electronics

MDPI

Article

A New Approach for Anonymizing Transaction Data with Set Values

Soon-Seok Kim

Department of AI Convergence Security, Halla University, Wonju 26464, Republic of Korea; sskim@halla.ac.kr

Abstract: This article proposes a new method that can guarantee strong privacy while minimizing information loss in transactional data composed of a set of each attribute value in a relational database, which is not generally well-known structured data. The proposed scheme adopts the same top-down partitioning algorithm as the existing *k*-anonymity model, using local generalization to optimize safety and CPU execution time. At the same time, the information loss rate, which is a disadvantage of the existing local generalization, is further improved by reallocating transactions through an additional bottom-up tree search process after the partitioning process. Our scheme shows a very fast processing time compared to the HgHs algorithm using generalization and deletion techniques. In terms of information loss, our scheme shows much better performance than any schemes proposed so far, such as the existing local generalization or HgHs algorithm. In order to evaluate the efficiency of our algorithm, the experiment compared its performance with the existing local generalization and the HgHs algorithm, in terms of both execution time and information loss rate. As a result of the experiment, for example, when *k* is 5 in *k*-anonymity for the dataset BMS-WebView-2, the execution time of our scheme is up to 255 times faster than the HgHs algorithm, and with regard to the information loss rate, our method showed a maximum rate of 62.37 times lower than the local generalization algorithm.

Keywords: anonymization; transaction data; set value; de-identification; personal information

Citation: Kim, S.-S. A New Approach for Anonymizing Transaction Data with Set Values. *Electronics* **2023**, *12*, 3047. https://doi.org/10.3390/electronics12143047

Academic Editor: Yu-an Tan

Received: 3 June 2023
Revised: 27 June 2023
Accepted: 10 July 2023
Published: 12 July 2023

1. Introduction

This study set out to deal with the anonymization issue of semi-structured transaction data. Many researchers have proposed solutions to the issue [1]. In 2008, Manolis Terrovitis of Nikos Mamoulis [2–4] observed that when an attacker had partial knowledge about the subsets of items purchased by an individual, the direct disclosure of Database D would make public the identity of a person related to a certain transaction. For instance, Bob purchased a set of items including coffee, bread, brie cheese, diapers, milk, tea, scissors, and light bulbs at a supermarket on a certain day. Bob's neighbor Jim was on the same bus as him and saw some of his items (e.g., brie cheese, scissors, and light bulbs) in the shopping bag. Bob would not want Jim to find out the rest of the items he purchased. If the supermarket decided to make public the transaction information with only one transaction including brie cheese, scissors, and light bulbs, Jim could immediately infer that this transaction was made by Bob and find out the entire content of his shopping bag. A proposed solution to this issue was called k^m-anonymity (Global Generalization). When the attacker has maximum knowledge of the maximum number of items '*m*' in a certain transaction (in a set of transaction records), the attacker will be prevented from distinguishing the number of transaction sets and transactions '*k*' in Database D. Likewise, Database D, for the set of items under '*m*', should include minimum '*k*' number of transactions, including the set. In this example, Jim cannot distinguish Bob's transaction in the minimum five transaction sets in D that has 5^3-anonymity. This definition of anonymity emphasizes the fact that the attacker can identify a person based on some of the sensitive set values about which the attacker has prior knowledge of in the set value data. In some cases, however, it would be

impossible for the attacker to determine in advance the threshold of the number of items in transactions of which he or she has prior knowledge. In such a case, it is impossible to select a safe value for m itself. Another example is when certain transactions can be excluded that are not connected to an individual based on some items of a set. The attacker, for instance, may have knowledge of certain items purchased only by people 65 years or older and other items purchased only by people living in a certain area. In such a case, k^m-anonymity cannot protect individuals from attacks. Puri et al. [5] used a bottom-up strategy based on global generalization to further strengthen k^m-anonymity. Vartika et al. [6] proposed a (k, m, t)-anonymity algorithm to effectively prevent skewness attacks and disclosure of identity and attributes on transactions. This algorithm clusters k records of combinations of m items. Then, it was constructed by adding a threshold value t for the distance between the distributions of sensitive attributes within the cluster. Andrew et al. [7] proposed a greedy heuristic protection method against an attack, in which elected representatives in communication with data owners and collectors cooperate and collude, based on the above clustering.

Y. Xu et al. [8,9] proposed the (h, k, p)-consistency as a more specific privacy criterion to anonymize set value data. The (h, k, p)-consistency guarantees that a database containing insensitive p item combinations includes a minimum k number of transactions, and that a maximum h% of transactions include some sensitive items. In other words, this approach models the attacker's prior knowledge of the parameter p and provides the flexibility of anonymization. k^m-anonymity concerns special cases of h = 100% and $p = m$, but this method has the problem of high rates of information loss by employing suppression technology for all the items whose distribution is relatively small, in order to reinforce safety. The next chapter offers examples of such cases. In addition, this model is based on the assumption that it cannot be directly applied to issues not based on the assumption because there are categories of quasi-identifiers and sensitive information. Cao et al. [10] proposed a ρ-uncertainty model to provide privacy protection from attribute disclosure as opposed to (h, k, p)-consistency. The ρ-uncertainty model limits the likelihood that an individual correlates with a sensitive item that is less than a threshold ρ. Related anonymization techniques include the global generalization-based algorithm and suppression-based algorithm in [10], partial suppression through divide and conquer proposed by Jia et al. [11], and the personalized ρ^m- and (ε, σ)-ρ^m-uncertainty models [12] have also been proposed.

Yeye He and Jeffrey F. Naughton [13] proposed the k-anonymity technique—or the "Local Generalization" technique—to solve the issues raised in [2,4,8,9] above. Based on the "Local Recording" technique defined by the anonymization categorization proposed by [14], this technique provides a definition that k-anonymity will be satisfied if each transaction is the same as a minimum $k - 1$ number of other transactions. Unlike k^m-anonymity, which only protects personal privacy when the attackers have knowledge of items under the number of m, k-anonymity requires no limits to the number of items that the attacker may have knowledge of in the absence of the parameter m. In general, a lower m of k^m-anonymity inevitably means weaker privacy provided by k^m-anonymity. k-anonymity, however, guarantees privacy more strongly than k^m-anonymity. While k^m-anonymity takes a bottom-up approach, k-anonymity uses a top-down greedy partitioning (tree separation) algorithm and, thus, takes much less time to perform CPU functions than the existing techniques [2–4,8,9]. Since k-anonymity takes a top-down partition approach, it has the weakness of huge information loss by applying the same generalization to the domains partitioned under the generalization tree structure of each transaction item, especially the items of unique values. Junqiang Liu and Ke Wang [15] pointed out this disadvantage of k-anonymity and proposed a new technique called the HgHs (Heuristic generalization with Heuristic suppression) algorithm. This technique finds optical heuristic cutting (tree separation) points in a generalization tree structure and applies generalization and suppression techniques to them. Although it guarantees less information loss than the k-anonymity technique, it focuses on CPU performance. As a different approach from the proposed method, Loukides and Gkoulalas-Divanis [16] aimed to achieve k^m-

anonymity by using generalization and suppression through a clustering-based anonymizer. Yao et al. [17] proposed bucketization, which separated the relationship between attributes without modifying the published data by dividing the dataset into non-overlapping subsets. These techniques are commonly referred to as anatomy [18] and slicing [19], and they are often used to anonymize a fixed number of attributes in relational databases. Recently, these disassociation-based techniques [20–28] have been proposed, some of which are applied to electronic health data anonymization [21,22], some of which are improved horizontal partition algorithms [20,26], and some are being considered for improving the k^m-anonymity-based vertical segmentation algorithm [28]. In addition, Refs. [23–25] are being proposed to prepare for the risk of property disclosure that can be experienced with disconnected data. J. Andrew et al. [29] proposed a fixed-spacing approach to protect sensitive medical numeric attributes and an l-diversity slicing approach to protect categories and sensitive attributes. In some studies, it is possible to carry out an analysis using a reconstructed dataset by disconnecting all cells with set values, but this is very time consuming [20].

This study proposes a new technique to reduce information loss to the minimum for transaction data with set values that guarantees strong privacy. The proposed algorithm optimizes security and CPU performance time by adopting a top-down segmentation algorithm such as the existing *k*-anonymity, rather than the anatomy and slicing-based previously discussed. In addition, the disadvantages of the existing *k*-anonymity are improved by reallocating transactions in the additional bottom-up tree search process after top-down partitioning. Compared to the existing *k*-anonymity, bottom-up tree search takes additional time, but the added time is a type of correction work to optimize the information loss of the remaining transactions after the anonymization of the final transaction. This algorithm takes very little time compared with the old *k*-anonymity algorithm for the entire process and cuts down processing time considerably compared with the HgHs technique. In terms of information loss, the proposed technique shows superior performance to other techniques developed so far, including the old *k*-anonymity algorithm and the HgHs technique. Figure 1 and Table 1 shows the outcomes of the proposed algorithm based on the work of Junqiang Liu and Ke Wang [15].

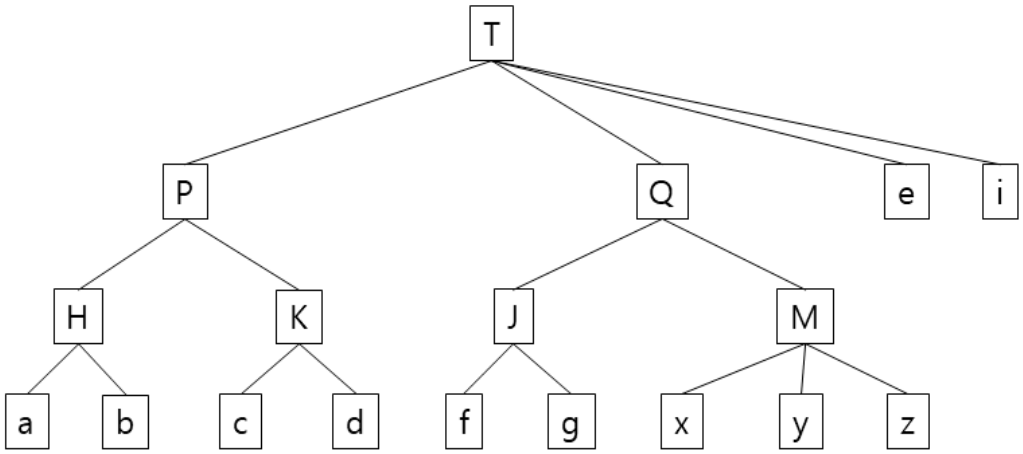


Figure 1. Example of a generalization tree for the anonymization of each item in the transactional Database D. (The values marked on each node mean a kind of item purchased by the customer (for example, when '*a*' is peanuts and '*b*' is walnuts, '*H*' can be seen as nuts.)).

Table 1. Comparison with existing anonymization processing technology in the transactional Database D, composed of each item in Figure 1 (* is for suppression).

TID	Transaction Database D	2^{∞}-Anonymity (Global Generalization) [1,2]	2^{∞}-Anonymity (Suppression) [4,5]	2-Anonymity (Local Generalization) [6]	HgHs [8]	Proposed Algorithm
1	b, c, d	T	*, *, d	T	P	P
2	a, f, g	T	a, f, g	P, f, g	P, f, g	P, f, g
3	d, f, y, z	T	d, f, *, *	K, f, M	P, f, M	K, f, M
4	c, d, f, x	T	*, d, f, *	K, f, M	P, f, M	K, f, M
5	a, b, c, f, g	T	a, *, *, f, g	P, f, g	P, f, g	P, f, g
6	e, i	T	e, *	T	e, *	T, i
7	e	T	e	T	e	T
8	i	T	*	T	*	i

In Table 1, 2^{∞}-anonymity (global generalization) [2,4] is vulnerable to excessive distortions in the presence of outliers. In k^{∞}-anonymity, m represents k^{m}-anonymity, which is the longest transaction length. In the third column in Table 1, for example, all the items are generalized to the top level due to outliers e and I to achieve 2^{∞}-anonymity. Since 2^{∞}-anonymity (suppression) [8,9] or (h, k, p)-consistency uses a technique to suppress all the items, all the occurrences of b, c, i, x, y, and z in the fourth row in Table 1 are suppressed as * indicates. In 2-anonymity (local generalization) [13], which uses a top-down partitioning generalization technique—the same safety indicator as 2^{∞}-anonymity—there is an information loss with b, c, and d generalized into T and with e, i, and e generalized into T in the fifth column in Table 1. HgHs [15] finds optimum heuristic cutting (tree separation) points and applies generalization and suppression techniques to them, having more information loss and a longer performance time than the proposed algorithm. In the sixth column in Table 1, K was generalized to a higher level of P, unlike the proposed algorithm whose T and i generalized into e and * with i suppressed, which suggests that generalizing a value is more useful than suppressing one in terms of information loss. The seventh column in Table 1 shows the performance results of the proposed algorithm. Satisfying the 2-anonymity requirement, it generalized b, c, and d into P rather than T and e and I into T, instead of treating them with T and i rather than e and * and suppressing i compared with local generalization to minimize information loss. These outcomes indicate that the proposed algorithm not only satisfies the 2-anonymity requirement, but it also causes the least information loss compared with older techniques.

In Section 2, we will examine the k-anonymity (local generalization) [13] technique and the HgHs [15] technique as related works. In Section 3, the proposed algorithm is introduced; in Section 4, the improvement points of the algorithm are analyzed; in Section 5, the experimental result of the proposed algorithm is presented; and in Section 6, we will offer conclusions.

2. Related Works

2.1. k-Anonymity (Local Generalization) [13]

The k-anonymity algorithm is used to determine transactions that should be grouped in a generalization hierarchical tree with a top-down greedy partitioning algorithm. Figure 2 and Table 2 generalize all transactions to the root level of the hierarchy. Since all transactions share the same expression ("*ALL*") after being generalized into a root, a single partition always leads to anonymization provided there are at least k number of transactions in the database. The initial partition is conveyed from this starting point to the next Anonymize routine, in which the current partition is made into a sub-partition, with Anonymize called recursively in all the outcome sub-partitions. The dividing process will end when dividing

is no longer possible. Figure 2 and Table 2 is an example of the algorithm proposed in paper [13] and Figure 3 shows the pseudo-codes of the algorithm.

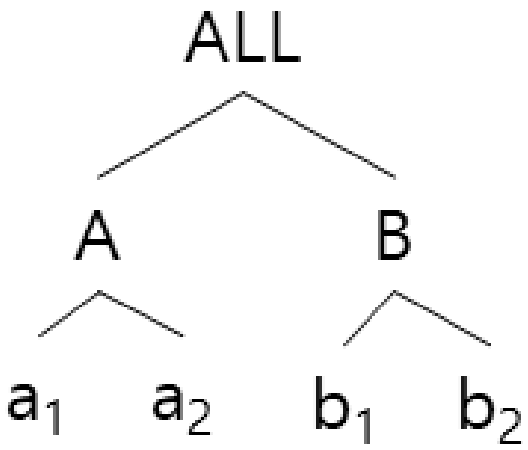


Figure 2. Example of generalization tree in transactional Database D.

Table 2. Example of 2-anonymity (local generalization) according to the generalization tree in Figure 2.

TID	Original Database D	2-Anonymity (Local Generalization)
T1	a_1	A
T2	a_1, a_2	A
T3	b_1, b_2	b_1, b_2
T4	b_1, b_2	b_1, b_2
T5	a_1, a_2, b_2	a_1, a_2, B
T6	a_1, a_2, b_2	a_1, a_2, B
T7	a_1, a_2, b_1, b_2	a_1, a_2, B

The recursive partition algorithm pseudo-codes of set value data in k-anonymity local generalization

```
Anonymize (partition)
if (no further drill down possible for partition) then
   return and put partition in global list of returned partitions
else
   expandNode <- pick node(partition)
   for each data in partition do
      resultPartitions <- distribute_data(data, expandNode)
   end for
   balance partitions(resultPartitions)
   for each subPartition in resultPartitions do
      Anonymize(subPartition)
   end for
end if
```

Figure 3. Pseudo-codes of the k-anonymity (Local Generalization Algorithm) [13].

This algorithm, however, adopts a top-down partition approach, as mentioned in the introduction. It applies the same generalization to the domains partitioned once under the generalization tree structure of each transaction item and items with a unique value, thus causing huge information loss.

2.2. HgHs (Heuristic Generalization Heuristic Suppression) [6]

The basic idea of the proposed technique is to integrate the global generalization technique [2,4] with the technique to suppress all items [8,9] to reinforce k^m-anonymity. Its detailed algorithm has two stages: generalization cuts and suppression scenarios (*SS*s). [Stage 1] searches for the cut of the smallest information loss in the generalization hierarchy tree with an external loop called full subtree generalization. [Stage 2] assesses if items satisfy k^m-anonymity in a subtree within the cut through an internal loop called total item suppression and deletes the items that do not satisfy it. As described in the [15], this algorithm is not efficient for processing large volumes of personal information threats due to its breadth-first search approach. Its CPU performance time is comparatively long, despite smaller information loss than the *k*-anonymity technique.

3. The Proposed Algorithm

The proposed algorithm, which is named *LGR* (Local Generalization and Reallocation), has improved the weaknesses of the old *k*-anonymity technique by adding a bottom-up tree search process after partitioning, and reallocating transactions while optimizing safety and CPU performance time by choosing the same top-down partitioning algorithm as the old *k*-anonymity (local generalization) [13]. Since it employs the top-down partition approach, the old *k*-anonymity technique applies the same generalization to domains partitioned once within the generalization tree structure of each item in transactions, especially for items with unique values. Its weakness comes from huge information loss. The proposed algorithm reallocates transactions in an additional bottom-up tree search process after top-down partitioning, thus improving the weakness of the old *k*-anonymity algorithm.

In Figure 4, when the anonymization algorithm starts, the *Anonymize(partition)* procedure is called first. In this procedure, a generalized tree for the transaction items to be processed is constructed and a partitioning process (tree splitting) is performed. Then, the *Pick_node()* procedure is called. At this time, partitions are sequentially selected from the first partition among the divided partitions, and the partitions selected through the *Expand_node()* procedure are expanded (drilled down) to lower nodes of the tree. In the *Distribute_data()* procedure, each transaction is allocated to the extended partition, and for the transactions allocated through the *Balance_partition()* procedure, the *k*-anonymity privacy model is checked to see whether it is satisfied for the value of *k* given in advance. At this time, if the allocated transactions satisfy *k*-anonymity, the transaction is finalized and the *Final_partition()* procedure is called. Otherwise, it rolls back, calls the *Expand_node()* procedure, and then checks whether *k*-anonymity is satisfied. Even in this case, if there is no satisfy partition, the state is saved and the *Sub_partition()* procedure is called. The *Sub_partition()* procedure calls the *Pick_node()* procedure to select the next divided partition. The above process is repeated for all partitions while continuously drilling down the tree. Finally, the *Final_partition()* procedure minimizes information loss by reprocessing (replacing the partition with the least information loss with a normalized value) for the partitions for which *k*-anonymity is not satisfied until the end through the *Balance_partition()* procedure. The algorithm ends after assigning all transactions to all partitions through the above process.

The proposed algorithm is almost the same as the *k*-anonymity (local generalization) [6] until the *Balance_partition()* routine; however, there are the following differences between them: the proposed algorithm first performs the *FinalQ_data()* routine when there are no more nodes to expand a drill-down or generalization hierarchy tree in the *Balance_partition()* routine, and stores a kind of candidate transaction that is finally generalized and its partition in separate *FinalQ*. When there are no other partitions to allocate the corresponding transactions within the divided partition in a drill-down process through the *Balance_partition()* routine, the proposed algorithm performs the *WaitForQ_data()* routine and stores the final drilled down partition and its transactions in a separate *WFQ*. Then it moves to the next *Final_partition()* routine, in which it searches each partition stored in *WFQ* bottom-up and matches them with the partitions stored in *FinalQ* to additionally check whether there are

partitions that satisfy *k*-anonymity. When there are no partitions that satisfy it in this process, the corresponding transactions and partitions will be moved from *WFQ* to *FinalQ*. When there are partitions that satisfy it, it moves on to the next step. In the last step, the proposed algorithm generalizes the remaining partitions in *WFQ* (only the ones right below the root remain) starting with the partitions of the smallest information loss up to the top-level root. By repeating this until *k*-anonymity is satisfied, the proposed algorithm sends the transactions and partitions of *WFQ* to *FinalQ*, whose transactions will be disclosed outside for the first time. Figure 5 shows the pseudo-codes of the proposed algorithm.

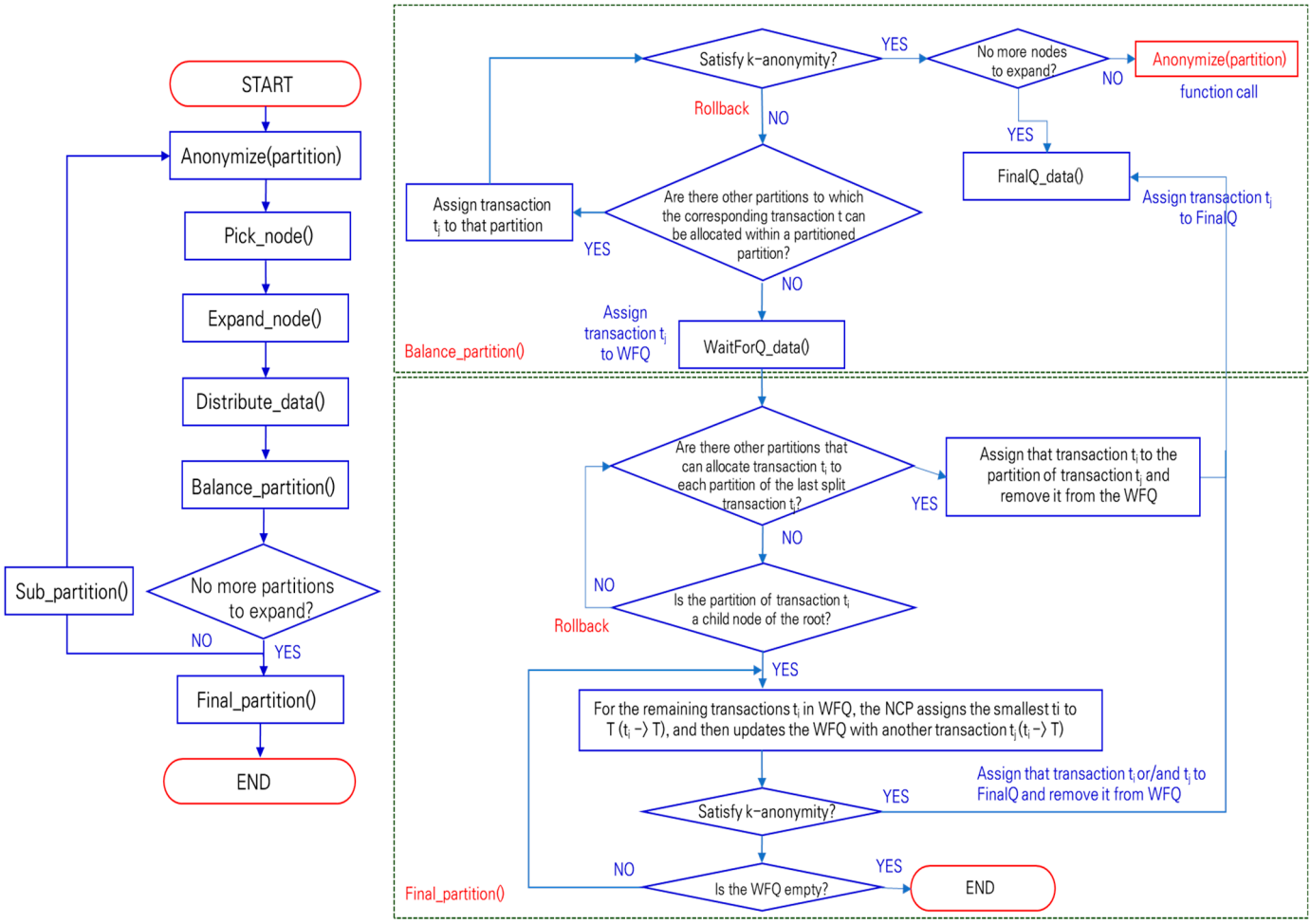


Figure 4. Flow chart of the proposed algorithm.

Figure 6 provides an example of the proposed algorithm being applied to the original Database D in Figure 1 and Table 1. In Figure 6, Rounds 1–6 represent the process of finding an optimal partition that satisfies $k = 2$ anonymity through top-down searches for the roots of a generalization hierarchy tree with the Anonymize (partition) routine in Figure 5. *Round 1* starts at Root *T* of a tree, divides nodes in the class right below *T*, and allocates transactions t1~t8 to each divided partition. *Round 2* repeats top-down searches again, expanding nodes to lower classes. Here, Partitions [P, Q] can choose between [P, J, M] and [H, K, Q] each. *NCP* (Normalized Certainty Penalty), which represents the information loss, is measured in [Q] -> [J, M] and [P] -> [H, K] for the allocated transactions, t2, t3, t4, and t5. It is drilled down (expanded) to the partition with a smaller value or smaller information loss.

```
Anonymize(partition)
if (no further drill down possible for partition) then
  return and put partition in global list of returned partitions
  else
    expandNode <- pick node(partition)
  for each data in partition do
    resultPartitions <- distribute_data(data, expandNode)
  end for
  balance partitions(resultPartitions)
  for each datain resultPartitions do
    WFQ <- WaitForQ_data(data, resultPartition);
    FinalQ<- FinalQ_data(data, resultPartition);
  end for
  for each subPartition in resultPartitions do
    Anonymize(subPartition)
  end for
end if
Final_partition(WFQ, FinalQ);
```

Figure 5. Pseudo-codes of the proposed algorithm.

There are a variety of measurements developed for information loss. The proposed paper [13] used the *NCP* measurement [30] used in k^m-anonymity [2,4]. *NCP* is defined as follows for categorical attributes such as a generalization tree structure:

$$NCP(P) = \begin{Bmatrix} 0, & |u_p| = 1 \\ |u_p|/|I|, & otherwise \end{Bmatrix} \quad (1)$$

where u_p represents the node of an item generalization class where item p (e.g., a_1, a_2, b_1, and b_2 in Figure 2) is generalized. $|u_p|$ and $|I|$ represent the number of leaves in the bottom and overall tree hierarchical structure. In the hierarchical structure in Figure 2, for instance, the information loss NCP(a_1) will be $2/4$ when a_1 is generalized from Transaction t to A. $|u_p|$ has two leaves under A, a_1 and a_2, $|I|$ has all four leaves: a_1, a_2, b_1, and b_2. The NCP for the entire database adds weight to the information loss of each generalized item by using an item-shaped ratio under the influence of all the items of the database. When the total number of Item p occurrences in the database is C_p, the information loss of the entire database due to generalization can be expressed in the following equation:

$$NCP(D) = \frac{\sum_{p \in I} C_p \cdot NCP(p)}{\sum_{p \in I} C_p} \quad (2)$$

The information loss range of a certain generalization (cut) is 0~1 and can be easily measured. For example, the information loss of <{a1, a2} -> A > is $2 \times 0.5 + 3 \times 0.5 + 0 + 0/11 = 2.5/11$ in Table 2. This *NCP* measurement is used in the algorithm proposed in this study.

Pseudo-codes of the proposed algorithm using top-down local generalization and bottom-up generalization hierarchy tree searches

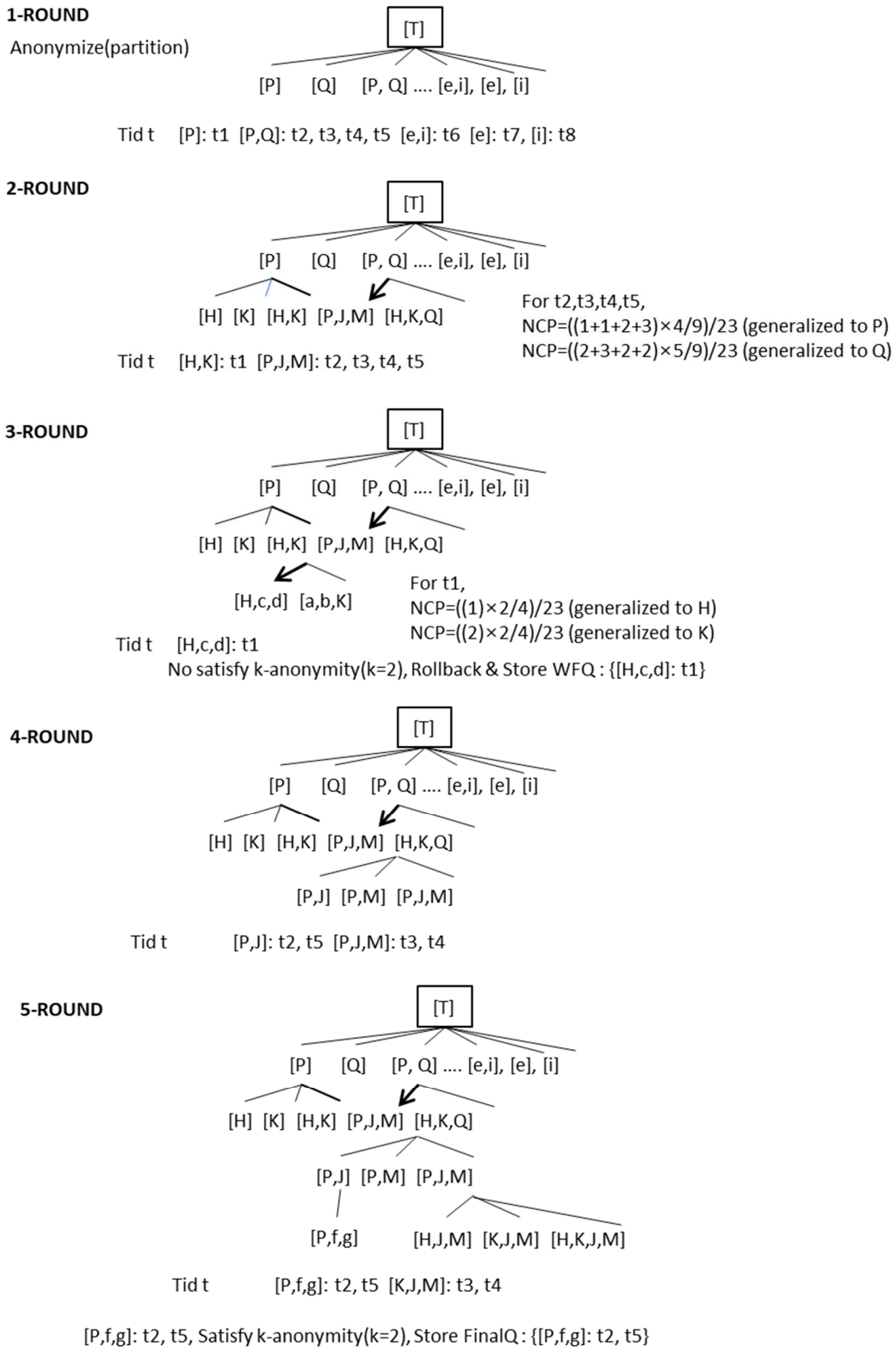
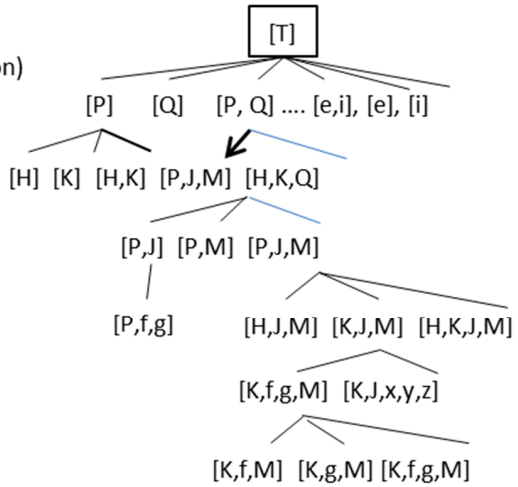


Figure 6. *Cont.*

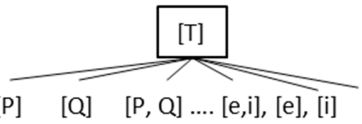
6-ROUND

End Anonymize(partition)



Tid t [K,f,M]: t3, t4, Satisfy k-anonymity(k=2), Store FinalQ : {[K,f,M]: t3, t4}
Anonymize(partition) terminates as there are no more nodes to expand

7-ROUND



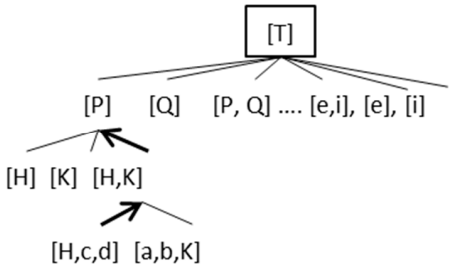
Tid t [P]: t1 [P,Q]: t2, t3, t4, t5 [e,i]: t6 [e]: t7, [i]: t8
In the case of [e,i], [e], and [i], they are all sent to WFQ because they are no longer split and do not satisfy k=2 anonymity

Perform Final_partition()

WFQ : {[H,c,d]: t1, [e,i]: t6, [e]: t7, [i]: t8}
FinalQ : FinalQ : {[P,f,g]: t2, t5, [K,f,M]: t3, t4}

While rolling back each partition of WFQ, search for a transaction that satisfies k=2 anonymity by matching the partition of FinalQ.

8-ROUND



Tid t [H,K]: t1, No satisfy K=2 anonymity, Rollback
[P]: t1, satisfy k=2 anonymity, [P]: Assign t1 to FinalQ and remove from WFQ

WFQ : {[e,i]: t6, [e]: t7, [i]: t8}
FinalQ : FinalQ : {[P,f,g]: t2, t5, [K,f,M]: t3, t4, [P]: t1}

Figure 6. *Cont.*

9-ROUND

1) The remaining transactions in (WFQ: {[e,i]: t6, [e]: t7, [i]: t8}) do not have a partition matching FinalQ
2) Between [e]: t7 and [i]: t8 with the smallest NCP value, [e]:t7 is assigned to T
3) Update WFQ with [T,i] for [e,i] with [e] among other transactions

WFQ : {[T,i]: t6, [T]: t7, [i]: t8}
FinalQ : FinalQ : {[P,f,g]: t2, t5, [K,f,M]: t3, t4, [P]: t1}

10-ROUND

1) Among the partitions in the WFQ, check whether k=2 anonymity is satisfied
2) Partitions {[T,i]: t6, [T]: t7, [i]: t8} all satisfied, assign all these partitions to FinalQ and remove them from WFQ

WFQ : { }
FinalQ : FinalQ : {[P,f,g]: t2, t5, [K,f,M]: t3, t4, [P]: t1, [T,i]: t6, [T]: t7, [i]: t8}

11-ROUND

1) WFQ is empty, so quit and output the value in FinalQ

FinalQ : {[P,f,g]: t2, t5, [K,f,M]: t3, t4, [P]: t1, [T,i]: t6, [T]: t7, [i]: t8}

Figure 6. An example of the proposed algorithm being applied to the original Database D in Figure 1 and Table 1 (the procedure diagram of the proposed algorithm's application by step).

Round 3 repeats the same process as *Round 2*, expanding to [H, K] -> [H, c, d]. The results do not satisfy $k = 2$ anonymity. It rolls back to the upper class, repeating the process to see partitions in the upper class satisfy $k = 2$ anonymity. Since there are no more partitions that satisfy it, Partition [H, c, d] stops expansion and is stored in the waiting queue *WFQ* along with the corresponding transaction t1. *Round 4* sees the resume of expansion for [P, J, M]. The final [P, f, g]: t2, t5 satisfies $k = 2$ anonymity, and is thus stored in *FinalQ*: {[P, f, g]: t2, t5}. *Round 6* repeats the same expansion process as *Round 5*. Partition [K, f, M]: t3, t4 satisfies $k = 2$ anonymity, and is thus stored in *FinalQ*: {[K, f, M]: t3, t4}. Since there are no more nodes to be expanded, the *Anonymize(partition)* routine will end. The *Final_partition()* routine in *Rounds 7–11* goes through two major processes. In the first process, the remaining partitions (and the transactions allocated to them) in *WFQ* for not matching *2*-anonymity roll back bottom-up in a hierarchy tree. They are compared with the partitions in *FinalQ*, and the ones that match *2*-anonymity are moved from *WFQ* to *FianlQ*. Partitions that find no match in this process fail and will move on to the next process. The second process generalizes the partitions (and transactions allocated to them) one by one that are divided no more and remain in *WFQ*, starting with the ones that have the smallest information loss toward an upper class in search for partitions (and the transactions allocated to them) that satisfy $k = 2$ anonymity. The partitions that satisfy it will move from *WFQ* to *FinalQ*. In *Round 7*, the remaining partitions, [e, i], [e], and [i], are divided no more and do not satisfy $k = 2$ anonymity, thus being sent to *WFQ*. In the *Final_partition()* routine, each partition in *WFQ* rolls back and matches a partition in *FinalQ* to find transactions that satisfy $k = 2$ anonymity. *Round 8* allocates Partition [P]: t1 to *FinalQ* and removes it from *WFQ*, since the partition satisfies $k = 2$ anonymity. In *Round 9*, [e]: t7 and [i]: t8 have the lowest *NCP* for the remaining transactions (t6, t7, and t8) in *WFQ*. From these, [e]:t7 is allocated to *T*. From the other transactions, [e, i] containing [e] is updated to [T, i] in WFQ. Round 10 checks whether the remaining partitions ([T, i], [T], and [i]) in *WFQ* satisfy $k = 2$ anonymity. All of the partitions {[T, i]: t6, [T]: t7, [i]: t8} satisfy the $k = 2$ anonymity, and they are all allocated to *FinalQ* and removed from *WFQ*. The final *Round 11* ends the *Final_partition()* routine, since *WFQ* is empty, printing out the values in *FinalQ* and finally disclosing them.

4. Analysis of the Proposed Algorithm

The proposed algorithm employs the *k*-anonymity (local generalization) [8] algorithm as seen in the example in Table 1. It has smaller information loss than the algorithm detailed in reference [13] and the HgHs [15] algorithm, thus providing analysts with greater utility in an analysis of open data. In the case of the existing *k*-anonymity (Local Generalization) [8] algorithm, it adopts a top-down partition approach, resulting in a significant loss of information. This is because the generalization tree structure applies the same generalization only within the partitioned domain once (for items within each transaction), and especially for items with unique values. However, in the proposed algorithm, information loss is minimized by reallocating transactions for all items that do not satisfy *k*-anonymity through an additional bottom-up tree search process after the existing top-down partitioning process. Transactions that do not satisfy *k*-anonymity are stored in a separate *WFQ* after the existing top-down partitioning process through the *Balance_partition()* procedure. Then, the *Final_Partition()* procedure generalizes all transactions to satisfy *k*-anonymity for each transaction in the *WFQ*, and releases the final result. In terms of CPU performance time, the proposed algorithm is much faster than the HgHs [15] that basically employs the breadth-first search approach. The proposed algorithm costs more than the *k*-anonymity (Local Generalization) [13] algorithm, since it goes through an additional *Final_partition()* routine. This process, however, relates to some of the remaining transactions after the completed expansion of a hierarchy tree, which means that the proposed algorithm does not require a very long performance time.

In terms of safety, the proposed algorithm has the same performance as the *k*-anonymity (Local Generalization) [13] algorithm and the HgHs [15] algorithm by similarly meeting the security requirements of *k*-anonymity. As described in the introduction, the proposed algorithm boasts much higher safety than the old 2^{∞}-anonymity (Global Generalization) [2,4] and 2^{∞}-anonymity (suppression) [8,9] algorithms. In other words, *k*-anonymity has no need to limit the number of items that may be exposed to the attacker in the absence of the parameter m, unlike the k^{m}-anonymity, which protects personal privacy only when the attacker has knowledge of the items under the number m. In general, a lower m of k^{m}-anonymity inevitably means weaker privacy provided by k^{m}-anonymity, but *k*-anonymity guarantees stronger privacy than k^{m}-anonymity.

The proposed algorithm is appropriate when quasi-identifiers and sensitive information cannot be distinguished, applying *k* values of *k*-anonymity to the entire transaction. If a given dataset can distinguish quasi-identifiers from sensitive information, it will be possible to additionally apply *l*-diversity [31] while satisfying *k*-anonymity.

5. Experiment Results

5.1. Experiment Environment and Design

This study compared the proposed algorithm with the local generalization method proposed in [13] and the HgHs method proposed in [15] to assess its efficiency by measuring implementation time and information loss or data quality. For information loss, the *NCP* metric, which was defined in [30,32], was applied in the same way as the existing methods in [13,15]. Algorithms were realized with Python. For implementation time, an experiment was conducted with 12th Gen Intel(R) Core(TM) i9-12900KS 3.40 GHz PC with 64GB memory under the Windows 10 environment.

As for data used in the experiment, *BMS-POS* and *BMS-WebView-2* [33] used in [13,15] were used. *BMS-POS* is the sales and transaction log of an electronics retailer with 515,597 for up to 1657 items. Each record holds a maximum of 164 items. An item is categorical data, a type of up to 4-digit product code purchased by a customer, with a distribution ranging from 0 to 1404. For example, if the product code is '1402′, the major category is '1′, the middle category is '4′, the small category is '0′, and the subcategory is '2′. Based on this, we constructed a generalization tree with a maximum depth of 5 (see Figure 7). The experiment used a total of 345,204 left after preprocessing and refinement including data errors.

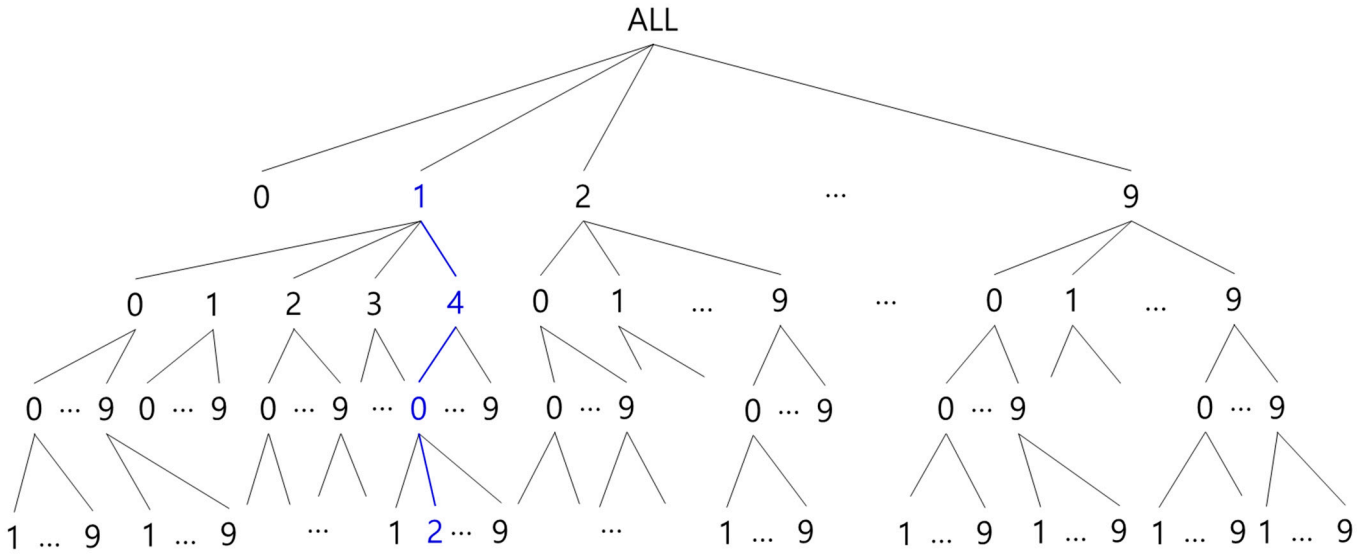


Figure 7. An example of a generalization tree for product code in BMS_POS data (The blue numbers in the figure mean, for example, when the product code is '1402', the major category is '1', the middle category is '4', the small category is '0', and the sub category is '2').

BMS-WebView-2 holds a total of 77,512 transactional data for a maximum of 3340 items. These are months of click stream data collected from an e-commerce website. Just as with BMS-POS, it is a kind of product code purchased by the customer, and the distribution range is 55,267~330,285. Based on this, a generalization tree with a maximum depth of 7 was constructed. Each record holds a maximum of 161 items. The main specifications of the data used in the experiment are shown in Table 3.

Table 3. The main data specifications used in the experiment.

Data Name	Number of Transactions	Maximum Transaction Size	Total Number of Leaf Nodes in the Generalization Tree	Maximum Depth of Generalization Tree
BMS-WebView2	77,512	161	3340	7
BMS-POS	345,204	86	1657	5

Table 4 is an example of some experimental data, and denotes ' | ' as a separator for classification by category.

Table 4. Some examples of original data used in experiments.

BMS-POS	*BMS-WebView-2*
166 | 167	84475 | 84211 | 86919 | 86927 | 86943
168 | 169	56109 | 22699
166 | 175 | 179 | 180 | 181 | 182 | 183 | 184 | 185	55455
194	91795 | 81991
185 | 192 | 193 | 197 | 198 | 199	84947 | 84999

To implement the proposed algorithm, data structures such as the generalization tree, *WFQ*, and *FinalQ*, which were discussed in Figure 7, are required. For this purpose, trees and queues were defined and used. A generalization tree must be prepared in advance, in order to generalize the lower category data entered as transaction data to the upper category. Since there is no fixed number of subcategories that can be linked by classification, the nodes in the generalization tree are implemented such that multiple subnodes can be

linked. *WFQ* is defined as a queue data structure because it requires data to be output in the order in which it was entered. At this time, a circular queue is defined so that input and output can be repeated. Data entered into *FinalQ* is posted to the transaction data in order, so it does not have to follow the characteristics of the Queue to exit in the order entered. Therefore, *FinalQ* does not define a circular queue. Most of the features required by the proposed algorithm are implemented inside the generalization tree class to optimize performance. However, since we need to utilize both the generalization tree class and the circular queue class, we implement the three functions data input, partition, and generalization as separate functions.

- Data input function: Data for constructing the generalization tree and transaction data to be generalized are entered in the anonymization of transaction data. Since these two data are input as a csv file, refer to the csv library for csv file input in Python. The generalization tree is built right at the input stage, and in the case of transactional data, the performance improves the most when running the algorithm after storing it in the generalization tree class.
- Partitioning function: Partitioning is the first step in generalizing transactional data using a generalization tree. In the partitioning process, starting from the root, all combinations consisting of subnodes are found, and among them, only those combinations that can generalize the transaction data are selected and partitioned. This process is repeated from the divided partition to the lowest leaf node, and the partitioning process ends when one or more leaf nodes exist in all partitions. Candidates capable of partitioning are determined through the *k*-anonymity test, and when there are several candidates capable of partitioning, the optimal partitioning direction is determined through a comparison of *NCP*s. If there is only one candidate for indivisibility, divide in that direction.
- Generalization function: Generalization is a function that generalizes transactions that do not satisfy *k*-anonymity when the partitioning process is completed. Generalization was implemented to first calculate the occurrence frequency of each data in the split result data, make it into a dictionary, and roll back the data while checking *k*-anonymity. Rollback is a process of generalizing data corresponding to nodes at a lower level in the generalization tree to higher nodes one by one. The rollback process proceeds recursively until the data goes up to the root node, and ends when it becomes the root node. When the rollback process ends, most transactions satisfy *k*-anonymity, but other transaction data may be rolled back and fail to satisfy *k*-anonymity. To prevent this, the final rollback process is performed once more, and if *k*-anonymity is not satisfied based on the entire data, batch rollback is implemented.

5.2. Experiment Results

Figures 8 and 9 show the experiment results that are in line with the analysis results in Section 4. Figure 8 shows the measurements of the three algorithms in (a) implementation time; and (b) *NCP* information loss rates for each *k* value of *k*-anonymity with *BMS-POS* data. The method [6] using local generalization was the fastest in implementation. The proposed algorithm recorded lower *NCP* information loss rates than the [13,15] algorithms, thus providing superior results. Figure 9 shows comparison results of the three algorithms for each *k* value of *k*-anonymity with the *BMS-WebView-2* data in (a) a detailed implementation time of up to 1000; (b) an implementation time of up to 500,000; (c) *NCP* information loss rates; and (d) measurements of *NCP* information loss rates between the proposed algorithm and the local generalization algorithm [13]. In terms of implementation time, reference [13] was the fastest, but the proposed method was close to it, with a slight difference. The proposed method recorded a relatively faster performance compared with the HgHs [15] method. When *k* of *k*-anonymity was 3 in the real experiment, the proposed method took 411 s, whereas the HgHs [15] method took 468,889 s. When k was 5, the proposed method took 417 s and the HgHs [15] method took 106,338 s. In Figure 8a, a rather peculiar point was that the execution time gradually decreased as the value of *k* increased

in the case of HgHs [15]. Analysis shows that the reason for this is that a lot of time was spent in the initial process of finding the optimal cut with the least information loss through an external loop called 'full subtree generalization'. Afterwards, the process of evaluating whether or not *k*-anonymity is satisfied in the subtree in the cut and deleting items that are not satisfied was judged to be processed very quickly. The proposed algorithm showed relatively much faster performance than the HgHs [15] algorithm, considering that the *k*-anonymity model uses 3, 5, and 10 for *k* values in most actual applications. In terms of *NCP* information loss rates, the proposed method recorded low information loss rates, which were close to those of the [15] method, and also showed outstanding results that were very different from the [13] method. One important point in Figure 9c is that the information loss rate does not change significantly, even when the value of *k* increases in the case of the proposed method, whereas the information loss rate increases in the case of HgHs [8] (when *k* values are 3, 5, 10, 25, and 30, the information loss rates are 0.29, 0.38, 0.5, 0.93, and 2.44, respectively). This is interpreted as the fact that the generalization level does not change, even if the value of *k* increases after the optimal cut is initially set through an external loop called 'full subtree generalization'. Therefore, it was confirmed that the proposed method performed much better than the existing methods [13,15] when considering both the execution time and information loss aspects.

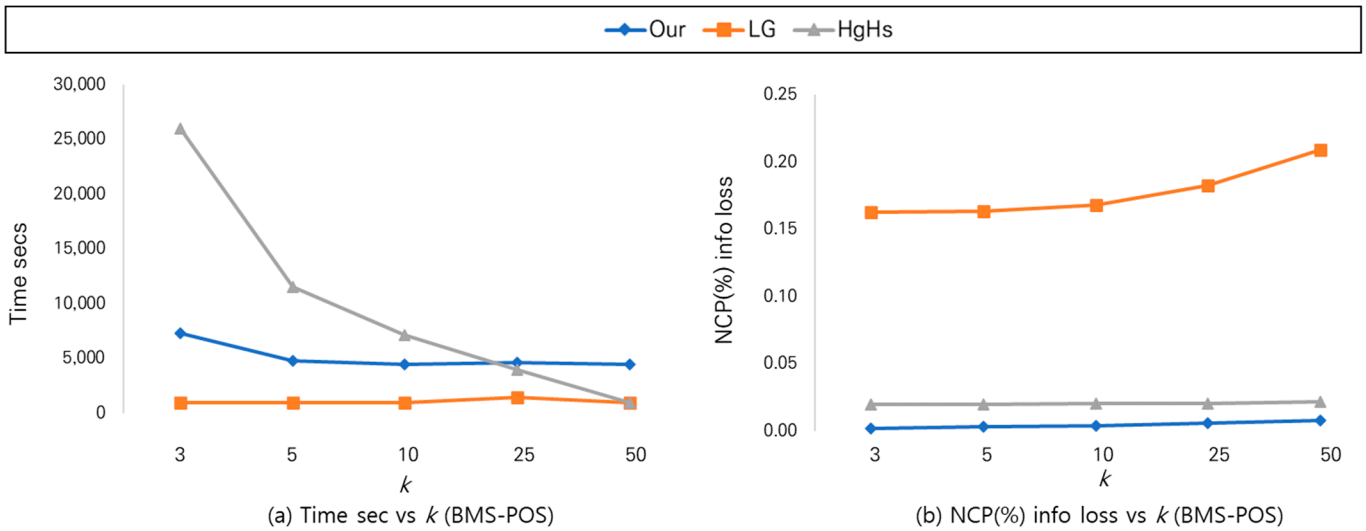


Figure 8. Comparison of the proposed method with the old local generalization [13] method and the HgHs [15] method in implementation time and information loss rates for *BMS-POS* data.

Table 5 shows the results of each processing for the original example data (3-anonymity standard) presented in Table 4. In the case of *BMS_POS* data in Table 5, for example, when the product code is '166', the k value of *k*-anonymity exceeds the criterion 3 set in the test, so the result is '166' without loss of information. In the case of '167', '168', and '169', they are generalized to '16' because they do not satisfy *k*-anonymity. In the case of *BMS-Webview-2* data, 5-digit product code data is mostly generalized to 4-digit data. However, when the number of items increases, such as '82475 | 84211 | 86919 | 86927 | 86943', it is generalized to a two-digit code, that is, '84 | 86 | ', to satisfy 3-anonymity.

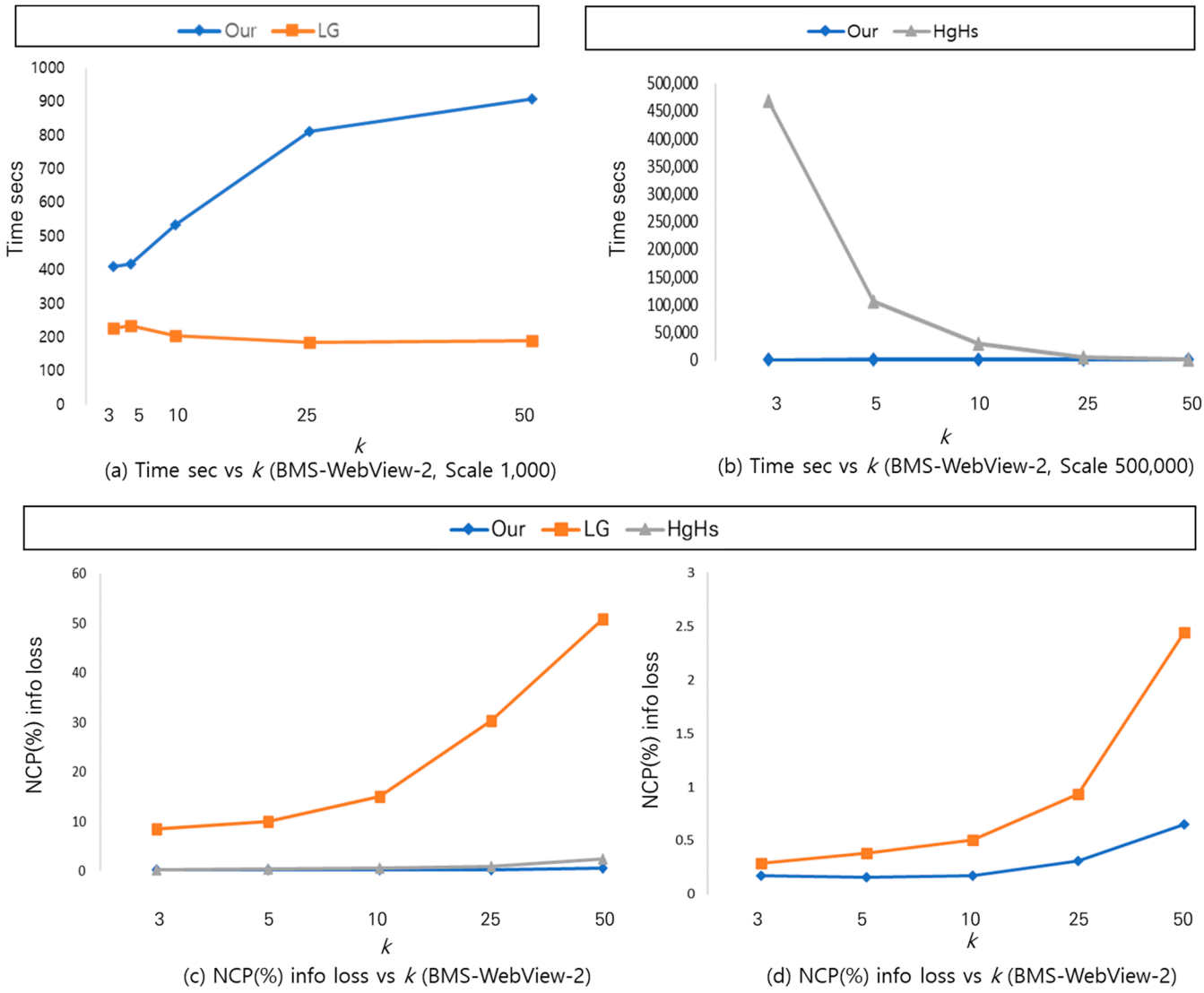


Figure 9. Comparison of the proposed method with the old local generalization [13] method and the HgHs [15] method in implementation time and information loss rates for *BMS-WebView-2* data.

Table 5. Experimental results for some examples of the original data used in the test (when the *k* value of *k*-anonymity is 3).

Data Name	Example of Raw Data	Result of Anonymisation
BMS_POS	166 | 167	16
	168 | 169	16
	166 | 175 | 179 | 180 | 181 | 182 | 183 | 184 | 185	166 | 17 | 18
	194	19
	185 | 192 | 193 | 197 | 198 | 199	18 | 19
Web-View-2	84475 | 84211 | 86919 | 86927 | 86943	84 | 86
	56109 | 22699	5610 | 2269
	55455	5545
	81795 | 81991	8179 | 8199
	84947 | 84999	8494 | 8499

6. Conclusions

The proposed algorithm employs the *k*-anonymity (Local Generalization) algorithm. It has less information loss than the existing algorithms, thus providing analysts with

greater utility in an analysis of open data. In the case of the existing *k*-anonymity (Local Generalization) algorithm, it adopts a top-down partition approach, resulting in a great loss of information. This is because the generalization tree structure applies the same generalization only within the partitioned domain once (for items within each transaction), and especially for items with unique values. However, in the proposed algorithm, information loss is minimized by reallocating transactions for all items that do not satisfy *k*-anonymity through an additional bottom-up tree search process after the existing top-down partitioning process. In terms of CPU performance time, the proposed algorithm is much faster than the HgHs algorithm that basically employs the breadth-first search approach. The proposed algorithm costs more than the *k*-anonymity (Local Generalization) algorithm, since it goes through an additional *Final_partition()* routine. This process, however, relates to some of the remaining transactions after the completed expansion of a hierarchy tree, which means that the proposed algorithm does not require a very long performance time. In terms of security, the proposed algorithm has the same performance as the *k*-anonymity (Local Generalization) algorithm and the HgHs algorithm by similarly meeting the safety requirements of *k*-anonymity. The proposed algorithm boasts much higher safety than the existing 2^{∞}-anonymity (Global Generalization) and 2^{∞}-anonymity (suppression) algorithms. In other words, *k*-anonymity has no need to limit the number of items that may be exposed to the attacker in the absence of the parameter m, unlike the k^{m}-anonymity that protects personal privacy only when the attacker has knowledge of the items under the number m. In general, a lower *m* of k^{m}-anonymity inevitably means weaker privacy provided by k^{m}-anonymity, but k-anonymity guarantees stronger privacy than k^{m}-anonymity. Future work will further expand the proposed algorithm, apply an additional *l*-diversity model, and propose a pseudonymization algorithm that complies with domestic laws and EU GDPR.

Funding: This research was funded by [Personal Information Protection Commission of Korea] grant number [1781000005].

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: [https://kdd.org/kdd-cup/view/kdd-cup-2000] accessed on 9 July 2023.

Acknowledgments: This work was supported by a Korea Internet Security Agency (KISA) grant funded by the Korea government (PIPC) (No. 1781000005, Development of personal information pseudonymization and anonymization processing technology in semi-structured transactions and real-time collected structured data).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cunha, M.; Mendes, R.; Vilela, J.P. A survey of privacy-preserving mechanisms for heterogeneous data types. *Comput. Sci. Rev.* **2021**, *41*, 100403. [CrossRef]
2. Terrovitis, M.; Mamoulis, N.; Kalnis, P. Privacy preserving anonymization of set-valued data. In Proceedings of the VLDB Endowment, Auckland, New Zealand, 24–30 August 2008; pp. 115–125.
3. Terrovitis, M.; Liagouris, J.; Mamoulis, N.; Skiadopoulos, S. Privacy preservation by disassociation. In Proceedings of the VLDB Endowment, Istanbul, Turkey, 27–31 August 2012; pp. 944–955.
4. Terrovitis, M.; Tsitsigkos, D. Amnesia, Institute for the Management of Information Systems. Available online: https://amnesia.openaire.eu/ (accessed on 27 May 2023).
5. Puri, V.; Sachdeva, S.; Kaur, P. Privacy preserving publication of relational and transaction data: Survey on the anonymization of patient data. *Comput. Sci. Rev.* **2019**, *32*, 45–61. [CrossRef]
6. Puri, V.; Kaur, P.; Sachdeva, S. (k, m, t)-anonymity: Enhanced privacy for transactional data. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e7020. [CrossRef]
7. Andrew, J.; Jennifer, E.R.; Karthikeyan, J. An anonymization-based privacy-preserving data collection protocol for digital health data. *Front. Public Health* **2023**, *11*, 1125011. [CrossRef] [PubMed]
8. Xu, Y.; Fung, B.C.M.; Wang, K.; Fu, A.W.C.; Pei, J. Publishing sensitive transactions for itemset utility. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008. [CrossRef]

9. Xu, Y.; Wang, K.; Fu, A.W.; Yu, P.S. Anonymizing transaction databases for publication. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08), Las Vegas, NV, USA, 24–27 August 2008; pp. 767–775.
10. Cao, J.; Karras, P.; Raïssi, C.; Tan, K.-L. ρ-uncertainty: Inference-proof transaction anonymization. In Proceedings of the VLDB Endowment, Singapore, 13–17 September 2010; pp. 1033–1044.
11. Jia, X.; Pan, C.; Xu, X.; Zhu, K.Q.; Lo, E. ρ-uncertainty anonymization by partial suppression. In Proceedings of the International Conference on Database Systems for Advanced Applications, Bali, Indonesia, 21–24 April 2014; pp. 188–202.
12. Nakagawa, T.; Arai, H.; Nakagawa, H. Personalized anonymization for set-valued data by partial suppression. *Trans. Data Priv.* **2018**, *11*, 219–237.
13. He, Y.; Naughton, J. Anonymization of set-valued data via top-down, local generalization. In Proceedings of the VLDB Endowment, Lyon, France, 24–28 August 2009; pp. 934–945.
14. Agrawal, R.; Srikant, R. Privacy-preserving data mining. *ACM SIGMOD Rec.* **2000**, *29*, 439–450. [CrossRef]
15. Liu, J.; Wang, K. Anonymizing transaction data by integrating suppression and generalization. In Proceedings of the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining 2010, Advances in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6118.
16. Loukides, G.; Gkoulalas-Divanis, A. Utility-aware anonymization of diagnosis codes. *IEEE J. Biomed. Health Inform.* **2013**, *17*, 60–70. [CrossRef] [PubMed]
17. Yao, L.; Chen, Z.; Wang, X.; Liu, D.; Wu, G. Sensitive label privacy preservation with anatomization for data publishing. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 904–917. [CrossRef]
18. Xiao, X.; Tao, Y. Anatomy: Simple and effective privacy preservation. In Proceedings of the VLDB Endowment, Seoul, Republic of Korea, 12–15 September 2006; pp. 139–150.
19. Li, T.; Li, N.; Zhang, J.; Molloy, I. Slicing: A new approach to privacy preserving data publishing. *IEEE Trans. Knowl. Data Eng.* **2012**, *24*, 561–574. [CrossRef]
20. Awad, N.; Couchot, J.-F.; Bouna, B.A.; Philippe, L. Publishing anonymized set-valued data via disassociation towards analysis. *Future Internet* **2020**, *12*, 71. [CrossRef]
21. Loukides, G.; Liagouris, J.; Gkoulalas-Divanis, A.; Terrovitis, M. Disassociation for electronic health record privacy. *J. Biomed. Inform.* **2014**, *50*, 46–61. [CrossRef]
22. Loukides, G.; Liagouris, J.; Gkoulalas-Divanis, A.; Terrovitis, M. Utility-constrained electronic health record data publishing through generalization and disassociation. In *Medical Data Privacy Handbook*; Gkoulalas-Divanis, A., Loukides, G., Eds.; Springer: Cham, Switzerland, 2015; pp. 149–177.
23. Sara, B.; Al Bouna, B.; Mohamed, N.; Christophe, G. On the evaluation of the privacy breach in disassociated set-valued datasets. In Proceedings of the 13th International Joint Conference on e-Business and Telecommunications, Lisbon, Portugal, 26–28 July 2016; pp. 318–326.
24. Awad, N.; Al Bouna, B.; Couchot, J.F.; Philippe, L. Safe disassociation of set-valued datasets. *J. Intell. Inf. Syst.* **2019**, *53*, 547–562. [CrossRef]
25. Puri, V.; Kaur, P.; Sachdeva, S. Effective removal of privacy breaches in disassociated transactional datasets. *Arab. J. Sci. Eng.* **2020**, *45*, 3257–3272. [CrossRef]
26. Awad, N.; Couchot, J.F.; Al Bouna, B.; Philippe, L. Ant-driven clustering for utility-aware disassociation of set-valued datasets. In Proceedings of the 23rd International Database Applications and Engineering Symposium, Athens, Greece, 10–12 June 2019; pp. 1–9.
27. Bewong, M.; Liu, J.; Liu, L.; Li, J.; Choo1, K.-K.R. A relative privacy model for effective privacy preservation in transactional data. *Concurr. Comput. Pract. Exp.* **2019**, *31*, e4923. [CrossRef]
28. Liu, X.; Feng, X.; Zhu, Y. Transactional data anonymization for privacy and information preservation via disassociation and local suppression. *Symmetry* **2022**, *14*, 472. [CrossRef]
29. Andrew Onesimu, J.; Karthikeyan, J.; Jennifer, E.; Marc, P.; Hien, D. Privacy preserving attribute-focused anonymization scheme for healthcare data publishing. *IEEE Access* **2022**, *10*, 86979–86997. [CrossRef]
30. Liu, J.; Wang, K. On Optimal Anonymization for l^+-Diversity. In Proceedings of the 2010 IEEE 26th International Conference on Data Engineering, Long Beach, CA, USA, 1–6 March 2010. [CrossRef]
31. Machanavajjhala, A.; Gehrke, J.; Kifer, D.; Venkitasubramaniam, M. L-diversity: Privacy beyond k-anonymity. In Proceedings of the 22nd International Conference on Data Engineering 2006, Atlanta, GA, USA, 3–7 April 2006. [CrossRef]
32. Xu, J.; Wang, W.; Pei, J.; Wang, X.; Shi, B.; Fu, A. Utility-based anonymization using local recoding. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2006, Philadelphia, PA, USA, 20–23 August 2006; pp. 785–790. [CrossRef]
33. Zheng, Z.; Kohavi, R.; Mason, L. Real world performance of association rule algorithms. In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2001, San Francisco, CA, USA, 26–29 August 2001; pp. 401–406. [CrossRef]