{a, b, c, c, d, d\}$

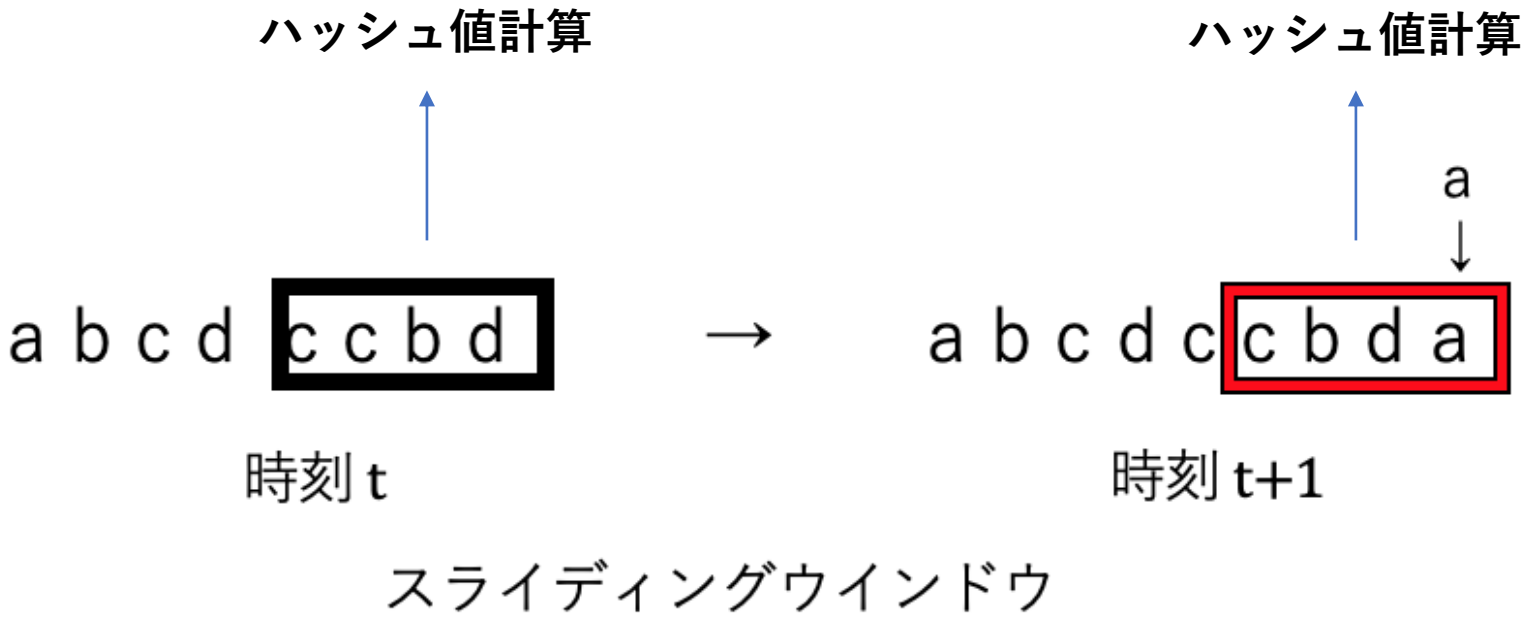
集合Sの要素	a_1	b_1	c_1	c_2	d_1	d_2
割り当てた値	3	4	7	8	6	2

↓

ハッシュ値 ： $h(S) = 2$

ストリームデータに対するMin-hash

- ハッシュ値計算



毎時刻ハッシュ値の再計算が必要

本研究の内容

- スケッチの更新を効率化するため、動的に変化する集合に対する Min-Hash のハッシュ値計算方法を提案

- スライディングウインドウで多重集合を取り扱える初めての手法である**SWMH**を提案
  - ➢既存手法では、(1)SWで発生する要素の削除と(2)多重集合を同時に取り扱えない
  - ➢SWで(多重集合ではなく)集合を取り扱ったDatarらの手法を拡張

Datarらによる手法 **多重集合を取り扱うことができない**

- 将来的に最小値になり得ない要素を削除
- 残りをMinlistで管理し、Minlistの最小値をハッシュ値とする
- 最小値をMinlistから選択することでハッシュ値更新を高速化

最小値になり得ない条件：自分より後ろに小さい要素が存在

スライディングウインドウ

g	x	r	y	w	e	m	l
2	7	1	4	10	5	9	8

候補集合 Minlist

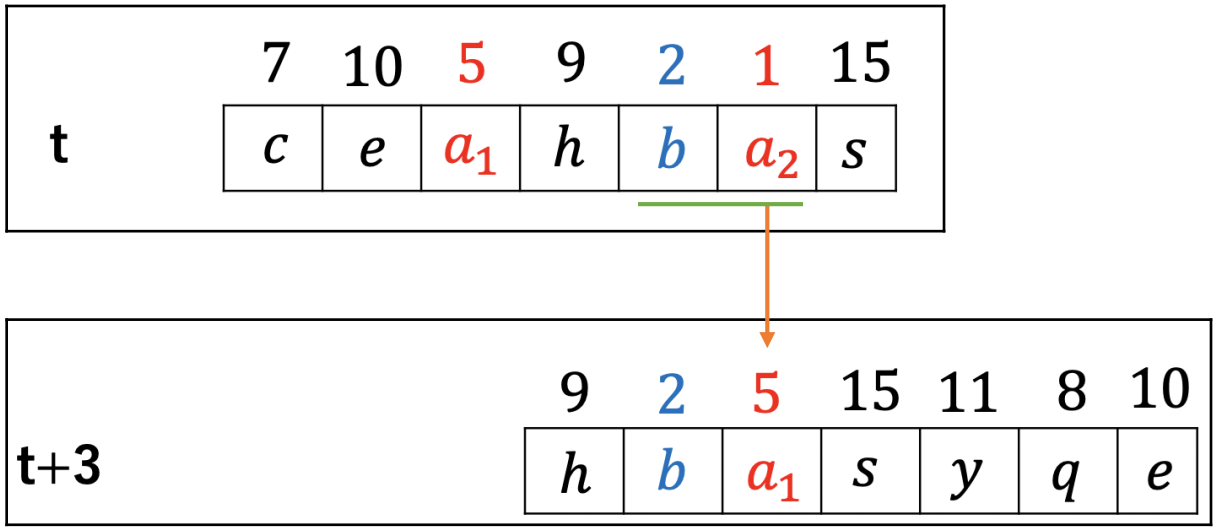
1	4	5	8

多重集合の難しさ

- 多重度によってラベルの割り当て値が変化する

→最小値になり得るかの判定が難しい

- SW内において先頭からi番目のアルファベットaをa_iとする
- 時刻 tの時、bの後方に小さい値を持つaが存在
- t+3の時
  - ➢後ろのaが1から5に変化する
  - ➢bは最小値になる



提案手法：SWMH(Sliding Window Min-Hash)

1. ハッシュ値を変えることなく割り当て値を修正
   - Minlistに同一ラベルの要素が高々1つしか残らないことを保証
   - 任意の要素の割り当て値が減少しないことを保証

2. 最小値になりえるかの判定条件を修正
   - 後方の要素の割り当て値の上限値との比較
   - 後方の要素の割り当て値との比較ではない

割り当て値の修正

- 同じアルファベットで、i番目の割り当て値$\pi(a_i)$より$i+1$番目の割り当て値$\pi(a_{i+1})$が大きい

↓

- $\pi(\alpha_{i+1})$を減らして$\pi(\alpha_i)$にする
- $\pi(\alpha_i)$がiに対して増加しないことを保証

	a	b	c	d
1個目	3	4	7	6
2個目	5	1	8	2

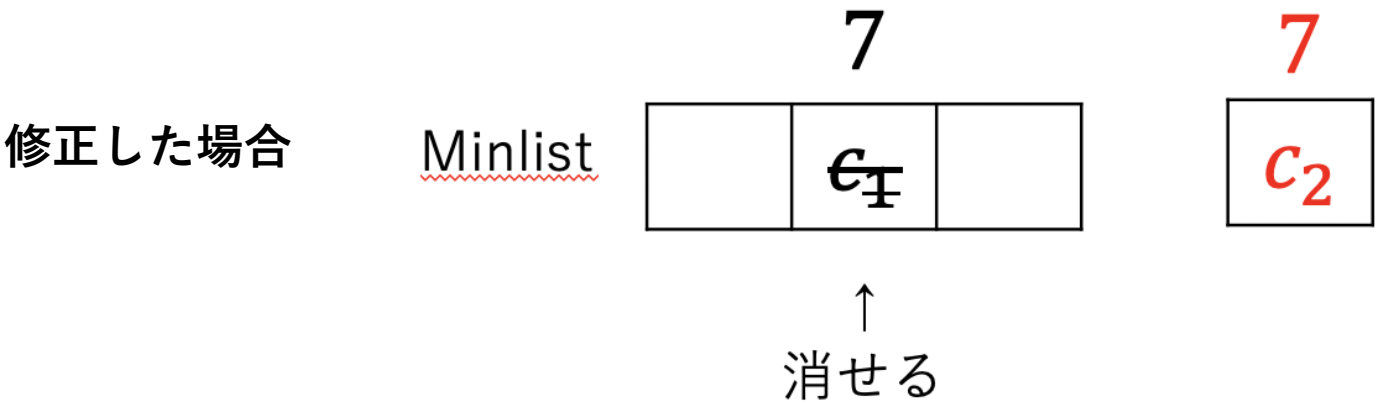
↓

	a	b	c	d
1個目	3	4	7	6
2個目	3	1	7	2

→この修正により、ハッシュ値は変わらない

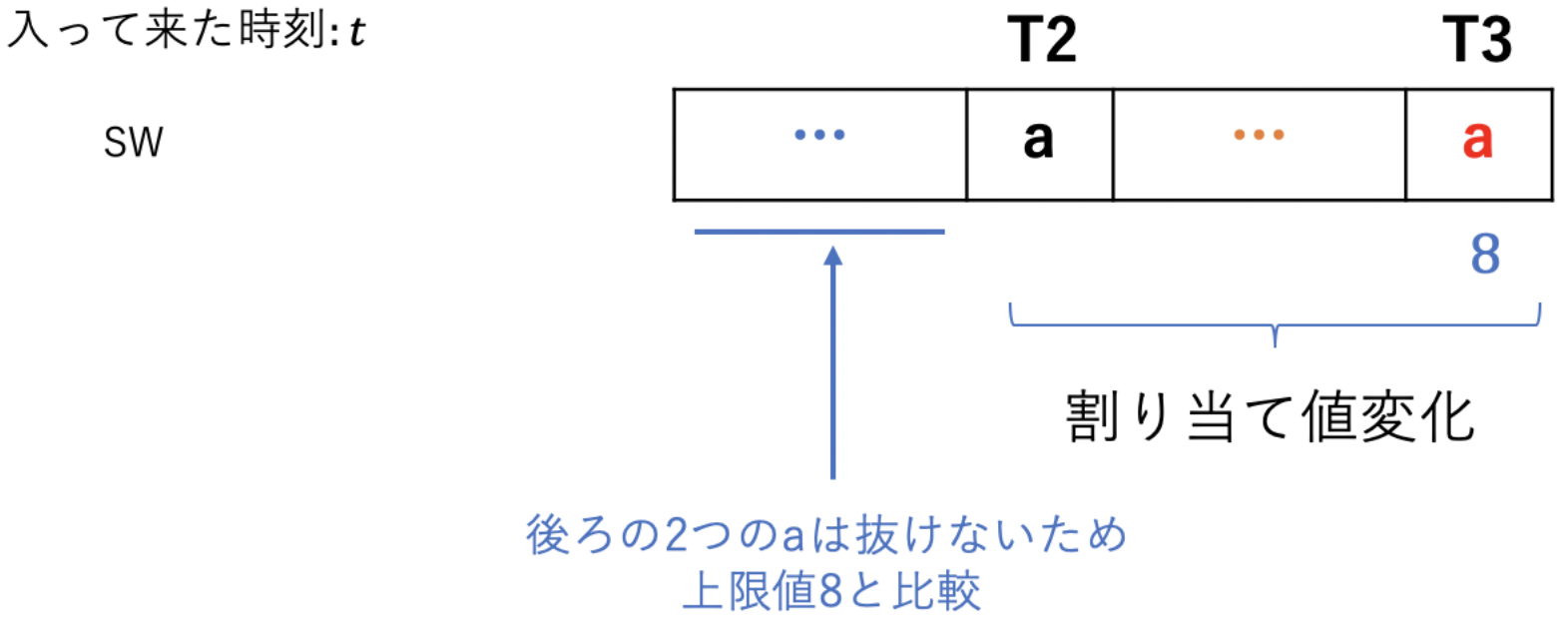
修正の効果

- Minlistに同一ラベルの要素が高々1つしか残らない



最小値になり得るかの判定条件

- 到着した要素eの現在の値ではなく、**上限値**と消される側の値の大小関係によって、消してよいかを判定

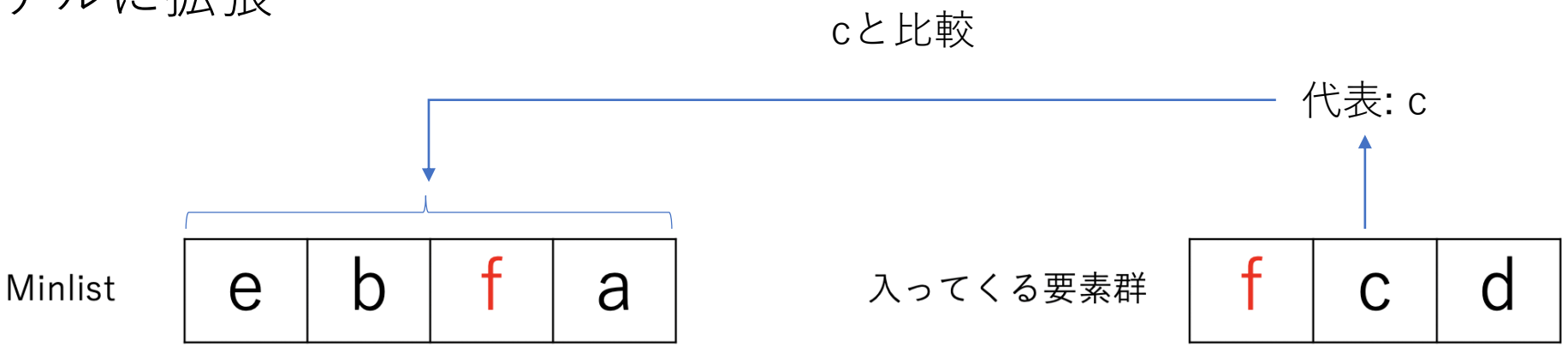


aの割り当て値

	1	2	3
a	10	8	3

バッチSWMH

- SWMHをデータストリームに毎時刻複数個の要素が到着するモデルに拡張



- Minlist更新の手順
  1. 到着した要素群から代表アルファベットの選択
  2. 代表アルファベットと比較して、Minlistから必要ない要素の削除
  3. Minlistへ到着した要素群を追加

- 問題点
  - ➢代表アルファベット以外とは比較しない
  - ➢代表以外の到着した要素と同じラベルの要素が削除されない

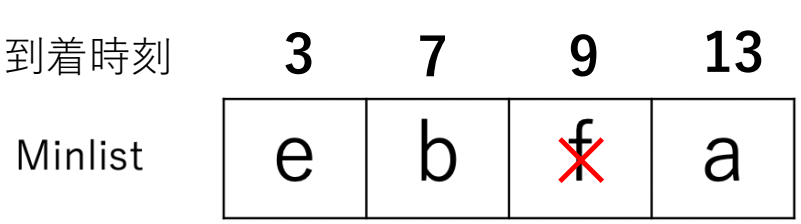
問題点への対策

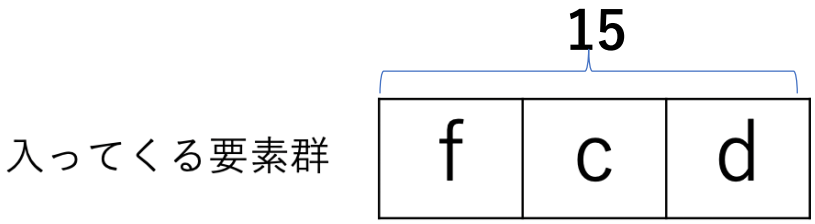
- 狙い：到着した要素群と比較なしで、同じラベルの要素を削除
- algorithm
  - ➢各アルファベットの最新の到着時刻を表で管理

アルファベット	a	b	c	d	e	f
時刻 T	13	7	15	15	3	15

  - ➢Minlist内の要素eの到着時刻$t(e)$と表の時刻$T(l(e))$を比較
    - ■$l(e)$: eのラベル

if ($t(e)$! = $T(l(e))$) **Minlistからeを削除**





$t(e)$!= $T(l(e))$ ⇨ 後方に同じラベルが存在

実験評価

- SWMHとバッチSWMHを人工データと実データを用いて実験的に評価

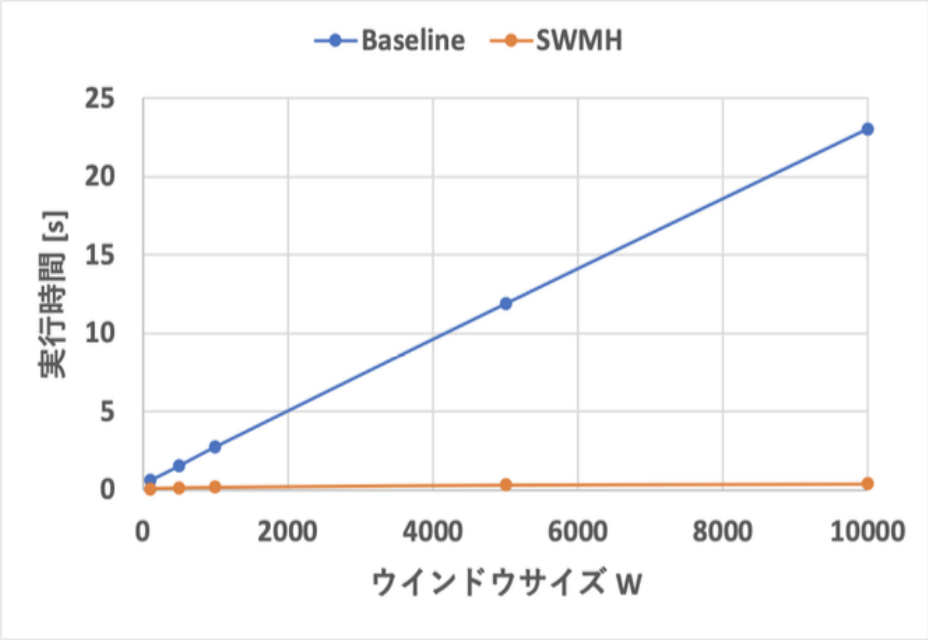
- 人工データセット：zipf分布に従った長さ100,000のストリーム
  - α : zipf分布の偏りをコントロールするパラメータ
  - $|\phi|$: アルファベットの種類数

- 実データセット:2種類のデータからそれぞれ長さ100,000のストリームを生成
  - connect dataset :117種のラベル
  - mushroom dataset :127種のラベル

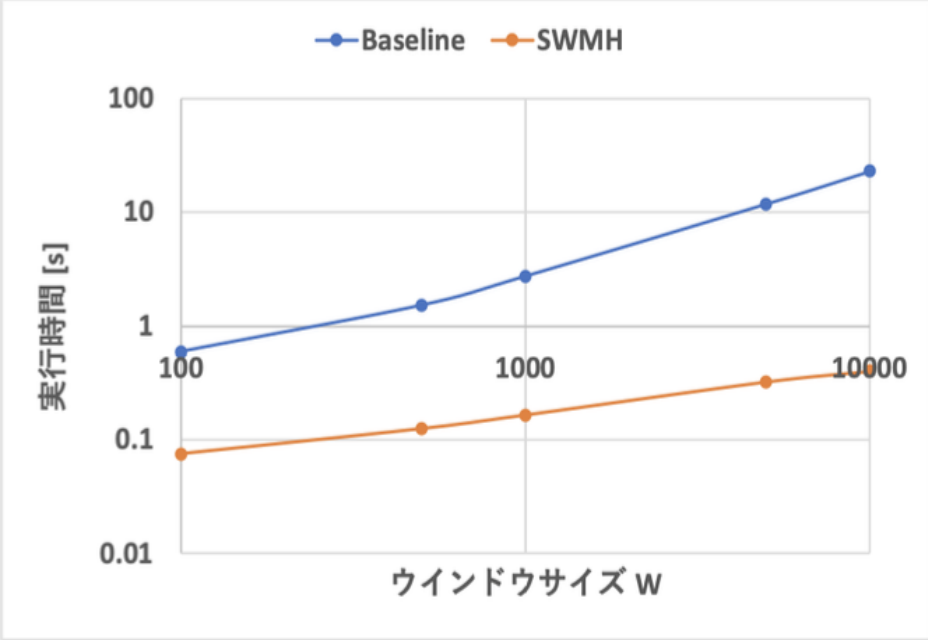
- デフォルトパラメータ： $\alpha = 1, |\phi| = 100, W = 100$

SWMH:ウインドウサイズWを変えて実験

Baseline: スライディングウインドウから毎時刻Min-hashを再計算する手法

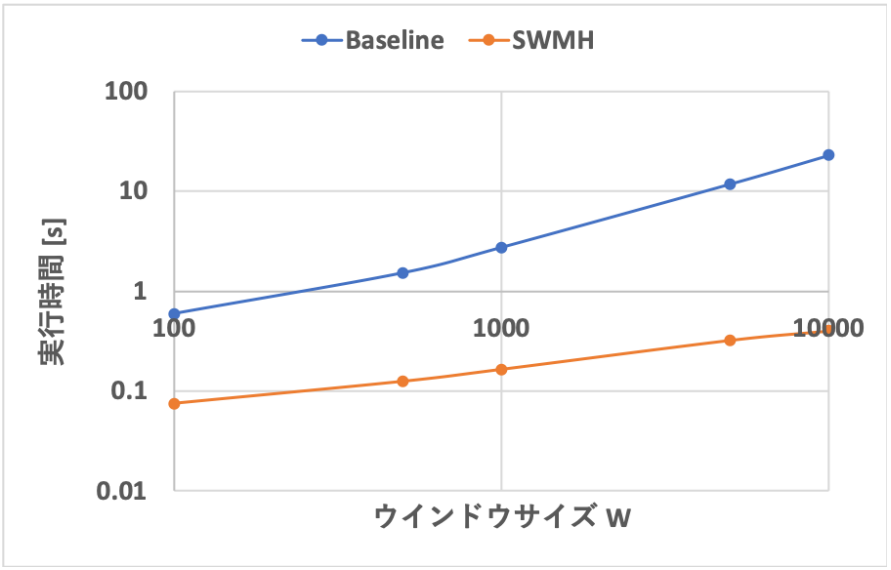


mushroom dataset

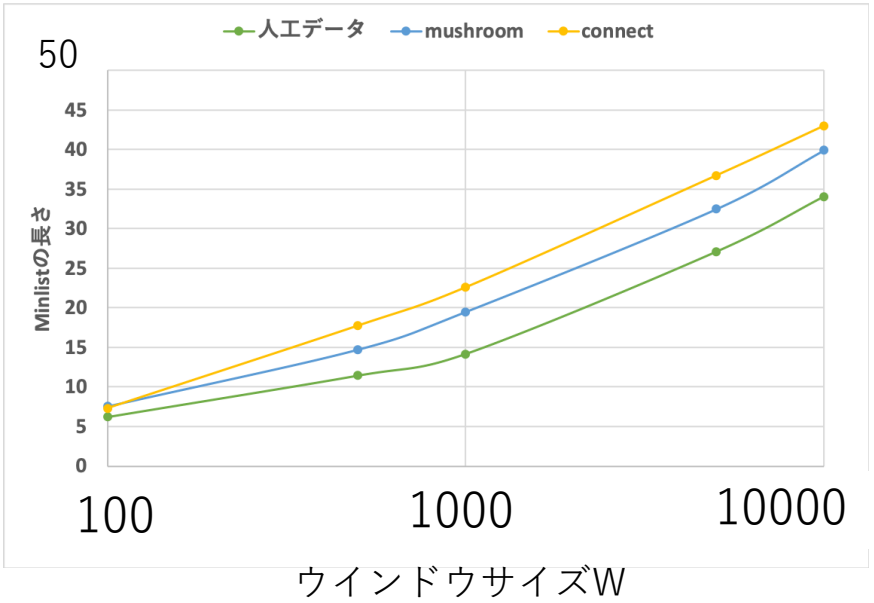


mushroom datasetにおける対数グラフ

Minlistの長さ



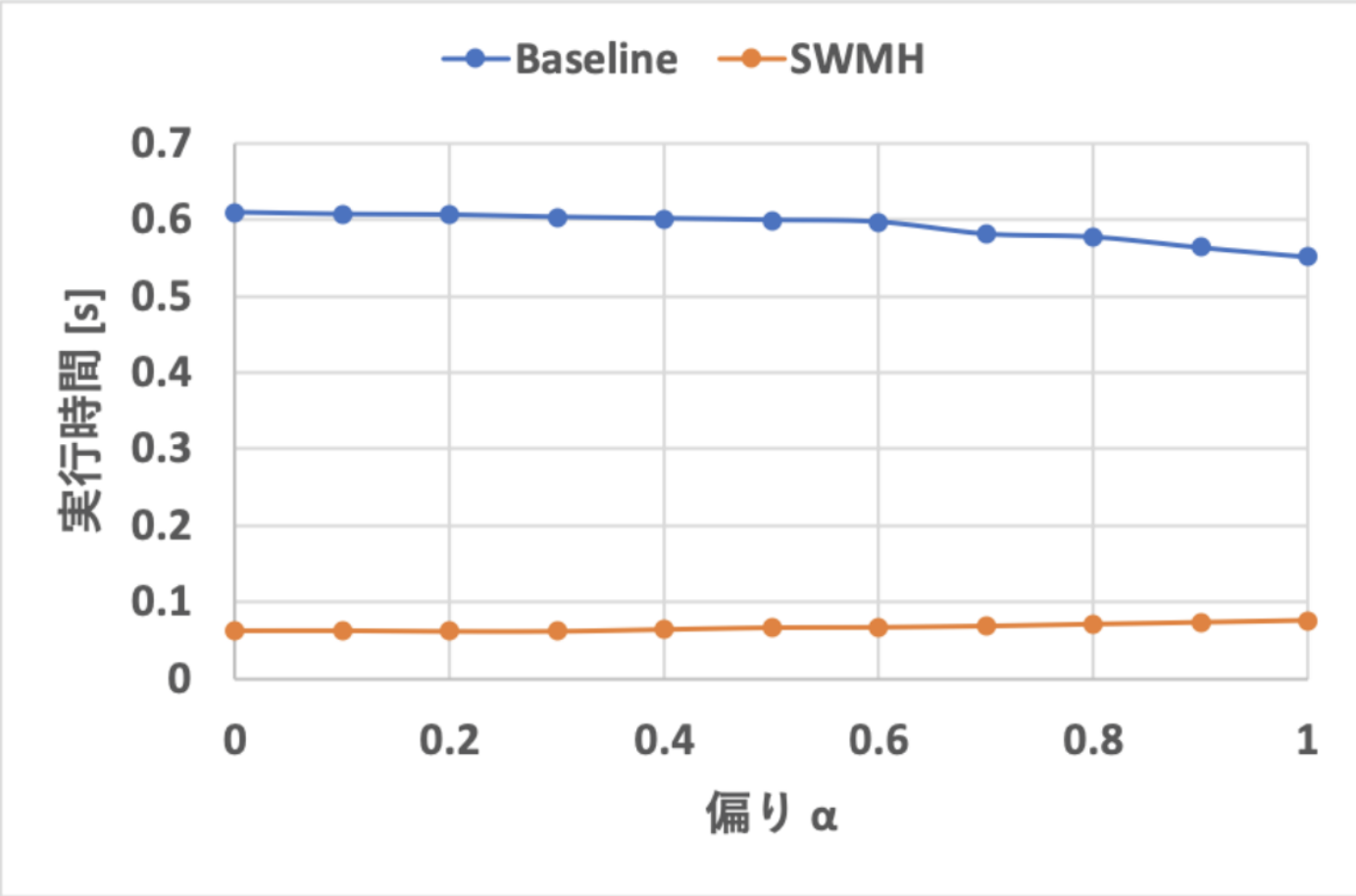
実行時間



Minlistの長さ

W=1,000、10,000に対して、Minlist95%以上削減

SWMH : zipf分布の偏りαを変えて実験



SWMHの方がどのαでも10倍ほど早い

バッチSWMHの実験評価

- $\alpha = 1, |\phi| = 100, W = 100$ という組み合わせをデフォルトパラメータとし，実験を行 なった.

- 到着レート：$c = 5$

実行時間 [s]

データセット	人工	connect	mushroom
SWMH	0.0742	0.0743	0.073
バッチ SWMH	0.042	0.0365	0.037

バッチSWMHの方がどのデータセットでも約2倍早い

まとめ

- 本研究ではデータストリームに対 するハッシュ値の更新アルゴリズムを取り扱った
- スライディングウィンドウに対するMin-Hashのハッシュ 値更新アルゴリズムSWMH
- SWMHを複数個の要素がスライディングウインドウに到着するモデルに対応するよう拡張したバッチSWMH
- 要素の削除と多重集合を取り扱うことができ、実験評価より効果的であることを示せた
- 今後の研究課題
  - 近似ヒストグラムを用いたメモリ使用量の削減