A-015

カウンタを用いた ε 近似 ϕ 分位数のオンライン抽出の高性能化

前田浩丞 *　　岩沼宏治 †

1　はじめに

近年，実世界から情報を収集するためのセンサネットワークに関する研究が急速な進展を見せている．それに伴い，大量のセンサの情報を統合・圧縮する技術もその重要性を増している．情報を圧縮・抽象化する技術の一つとして，分位数を用いる方法がある．

Greenwald and Khanna [1] では，分位数の ϵ 近似という概念を提案し，ストリーム中のデータを圧縮保持しながら，ϵ 近似分位数をオンライン出力できる GK アルゴリズムを提案している．そこでは，これまで入力されたのデータの数を N とするとき，空間計算量 $O(\frac{1}{\epsilon}\log \epsilon N)$ へ圧縮する ϵ 近似分位数サマリが構築されている．Zhang and Wang [2] では，GK アルゴリズムを分割統治法を用いて更なる高速化を試みている．

しかし，これらの 2 つのアルゴリズムは同一データがストリーム中に複数出現した場合には，殆ど同一の出現記録をサマリ中に重複して保存しており，明らかに冗長と考えられる．何らかの**カウンタ**機構を導入すれば，この冗長性を除去できると考えられる．そこで我々は [3] においてカウンタ併用型の GK アルゴリズムを提案し，性能評価実験を行った．出力される近似分位数のランク誤差を抑制しながら，分位数サマリを更に圧縮縮小できることが確認している．ただ，[3] の提案アルゴリズムでは，近似分位数の近似誤差の保証が困難であった．分位数誤差は分位数サマリを縮小圧縮するときに発生する．そこで本研究では，分位数サマリの縮小圧縮処理に制約条件を課すことにより，発生する誤差の制御を試みる．また，制約条件による圧縮の効果の変化等を評価実験を透して確認する．

2　準備

$\mathcal{I}$ を全順序関係 $<$ を持つ**アイテム**の全順序集合 $\{e_1, ..., e_k\}$ とする．アイテム $e_{t_i} \in \mathcal{I}$ を時刻 t_i に到着したアイテムとするとき，アイテムの列 $\langle e_{t_1}, e_{t_2}, \ldots, e_{t_N}\rangle$ を長さ N の**データストリーム** $\mathcal{DS}$ と呼ぶ．このとき $\mathcal{DS}$ 中の e_i の**ランク** $\mathrm{rank}(e_i)$ を以下のように定義する．

$$\mathrm{rank}(e_i) = \sum_{e_j \le e_i} \mathrm{freq}(e_j)$$

但し $\mathrm{freq}(e_j)$ はアイテム e_j の**頻度**であり，$\mathcal{DS}$ 中で e_j が現れた回数を表す．

長さ N のストリーム $\mathcal{DS}$ における ϕ **分位数** (但し $\phi \in [0,1]$) とは，$\mathrm{rank}(e_i) = \lfloor \phi N \rfloor$ なるアイテム $e_i \in \mathcal{DS}$ を表す．例えば，長さ 10 のストリーム $\mathcal{DS}_0 = \langle 31, 78, 11, 11, 58, 99, 7, 58, 22, 58\rangle$ において，0.1 分位数は 7，0.3 分位数は 11 であり，0.7 分位数は 58 となる．

ϕ 分位数は，全てのアイテムに一度ソートすることで容易に求めることができるが，これはオフライン処理とである．アイテムの数が増えればソートに必要な時間も無視できないほど大きくなり，リアルタイム処理には向かない．また正確な ϕ 分位数を求めるためには，全てのアイテムの頻度を記録しておく必要がある [1] が，ストリームに流れるアイテムの種類は膨大な数になる場合には実際的ではない．そこで Greenwald ら [1] は，ランクに誤差 $\lfloor \varepsilon N \rfloor$ を許容した ε **近似分位数**を考え，記録しておくアイテムの数を大幅に削減しながらオンライン ϵ 近似計算を行う GK アルゴリズムを提案している．

長さ N ストリーム $\mathcal{DS}$ 上での ϵ **近似** ϕ **分位数**は以下の不等式を満たすアイテム e のことであり，一般に複数個存在する．

$$\lfloor (\phi - \epsilon)N \rfloor \le \mathrm{rank}(e) \le \lfloor (\phi + \epsilon)N \rfloor$$

定義から明らかなように，N が増加した際には ϵ 近似 ϕ 分位数の数も増加する．分位数に ϵ 近似を許容する場合，全てアイテムの頻度を記録する必要は無く，少数の代表的なアイテムの頻度のみを記録しておけば十分である．代表アイテムの情報を保持する構造は**分位数サマリ**と呼ばれるが，そのサマリ構造もまた複数考えられる．サマリ構造の性能はアルゴリズムの性能に直結し，極めて重要である．

3　先行研究：GK アルゴリズム

GK アルゴリズム [1] は，ϵ 近似分位数サマリを構築するアルゴリズムである．まず，サマリ中に記録されている代表アイテム e が近似するランクの下界を $r_{\min}(e)$，上界を $r_{\max}(e)$ と表記する．このとき，$\mathrm{rank}(e) - r_{\min}(e) \le \lfloor \epsilon N \rfloor$ かつ $r_{\max}(e) - \mathrm{rank}(e) \le \lfloor \epsilon N \rfloor$ が成り立っているならば，e はこの上限と下限の間のランクを持つ分位数の ϵ 近似となる．この $r_{\min}(e)$ と $r_{\max}(e)$ は，サマリ中には相対的な形で記録することにより，サマリの更新の演算を高速に行うことができる．そのため，GK アルゴリズムでは以下のような 3 項組の列を分位数サマリ $\mathcal{S}$ とし

* 山梨大学大学院医農工学総合教育部コンピュータ理工学コース
† 山梨大学大学院総合研究部

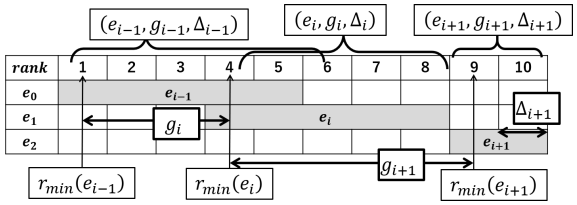

図 1　GK サマリの構造

て保持している．

$$\langle\langle e_0, g_0, \Delta_0\rangle, \langle e_1, g_1, \Delta_1\rangle, \langle e_2, g_2, \Delta_2\rangle, \ldots, \langle e_{s-1}, g_{s-1}, \Delta_{s-1}\rangle\rangle$$

但し，$e_0 \le e_1 \le e_2 \le \ldots \le e_{s-1}$ である．GK サマリの模式図を図 1 に示す．各 g_i と Δ_i は, $g_i = r_{\min}(e_i) - r_{\min}(e_{i-1})$ と $\Delta_i = r_{\max}(e_i) - r_{\min}(e_i)$ となるように設定される．また $\mathcal{DS}$ 中の最小値と最大値は, それぞれ $\langle e_0, 1, 0\rangle, \langle e_{s-1}, 1, 0\rangle$ として常に $\mathcal{S}$ に保持されるように構築される．このとき下界 $r_{\min}(e_i)$ と上界 $r_{\max}(e_i)$ は, それぞれ以下のように求めることができることに注意されたい．

$$r_{\min}(e_i) = \sum_{j=0}^{i} g_i, \quad r_{\max}(e_i) = r_{\min}(e_i) + \Delta_i$$

簡易型 GK アルゴリズムの擬似コードを Algorithm 1 に示す．ストリームが終了するまで，サマリに対して以下の 3 項組の挿入と削除を繰り返している．

INSERT$(e, \mathcal{S})$: サマリ $\mathcal{S}$ 中で $e_{i-1} \le e < e_i$ となるような最小の e_i を見つけ，$\langle e, 1, g_i + \Delta_i - 1\rangle$ を e_{i-1} と e_i の 3 項組の間に挿入する．但し，$\mathcal{S}$ の先頭もしくは末尾として挿入する際は $\langle e, 1, 0\rangle$ を挿入する．

DELETE$(\mathcal{S}, N, \epsilon)$: 削除できる 3 項組が $\mathcal{S}$ 中に無くなるまで，以下の削除ステップを繰り返す．

削除ステップ　$g_i + g_{i+1} + \Delta_{i+1} \le 2\epsilon N$ である e_i を見つけ，その 3 項組 $\langle e_i, g_i, \Delta_i\rangle$ を削除し，e_{i+1} の 3 項組を $\langle e_{i+1}, (g_i + g_{i+1}), \Delta_{i+1}\rangle$ に更新する．

GK アルゴリズム [1] では，サマリ $\mathcal{S}$ の空間計算量を $O(\frac{1}{\epsilon}\log \epsilon N)$ に抑えこむために，DELETE の計算戦略が工夫されている．本論文の Algorithm1 はその部分を省略した簡易型コードとなっている．この簡易型コードは空間計算量の保証はできないが，以下のように計算解の ϵ 近似性は保証できるものである．

定理 1（計算解の ϵ 近似性の保証 [1]） Algorithm1 の OUTPUTQUANTILE が返す値 e_{j-1} は以下の性質を満たす．

$$\lfloor(\phi - \epsilon)N\rfloor \le r_{\min}(e_{j-1}) \text{ and } r_{\max}(e_{j-1}) \le \lceil(\phi + \epsilon)N\rceil$$

Algorithm 1 簡易型 GK アルゴリズム

Input: $\mathcal{DS}$: ストリーム, ϵ: 許容誤差
Output: $\mathcal{S}$: 分位数サマリ

```
==================================
S ← ∅; N ← 1; e ←read(DS)
while e is not EOS do          ▷ ストリームが終了するまで
    INSERT(e, S)                ▷ 要素 e を S に挿入する
    if N mod ⌊1/(2ε)⌋ = 0 then  ▷ 周期 ⌊1/(2ε)⌋ で削除処理を行う
        DELETE(S, N, ε)
    end if
    if φ 分位数を出力する要求があった then
        OUTPUTQUANTILE(S, N, ε, φ)
    end if
    N ← N + 1; e ← read(DS)
end while
return S
==================================
function OUTPUTQUANTILE(S, N, ε, φ)
    if ⌈φN⌉ ≥ N − ⌊εN⌋ then
        return e_{s−1}          ▷ S の最後のアイテムを返す
    else
        不等式 r_max(e_j) > ⌊φN⌋ + ⌊εN⌋ を満たす最小（最初) の e_j を S から見つける
        return e_{j−1}  ▷ 見つけた e_j の直前のアイテムを返す
    end if
end function
```

4　カウンタを用いた GK アルゴリズムの効率化

GK アルゴリズムはストリーム中に何度も同じアイテムが出現している場合，何度も殆ど同じ 3 項組を挿入するので時間的にも空間的にも効率が悪い．この問題の最もシンプルな解決法はカウンタの利用と思われる．我々は文献 [3] において，カウンタを用いた GK アルゴリズムの自然な改良として C-GK アルゴリズムを提案し，幾つかの考察を行っている．性能比較実験を行った結果，高速化とサマリの縮小効果が確認できている．しかし，に ϵ 近似性を理論的に保証することが難しく，未解決問題として残っていた．本研究ではアルゴリズムの再考察を行い，計算解の ϵ 近似性について保証を考察する．また，さらなる高速化するために，サマリを格納するデータ構造の改良を検討を行う．

4.1　C-GK アルゴリズムの再考察

GK アルゴリズムとその拡張を考えるとき，分位数サマリの縮小演算が情報損失と誤差をもたらすことから，その設計は極めて重要な課題である．

例として表 1 のストリーム $\mathcal{DS}_1$ を考える．このストリームは，アイテム a, b, c が連続して出現しているため，それぞれの出現に対して別個に 3 項組を生成して記録するよりも，アイテム毎にカウンタを用意して出現

表 1　ストリームの例 $\mathcal{DS}_1$

時刻	1	2	3	4	5	6	7	8	9	10
アイテム	a	a	a	a	a	b	c	c	c	c

回数を記録する方が自然かつ合理的と思われる．そのためのサマリ要素としては，カウンタ c_i を組み込んだ 4 項組 $\langle e_i, c_i, g_i, \Delta_i\rangle$ を新しく導入する．この 4 項組をした C-GK アルゴリズムでは，$\mathcal{DS}_1$ を読み終えたときの分位数サマリは，アイテム間の全順序を $a < b < c$ と仮定すれば，以下のようになるのが自然と思われる．

$$\langle a, 5, 1, 0\rangle, \langle b, 1, 1, 0\rangle, \langle c, 4, 1, 0\rangle$$

一方で標準 GK アルゴリズムは，サマリ要素の削除処理を行わない場合には，サマリは以下のようになる．

$$\overbrace{\langle a,1,0\rangle,\ldots,\langle a,1,0\rangle}^{5\text{ 個}},\langle b,1,0\rangle,\overbrace{\langle c,1,0\rangle,\ldots,\langle c,1,0\rangle}^{4\text{ 個}},$$

上の 2 つのサマリの比較から分かる通り，C-GK サマリ中の個々の 4 項組は，標準 GK サマリ中の**複数の 3 項組**を表現している．このため，C-GK サマリ中の一つの 4 項組の削除は，標準 GK サマリ中の複数の 3 項組の削除を意味することになる．標準 GK アルゴリズムにおいても，複数の 3 項組の同時同時削除は強い条件下でのみ可能であり，自由に削除した場合には ϵ 近似性を保証することはできない．そこで，C-GK アルゴリズムでも，4 項組の削除は，それが標準 GK サマリ中の一つの 3 項組に対応している場合のみ実行可能といる制約条件を考える．これは 4 項組のカウンタの値が 1 であるか否かにより簡単に判定できる．この制約条件により，制約付き C-GK アルゴリズムの任意実行列は標準 GK アルゴリズムの実行列に変換可能となるので，出力解の ϵ 近似性が保証できることになる．

C-GK アルゴリズムの大枠は Algorithm 1 と同じである．そこでの Insert と Delete を，それぞれ以下の C-Insert と C-Delete へ置き換えればよい．

C-Insert$(e, \mathcal{S})$: サマリ $\mathcal{S}$ 中に $e = e_i$ である e_i が存在する場合には，e_i の 4 項組の中のカウンタを増分する．存在しない場合には，$e_{i-1} \le e < e_i$ となるような最小の e_i を見つけ，$\langle e, 1, 1, g_i + \Delta_i - 1\rangle$ を e_{i-1} と e_i の 4 項組の間に挿入する．但し，$\mathcal{S}$ の先頭もしくは末尾として挿入する際は $\langle e, 1, 1, 0\rangle$ を挿入する．

C-Delete$(\mathcal{S}, N, \epsilon)$: 削除できる 4 項組が $\mathcal{S}$ 中に無くなるまで，以下の削除ステップを繰り返す．

削除ステップ $g_i + g_{i+1} + \Delta_{i+1} \le 2\epsilon N$ かつ $c_i = 1$ である e_i を見つけ，その 4 項組 $\langle e_i, c_i, g_i, \Delta_i\rangle$ を削除し，e_{i+1} の 4 項組を $\langle e_{i+1}, c_i, (g_i + g_{i+1}), \Delta_{i+1}\rangle$ に更新する．

以上の C-GK サマリに記録されているアイテム e_i が近似するランクの下界と上界はそれぞれ次のようになる．

$$\begin{aligned} r_{\min}(e_i) &= \sum_{j=0}^{i-1}(g_j + c_j - 1) + g_i, \\ r_{\max}(e_i) &= r_{\min}(e_i) + (c_i - 1) + \Delta_i \\ &= \sum_{j=0}^{i}(g_j + c_j - 1) + \Delta_i \end{aligned}$$

ストリーム中の ϕ 分位数を答える OutputQuantile は，不等式の判定条件を以下のものに置き換えれば十分である．

$$r_{\min}(e_j) + (g_j + \Delta_j - 1) > \lfloor \phi N \rfloor + \lfloor \epsilon N \rfloor$$

C-GK アルゴリズムに対しては，Theorem 1 を少し弱めた形の以下の定理が成り立つ．

定理 2（計算解の ϵ 近似性の保証） C-GK アルゴリズムにおける OutputQuantile が返す値 e_{j-1} の真のランクを $r(e_{j-1})$ と表記するとき，以下が成り立つ．

$$\lfloor (\phi - \epsilon)N \rfloor \le r(e_{j-1}) \le \lceil (\phi + \epsilon)N \rceil$$

4.2　探索機構の工夫による高速化

ストリームから読み込んだアイテム e を処理するためには，標準 GK および C-GK どちらの場合も，e が新出か否か，即ち e がサマリ内に存在するか否かの判定を最初に行う必要がある．この探索にハッシュ法を用いることで計算量を線形探索の O(N) から，O(1) へ改良することができる．e が新出だった場合には，次は $e_{i-1} < e < e_i$ となる e_i を探索する必要が生じる．この探索は，二分木を用いることにより，その計算量を線形探索の O(N) から O(log$N$) へ改良することができる．以上の効果を性能評価実験により確認する．

5　性能評価実験

表 2 に使用したデータセットの概要を示す．評価実験では，先行研究 [1] を参考にして，ϵ を 0.001 に固定して，サマリのタプル数 $|S|$，実行時間 [s]，最大誤差，平均誤差を計測した．

誤差は，先行研究 [1] に倣い，発生したランク誤差をサマリのタプルの総数で割った商としている．C-GK アルゴリズムは $2\epsilon N$ 未満のランク誤差を保証しているため，最大誤差は必ず $\frac{2\epsilon N}{N} = 0.002$ 未満となる．なお下記の最大誤差と平均誤差の表中の数値の単位は % である．

表 3 の結果から，全てのデータセットおよび探索法において，C-GK アルゴリズムが簡易型 GK よりも高速であることが分かる．ハッシュや 2 分木探索を利用しなくても十分に高速であるという結果が得られた．ハッシュおよび 2 分木探索はサマリのタプル数がどうしても多くなる GK アルゴリズムにおいて効果が良く発揮されていることが確認できる．なお，T.O. は 3600[s] を超えた場合を示している．

タプル数についても全てのデータセットで C-GK サ

表 2　実験データの概要

データセット	データ長	アイテム種類数	種類
chess	118,254	75	密
mushroom	186,854	119	密
connect	2,904,953	129	密
accidents	11,500,872	468	密
retail	908,578	16,470	疎
kosarak	8,019,0177	41,270	疎
pumsbstar	2,475,949	2,088	中間
pumsb	3,629,406	2,113	中間

表 3　実行時間 [s]

	C-GK			簡易型 GK		
		hash	hash b-tree		hash	hash b-tree
chess	0.110	0.410	0.620	13.5	4.36	1.16
mushroom	0.140	0.420	0.700	32.9	6.52	2.11
connect	1.86	1.20	1.05	T.O.	7.50	2.35
accidents	7.47	3.94	2.85	T.O.	14.9	4.76
retail	47.1	9.63	2.21	940	36.2	10.5
kosarak	304	93.4	21.7	T.O.	287	85.2
pumsbstar	15.4	1.42	0.950	T.O.	7.12	2.20
pumsb	26.3	1.65	1.11	T.O.	10.2	3.42

表 4　サマリの大きさ（タプル数）

データセット	C-GK	簡易型 GK
chess	75	23,204
mushroom	118	39.632
connect	129	706,542
accidents	326	3,237,871
retail	10,879	219,536
kosarak	29,003	24,827,551
pumsbstar	1,595	642,098
pumsb	1,623	727,203

マリが圧倒的に小さいことが確認できた．実験に使用した GK アルゴリズムは簡易型であることを考慮しても，カウンタを利用することの効用が良く分かる結果となった．C-GK アルゴリズムを利用した場合，密なデータでは，タプル数はアイテム種類数と同じサイズになった．疎なデータではアイテム種類数よりもタプル数が少なくなったが，これは 1 度しか出現しないアイテムのタプルが削除が行われたためと考えられる．

誤差については，全てのデータの最大誤差が理論保証される 0.2% を下回ることが確認できた．密や中間のデータでは，最大誤差と平均誤差どちらも C-GK アルゴリズムが優るが，疎なデータでは最大誤差と平均誤差の双方が劣る結果となった．これはタプルの削除が多く行

表 5　最大誤差 [%]

データセット	C-GK	簡易型 GK
chess	0.0668	0.0922
mushroom	0.0716	0.0937
connect	0.0853	0.0967
accidents	0.0962	0.128
retail	0.0667	0.0558
kosarak	0.0691	0.0563
pumsbstar	0.0739	0.0875
pumsb	0.0712	0.0886

表 6　平均誤差 [%]

使用データセット	C-GK	簡易型 GK
chess	0.0151	0.0570
mushroom	0.0188	0.0577
connect	0.0245	0.0593
accidents	0.0320	0.0616
retail	0.0529	0.0322
kosarak	0.0556	0.0467
pumsbstar	0.0295	0.0392
pumsb	0.0285	0.0434

われたため，誤差が増加したためと考えられる．

6　まとめと今後の課題

本稿では，カウンタを用いた GK アルゴリズムの計算解の精度保証を与えた．性能評価実験の結果，GK アルゴリズムに比べ大幅な高速化を達成することができた．また，ハッシュを用いることでさらなる高速化が見られた．さらに，タプル数の削減，密・中間のデータセットでは誤差が勝る結果を確認することができた．

今後の課題として，Zhang らによる分割統治法利用する分位数オンライン近似計算法への適用があるカウンタを用いることで冗長な部分を解消でき，より高速なアルゴリズムが開発できると考えられる．

謝辞： 本研究の一部は JSPS 科学研究費補助金 (No.19K12096) の援助を受けている．

参考文献

[1] M. Greenwald and S. Khanna: Space-Efficient Online Computation of Quantile Summaries, SIGMOD, 2001.

[2] Q. Zhang and W. Wang: A Fast Algorithm for Approximate Quantiles in High Speed Data Streams, SSDBM, 2007.

[3] 前田浩丞, 岩沼宏冶, カウンタを用いた ε 近似分位数サマリ構築の高速化に関する研究, 人工知能学会全国大会, 2020/06.