タイトル	An efficient algorithm for mining frequent items from a data stream
著者	TORIYABE, Naoya; KIDA, Takuya
引用	工学研究：北海学園大学大学院工学研究科紀要(21): 39-49
発行日	2021-12-24

研究論文

An efficient algorithm for mining frequent items from a data stream

Naoya TORIYABE* and Takuya KIDA**

Abstract

There are three major counter-based algorithms for mining frequent items from a data stream: Frequent, Lossy counting, and Space saving. Among these, the Frequent algorithm is considered less effective than the others with respect to its precision, because it does not guarantee a margin of error on the estimated frequency of each output item. In this study, we propose a new algorithm (KRB), based on the k-reduced bag concept used in the Frequent algorithm. Our algorithm stores information on the cumulative number of k-reductions, as well as the frequency of each item, upon entry of the item in a counting dictionary. This enables the margin of error to be guaranteed as theoretically better than that of the Space saving algorithm. The amortized time complexity of updating the dictionary is $\mathcal{O}(1)$, and the space complexity for the work space is $\mathcal{O}\left(\frac{1}{\epsilon}\right)$, both of which outperform the Lossy counting algorithm, where ϵ is a given error threshold. Our experimental results confirmed that KRB achieved the same level of speed and accuracy as the current methods.

Key Words *: Frequent Item Mining・Approximate Algorithm・Lossy Counting・Space Saving・Data Mining.*

1 Introduction

The Frequent Items Problem involves identifying all items occurring more than a given threshold from a sequence of items and represents a fundamental problem of computer science with several applications. A number of studies have previously addressed this problem in reference to a data stream [3, 8] as a sequence of items, where each item (of potentially unbounded sequence length) arrives online and, once processed, is discarded or archived [1]. Since data for past items usually cannot be freely accessed, it is necessary to manage information for only candidate items with a limited work space.

There are two typical approaches to this problem. One is the hash-based approach [2, 4, 5], where each item and its frequency are summarized using a data structure comprising hash tables. The other is the counter-based approach [6, 7, 9, 10], where each candidate item is stored together with its (partial or estimated) frequency, which is expected to be above a given threshold.

In this study, we focused on the counter-based approach, for which three major algorithms have previously been proposed: Frequent [6, 7], Lossy counting [9], and Space saving [10]. The Frequent algorithm is straightforward, and requires less memory than the others; however, it shows extremely low performance in precision [3, 8], because it does not guarantee the lower limit on the estimated frequency of each output item.

Here, we propose a new algorithm (KRB) that improves on the Frequent algorithm by enabling

* Currently working at FromSoftware, Inc.
Graduate School of Information Science and Technology (Computer Science and Information Technology), Hokkaido University
e-mail:toriyabe@ist.hokudai.ac.jp
** Graduate School of Engineering (Electronics, Information and Life Science Eng.), Hokkai-Gakuen University
e-mail:kida@hgu.jp

it to guarantee this lower limit. Our algorithm utilizes the k-reduced bag concept similar to the Frequent algorithm while storing information about the cumulative number of delete operations (k-reductions) together with each item newly entered in a counting dictionary, which enables a more precise estimation of the true frequency of the item. Our experimental results confirmed that the proposed algorithm achieves a high degree of precision similar to that of the Lossy counting and Space saving algorithms.

This paper is organized as follows: Section 2 presents related work in this area, defines the problem in terms of a data stream, and offers a brief overview of the Misra-Gries algorithm [11], on which the Frequent algorithm is based; Section 3 presents the KRB algorithm; Section 4 compares KRB with other count-based algorithms; and Section 5 presents the conclusions.

2 Preliminaries

Here, we define basic terms concerning the Frequent Items Problem in relation to a data stream and explain related work. Let $\mathcal{A}$ be a finite set of integers $\{1, 2, ..., m\}$ named *alphabet*, where each element is referred to as an *item*. Data stream $\mathcal{S}$ is a potentially unbounded sequence of items, and we denote a data stream where N items flow as $\mathcal{S}(N)=s_1 s_2 ... s_N$, where $\forall s_t \in \mathcal{A}$. Additionally we denote the true frequency by f_i, and the estimated frequency by $\widehat{f}_i$ for an item i.

2.1 Frequent Items Problem

We define three types of problems below. One that returns exact answers, and others that return less exact answers.

Definition 1 ((Exact) Frequent Items Problem). *Given a stream $\mathcal{S}$ of N items and a support threshold $\phi(0<\phi<1)$, the problem is to return the set of items with a frequency greater than ϕN.*

Since we need to accurately count the frequency of each item in order to solve the Frequent Items Problem for a data stream, this represents a difficult task when the size of the alphabet is large. Therefore, the following Definition 2 allows the inclusion of false-positive answers in the returned set, and Definition 3 provides an additional constraint to false-positive answers with respect to their frequency.

Definition 2 (ϕ-Frequent Items Problem). *Given a stream $\mathcal{S}$ of N items and a support threshold $\phi(0<\phi<1)$, the problem is to return a set satisfying the following conditions: the set (i) should contain all items with a frequency greater than ϕN, and (ii) may contain some other items.*

Definition 3 (ϵ -Approximate Frequent Items Problem). *Given a stream $\mathcal{S}$ of N items, a support threshold ϕ and an error parameter $\epsilon(0<\epsilon<\phi<1)$, the problem is to return a set satisfying the following conditions: the set (i) should contain all items with a frequency greater than ϕN, (ii) does not contain all items with a frequency less than or equal to $(\phi-\epsilon)N$, and (iii) may contain some other items.*

2.2 Hash-List Structure

In the counter-based approach, we store candidate items together with their estimated frequency in a dictionary using a **Hash-List structure** [7, 10], which combines a hash table storing all counters and a doubly linked list of buckets. Each counter is assigned to an item, and each bucket comprises an integer and a doubly linked list of counters, with the associated items having the same frequency as that of the integer. This achieved an $\mathcal{O}(1)$ average time complexity for insert, delete, and search operations and $\mathcal{O}(c)$ space complexity for the work space, where c is the maximum number of counters allowed using this structure. This is likely a similar data structure used in other algorithms addressing this problem.

2.3 Misra-Gries Algorithm (MG)

In 1982, Misra and Gries [11] proposed two

Algorithm 1 First pass of the MG algorithm

Input: an offline data $s_1, ..., s_N$, a threshold $\phi(0<\phi<1)$
Output: a k-reduced bag for a multiset storing N items

```
1: D ← ∅, k ← ⌈1/φ⌉.
2: for each t(1≤t≤N) do
3:   if s_t∈D then
4:     f̂_{s_n} ← f̂_{s_n} + 1
5:   else if |D|<k−1 then
6:     D ← D∪{s_n}, f̂_{s_n} ← 1
7:   else
8:     for each j∈D do
9:       f̂_j ← f̂_j − 1
10:      if f̂_j = 0 then
11:        D ← D\{j}
12:      end if
13:    end for
14:  end if
15: end for
```

algorithms that strictly solve the Frequent Items Problem, given an offline data of N items. The second algorithm (MG) uses a special set (k-reduced bag), which is defined as follows.

Definition 4. *We describe an operation that deletes distinct k items from a multiset as k-reduction. Given a multiset, the remainder obtained is a k-reduced bag for the multiset when we repeat the k-reduction as many times as possible.*

Note that a k-reduced bag depends on the combination of selected items. For example, for a multiset {1, 1, 1, 1, 2, 2, 2, 3, 3, 4}, its 3-reduced bag can be {1, 1, 1, 2} or {1}.

We use a dictionary to express a multiset in the algorithm, with the k-reduction for a dictionary used to decrement distinct k items.

The MG algorithm works in two passes. In the first pass, the dictionary is updated to preclude the necessity for k-reduction every time we access data. The update operations include the following: (i) increment the frequency of the accessed item, (ii) store the accessed item as a new entry, and (iii) decrement the frequency of each item in the dictionary and delete the items having no frequency. This results in a dictionary that describes the k-reduced bag. In the second pass, we scan the data again to calculate the accurate frequency of each item in the k-reduced bag. Algorithm 1 shows the first pass of MG.

The following lemma and theorem describe the maximum number of k-reductions in the first pass of MG and its correctness.

Lemma 1 ([11]). *Given an offline data of N items, the number of k-reductions is at most $\lfloor \frac{N}{k} \rfloor$ occurring in the first pass of the MG algorithm.*

Theorem 1 ([11]). *The MG algorithm strictly solves the Frequent Items Problem.*

Proof. Let the integer $k=\lceil \frac{1}{\phi} \rceil$ with a given support threshold ϕ. In the first pass, the number of k-reductions is at most $\lfloor \frac{N}{k} \rfloor$ from Lemma 1, and the frequency of each item is subtracted only 1 according to k-reduction. Since the upper bound of the total subtracted number is at most ϕN for each item, all items with a frequency greater than ϕN remain in the k-reduced bag. In the second pass, we calculate the true frequency of each item included in the k-reduced bag, resulting in the removal of items with a frequency less than or equal to ϕN. □

2.4 Frequent Algorithm (FR)

The first pass of the MG algorithm was repurposed by Demaine et al. [6] and Karp et al. [7], and referred to as **FR**. These studies show that FR works over a data stream to solve the ϕ-Frequent Items Problem.

Theorem 2 ([6, 7]). *FR solves the ϕ-Frequent Items Problem with $O(1)$ amortized time complexity associated with updating the dictionary and $O(\frac{1}{\phi})$ space complexity for the work space.*

2.5 Related Work

Cormode and Hadjieleftheriou [3] and Liu et al. [8] presented reviews concerning Frequent Item Mining over a data stream.

In the counter-based approach, counters are stored in the data structure, with each counter storing a candidate item and its frequency. The FR in Section. 2.3 stores $k-1$ counters. The Lossy counting algorithm (LC) proposed by Manku and Motwani [9] separates the given data using buckets, each of which has an equal interval. In LC, all items with a frequency less than the confirmed number of buckets are deleted from the dictionary. The improved version of LC (modified LC; mLC) has been reviewed by Liu et al. [8]. The Space saving algorithm (SS) proposed by Metwally et al. [10] uses k counters sorted according to their respective integer. In SS, an item with the lowest frequency of all the items in the dictionary is exchanged for a new item, the frequency of which is one greater than the frequency of the precluded item, if the number of stored items exceeds k when inserting the new item. Both LC and SS have a solution of the ϵ-Approximate Frequent Items Problem.

In the hash-based approach [2, 4, 5], we use a data structure called Sketch comprising hash tables.

3 Algorithms

FR is an algorithm that solves the ϕ-Frequent Items Problem. Previous experiments [3, 8] showed that FR has less precision than the other algorithms. We hypothesized that the precision is dependent on the method of relaxation used in the Frequent Items Problem. In the ϕ-Frequent Items Problem, returned items have no guarantee of error tolerance, which represents the maximum difference between the true frequency and the estimated frequency for a given item. In contrast, the other two algorithms (LC and SS) solve the ϵ-Approximate Frequent Items Problem, because they have a lower margin of error than FR, and items with low frequency are hard to be output.

Both LC and SS return a set comprising the chosen items from the final answer. The ϵ-Frequent Items Problem is a relaxed version of the (exact) Frequent Items Problem. For our algorithm, we established a support parameter $\phi=\epsilon$ in the ϕ-Frequent Items Problem, where ϵ is an error parameter. We describe methods to solve the ϵ-Approximate Frequent Items Problem using a k-reduced bag.

3.1 Modified Frequent Algorithm (mFR)

mFR is a natural extension of FR. To the best of our knowledge, this is the first presentation of mFR, although its theoretical possibility has been addressed previously [8].

The concept for mFR is similar to that for LC and SS. We first solve the ϵ-Frequent Items Problem according to FR, and then choose the items from the answer of the ϵ-Frequent Items Problem. The mFR algorithm is shown in Algorithm 2.

Lemma 2. *Let the true frequency be f_i and the estimated frequency in Algorithm 2 be $\widehat{f}_i$ for an item i. The following inequality holds for each returned item: $f_i-\epsilon N \leq \widehat{f}_i \leq f_i$.*

Proof. Let the integer $k=\lceil\frac{1}{\epsilon}\rceil$ by using an error parameter ϵ. The equation $\widehat{f}_i=f_i$ holds when the k-reduction does not occur; otherwise, $\widehat{f}_i$ is smaller than f_i by the number of k-reductions. Since the number of k-reductions is at most ϵN

Algorithm 2 mFR algorithm

Input: Data stream $\mathcal{S}(N)=s_1 \ldots s_N$, thresholds $\epsilon, \phi (0<\epsilon<\phi<1)$
Output: the solution of the ϵ-Approximate Frequent Items Problem

```
1: D ← ∅, k ← ⌈1/ε⌉.
2: for each t(1≤t≤N) do
3:   if s_t ∈ D then
4:     f̂_{s_n} ← f̂_{s_n} + 1
5:   else if |D| < k−1 then
6:     D ← D ∪ {s_n}, f̂_{s_n} ← 1
7:   else
8:     for each j ∈ D do
9:       f̂_j ← f̂_j − 1
10:      if f̂_j = 0 then
11:        D ← D \ {j}
12:      end if
13:    end for
14:  end if
15: end for
16: output the set {i | f̂_i > (φ−ε)N} from D
```

from Lemma 1, the following inequality holds: $f_i-\epsilon N\leq\widehat{f}_i\leq f_i$. □

Since we underestimate the frequency of the returned items as at most ϵN from Lemmas 1 and 2, we introduce Lemma 3 used for LC.

Lemma 3 ([9]). *Given two parameters, a support threshold ϕ and an error parameter $\epsilon(0<\epsilon<\phi<1)$, the set $\left\{i\,|\,\widehat{f}_i>(\phi-\epsilon)N\right\}$ is the solution of the ϵ-Approximate Frequent Items Problem, when the following inequality holds: $f_i-\epsilon N\leq\widehat{f}_i\leq f_i$.*

Proof. The item is included in the returned set if $f_i>\phi N$ for an item i. The item is not included if $f_i\leq(\phi-\epsilon)N$ for an item i, because the following inequality holds from Lemma 2: $f_i-\epsilon N\leq\widehat{f}_i\leq f_i$. Therefore, the set $\left\{i\,|\,\widehat{f}_i>(\phi-\epsilon)N\right\}$ is a solution of the ϵ-Approximate Frequent Items Problem. □

Theorem 3. *The mFR algorithm solves the ϵ-Approximate Frequent Items Problem with $\mathcal{O}(1)$ amortized time complexity associated with updating the dictionary and $\mathcal{O}\left(\frac{1}{\epsilon}\right)$ space complexity for the work space.*

Proof. The mFR algorithm solves the ϵ-Approximate Frequent Items Problem from Lemmas 2 and 3. This algorithm has three operations: insertion, increment, and k-reduction. Since we use a Hash-List, we are able to perform the previous two operations in $\mathcal{O}(1)$ average time.

Additionally, each k-reduction takes $\mathcal{O}\left(\frac{1}{\epsilon}\right)$ time, and the number of k-reductions is at most ϵN; therefore, we perform the latter operation in $\mathcal{O}(1)$ amortized time. Moreover, the space complexity for the work space is proportional to the number of managed counters; hence, it is $\mathcal{O}\left(\frac{1}{\epsilon}\right)$ space. □

3.2 KRB Algorithm

Here, we propose the **KRB algorithm** to solve the ϵ-Approximate Frequent Items Problem using the k-reduced bag similar to FR and mFR. We set $k=\lceil\frac{1}{\epsilon}\rceil$ and continue to update the dictionary expressing the k-reduced bag while we receive the item from the data stream.

The KRB algorithm counts the cumulative number of k-reductions and stores it in a counting dictionary as additional information along with each item and its estimated frequency. Let the cumulative number of k-reductions (Δ-value) at

Algorithm 3 KRB algorithm

Input: Data stream $\mathcal{S}(N)=s_1 \ldots s_N$, thresholds ϵ, ϕ $(0<\epsilon<\phi<1)$
Output: the solution of the ϵ-Approximate Frequent Items Problem

```
1: D ← ∅, k ← ⌈1/ε⌉, Δ^(0) ← 0.
2: for each t(1 ≤ t ≤ N) do
3:    if s_t ∈ D then
4:       f'_{s_n} ← f'_{s_n} + 1
5:    else if |D| < k − 1 then
6:       D ← D ∪ {s_n}, f'_{s_n} ← Δ^(t) + 1
7:    else
8:       for each j ∈ D do
9:          f'_j ← f'_j − 1
10:         if f'_j = 0 then
11:            D ← D \ {j}
12:         end if
13:      end for
14:      Δ^(t) ← Δ^(t) + 1
15:   end if
16: end for
17: output the set {i | f'_i > φN} from D
```

time t be $\Delta^{(t)}$. In practice, We store an item i and the sum of $\widehat{f}_i$ and $\Delta^{(t)}$ instead of $\widehat{f}_i$ and $\Delta^{(t)}$. We call the sum of $\widehat{f}_i$ and $\Delta^{(t)}$ as the *appended frequency* of i and denote it by f'_i. We set the appended frequency as $f'_i=\Delta^{(t)}+1$, when an item i is newly inserted in the dictionary at time t.

The other operations (incrementing and k-reduction) are performed in the same manner as in FR and mFR. The KRB algorithm is shown in Algorithm 3.

Lemma 4. *Let the true frequency be f_i, the estimated frequency be $\widehat{f}_i$ and the appended frequency in Algorithm 3 be $f'_i=\widehat{f}_i+\Delta^{(t)}$, where $\Delta^{(t)}$ is the Δ-value at time t for an item i. The following inequality holds for each returned item: $f_i \le f'_i \le f_i+\Delta^{(t)}$.*

Proof. Let the integer $k=\lceil \frac{1}{\epsilon} \rceil$ by using an error parameter ϵ. The true frequency f_i is less than or equal to the appended frequency f'_i for an item i in the dictionary, because the true frequency of each item not stored in the dictionary is less than or equal to the Δ-value at any given time. This suggests that the inequality $f_i \le f'_i$ holds for each item i in the dictionary. Let the time at which the item was inserted be t for each item i in the dictionary. The difference between the true frequency f_i and the appended frequency f'_i is at most $\Delta^{(t)}$ (i.e., the Δ-value at time t), resulting in the following equation: $f'_i \le f_i+\Delta^{(t)}$. □

Since we overestimate the frequency of the returned items as at most the Δ-value from Lemma 4, we introduce Lemma 5.

Lemma 5. *Given two parameters, a support threshold ϕ and an error parameter $\epsilon(0<\epsilon<\phi<1)$, the set $\{i \mid f'_i > \phi N\}$ is the solution of the ϵ-Approximate Frequent Items Problem when the following inequality holds: $f_i \le f'_i \le f_i+\Delta^{(t)}$.*

Proof. We assume that the following inequality holds for each item i in the returned set: $f_i \le f'_i \le f_i+\epsilon N$. The item i is included in the returned set if $f_i > \phi N$ for an item i. The item is not included if $f_i \le (\phi-\epsilon)N$ for an item i, because the following inequality holds by the hypothesis $f_i \le f'_i \le f_i+\epsilon N$.

Here, the inequality $f_i \le f'_i \le f_i+\Delta^{(t)} \le f_i+\epsilon N$ holds because the inequality $\Delta^{(t)} \le \epsilon t \le \epsilon N$ holds if the item i is inserted in the dictionary at time t, where $t \le N$. Therefore, the set $\{i \mid f'_i > \phi N\}$ is a

solution of the ϵ-Approximate Frequent Items Problem. □

Theorem 4. *The KRB algorithm solves the ϵ-Approximate Frequent Items Problem with $\mathcal{O}(1)$ amortized time complexity associated with updating the dictionary and $\mathcal{O}\left(\frac{1}{\epsilon}\right)$ space complexity for the work space.*

Proof. The KRB algorithm solves the ϵ-Approximate Frequent Items Problem
from Lemmas 4 and 5, and has the same time and space complexity as mFR. □

3.3 Comparison of Algorithm Performances

We compared the counter-based algorithms addressing the Frequent Items Problem according to time complexity, space complexity, and a margin of error (Table 1). We denoted margin of error by $[a, b]$, which means that the following inequality holds for the true frequency f_i and the estimated frequency e_i for an item i: $f_i+a \leq e_i \leq f_i+b$.

We first compared the algorithms using the k-reduced bag (FR, mFR, and KRB). The results showed the same time complexity as $\mathcal{O}(1)$ amortized time, with the space complexity being proportional to the number of counters owing to the feature of the data structure. Therefore, FR showed less space complexity than mFR and KRB. For the margin of error, FR does not guarantee a lower limit on the estimated frequency of each returned item, and mFR and KRB have the lower and higher limits, respectively. As $\Delta^{(t)} \leq \epsilon N$, the KRB algorithm has a narrower margin of error on the estimated frequency than the mFR algorithm.

We then compared KRB, mLC, and SS. The KRB algorithm showed similar time and space complexity as SS, with both KRB and SS exhibiting better complexities than mLC. For the margin of error, KRB and mLC were at most $\Delta^{(t)}$, and SS was at most ϵN. The $\Delta^{(t)}$ used for the KRB and mLC algorithms was less than or equal to ϵN. Given the case of $\Delta^{(t)}=\epsilon N$, for mLC, $\Delta^{(t)}$ was the number of confirmed buckets, indicating that the equation held in the following case: the item was inserted into the dictionary following confirmation of the most recent bucket. In the KRB algorithm, the equation satisfied all the following conditions: (i) the number of k-reduction was $\Delta^{(t)}$, which is rare; and (ii) an item was inserted between $\Delta^{(t)}-1$ th operation and $\Delta^{(t)}$ th operation. Additionally, the item with a Δ-value of ϵN remained when we returned the answer set in both algorithms. Therefore, most items had a Δ-value lower than ϵN, and the KRB and mLC algorithms had better margins of error than the SS algorithm.

4 Experiment

Here, we compared the KRB algorithm with the following counter-based approaches: FR, mFR,

Table 1. Comparison of time and space complexity, and the margin of error between algorithms.

Name	Time complexity	Space complexity	Margin of error
Frequent [Demaine et.al. 2002]	$\mathcal{O}(1)$ amortized	$\mathcal{O}\left(\frac{1}{\phi}\right)$	$[-\phi N, 0]$
Modified Frequent natural extension of FR	$\mathcal{O}(1)$ amortized	$\mathcal{O}\left(\frac{1}{\epsilon}\right)$	$[-\epsilon N, 0]$
KRB proposed	$\mathcal{O}(1)$ amortized	$\mathcal{O}\left(\frac{1}{\epsilon}\right)$	$[0, \Delta^{(t)}]$
Lossy counting [Manku and Motwani 2002]	$\mathcal{O}(\log \epsilon N)$ amortized	$\mathcal{O}\left(\frac{1}{\epsilon}\log \epsilon N\right)$	$[-\epsilon N, 0]$
Modified Lossy counting [Liu et.al. 2011]	$\mathcal{O}(\log \epsilon N)$ amortized	$\mathcal{O}\left(\frac{1}{\epsilon}\log \epsilon N\right)$	$[0, \Delta^{(t)}]$
Space saving [Metwally et.al. 2005]	$\mathcal{O}(1)$	$\mathcal{O}\left(\frac{1}{\epsilon}\right)$	$[0, \epsilon N]$

mLC, and SS.

FR returns a solution of the ϕ-Frequent Items Problem, and the others returns a solution of the ϵ-Approximate Frequent Items Problem. Our goal was to (i) compare algorithms using the k-reduced bag and (ii) compare solutions from KRB with those from LC, and SS for the ϵ-Approximate Frequent Items Problem.

We implemented our proposed algorithms, FR, SS, and mLC, using a Hash-List data structure. All the algorithms[3] were implemented using C compiled with g ++ (Homebrew GCC 8.2.0) and executed on a Mac Book Pro (MacOSX 10.14.1, Dual-core Intel Core i5 2.9GHz, 8 GB RAM). We used artificial data generated from a Zipf's distribution varying the skew parameter s from 0.0 to 2.0. The data had a size of 2×10^8 and included items chosen from an alphabet of size 10^7. We compared the efficiency of these algorithms with respect to the average time needed for five executions, the memory used for the data structure, and the recall and precision under the experimental conditions.

First, we moved the skew parameter s from 0.0 to 2.0 when we fixed a support threshold $\phi=0.01$. Second, we set the support threshold ϕ to 0.001, 0.005, 0.01, 0.05, and 0.1 on fixing the skew parameter as $s=1.0$. In both cases, the error parameter was set to $\epsilon=0.2\phi$ for each support threshold. The results are shown in Figure 1, where recall and precision are displayed only in cases of computable values because values cannot be calculated when no answer is determined.

A comparison of FR, mFR, and KRB indicated that the speeds of the three algorithms were approximately similar. FR used less memory than the other algorithms, which agreed with our approximation of its theoretical complexity. The three algorithms all guaranteed that false-negative answers were not included in the returned set, and we confirmed a recall of 1 (100 %) for each algorithm. Additionally, mFR and KRB showed much higher precision than FR, which was approximately 0.1 (10 %). Specifically, KRB returned no false-positive answers for each input data.

A comparison of KRB, mLC, and SS indicated approximately the same speed for all the algorithms. We found that mLC used less memory than KRB and SS. Furthermore, the recall and precision values for all three algorithms were 1.0 (100 %).

5 Conclusion

We improved FR to apply a tolerance error to its false-positive answers and proposed a new algorithm (KRB) to solve the ϵ-Approximate Frequent Items Problem. We introduced the k-reduced bag similar to its use in FR and introduced a new parameter (Δ-value) to represent the number of k-reductions. This new algorithm allows the estimation of the true frequency of items in a data stream more precisely.

Our algorithm was theoretically competitive according to complexity measurements with existing algorithms (mLC and SS), with an amortized time complexity for updating the dictionary of $\mathcal{O}(1)$ and a space complexity for the work space of $\mathcal{O}\left(\frac{1}{\epsilon}\right)$, both of which outperformed mLC, where ϵ is a given threshold for error tolerance. Moreover, the margin of error was at most the Δ-value at the time of item insertion into the dictionary, which outperformed SS.

Our experimental results confirmed that the proposed algorithm achieved higher precision than FR and mFR, and displayed similar values as mLC and SS with respect to both speed and precision.

The results suggest that the algorithm presented here can be applied to Frequent Itemset Mining [12].

References

1. Babcock, Brian and Babu, Shivnath and Datar, Mayur and Motwani, Rajeev and Widom, Jennifer.: Models and

[3] https://github.com/naobird/FrequentItemsMining

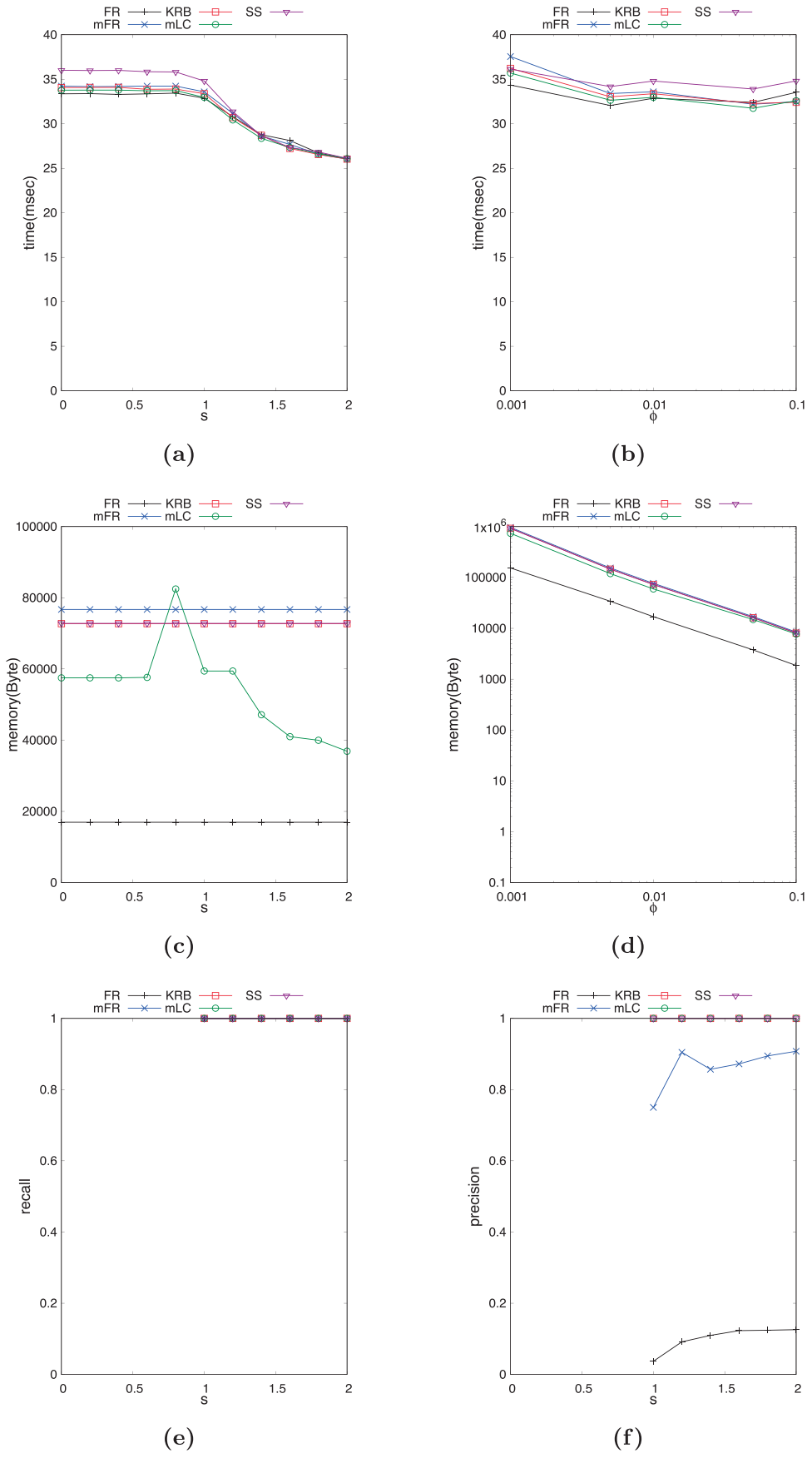

Fig. 1. Performance of algorithms on artificial data. a: Time vs. s. b: Time vs. ϕ. c: Memory vs. s. d: Memory vs. ϕ. e: Recall vs. s. f: Precision vs. s.

issues in data stream systems. In: Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems(PODS), pp. 1–16. ACM (2002)

2. Charikar, Moses and Chen, Kevin and Farach-Colton, Martin.: Finding frequent items in data streams. Theoretical Computer Science **312**(5), 3–15 (2004)

3. Cormode, Graham and Hadjieleftheriou, Marios.: Methods for finding frequent items in data streams. The VLDB Journal **19**(1), 3–20 (2010)

4. Cormode, Graham and Muthukrishnan, Shan.: An improved data stream summary: the count-min sketch and its applications. Journal of Algorithms **55**(1), 58–75 (2005)

5. Cormode, Graham and Muthukrishnan, Shan.: What's hot and what's not: tracking most frequent items dynamically. ACM Transactions on Database Systems (TODS) **30**(1), 249–278 (2005)

6. Demaine, Erik D and López-Ortiz, Alejandro and Munro, J Ian.: Frequency estimation of internet packet streams with limited space. In: European Symposium on Algorithms (ESA), pp. 348–360, Springer, Berlin, Heidelberg (2002)

7. Karp, Richard M and Shenker, Scott and Papadimitriou, Christos H.: A simple algorithm for finding frequent elements in streams and bags. ACM Transactions on Database Systems (TODS) **28**(1), 51–55 (2003)

8. Liu, Hongyan and Lin, Yuan and Han, Jiawei.: Methods for mining frequent items in data streams: an overview. Knowledge and information systems **26**(1), 1–30 (2011)

9. Manku, Gurmeet Singh and Motwani, Rajeev.: Approximate frequency counts over data streams. In: Proceedings of the 28th international conference on Very Large Data Bases(VLDB), pp. 346–357, Elsevier (2002)

10. Metwally, Ahmed and Agrawal, Divyakant and El Abbadi, Amr.: Efficient computation of frequent and top-k elements in data streams. In: International Conference on Database Theory (ICDT), pp. 398–412, Springer, Berlin, Heidelberg (2005)

11. Misra, Jayadev and Gries, David.: Finding repeated elements. Science of computer programming, **2** (2), 143–152 (1982)

12. Yamamoto, Yoshitaka and Iwanuma, Koji and Fukuda, Shoshi.: Resource-oriented approximation for frequent itemset mining from bursty data streams. In: Proceedings of the 2014 ACM SIGMOD international conference on Management of data (SIGMOD), pp. 205–216, ACM, Snowbird (2014)