

A New Closed High Utility Itemsets with Loose Restriction

Qinghua Zhang¹[0009–0002–4502–974X], Mu-En Wu²[0000–0002–4839–3849],
Chien-Ping Chung³[0000–0002–7953–5571],
Jimmy Ming-Tai Wu^{4*}[0000–0003–3740–2102]

- ¹ College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, China
zhqinghuaa@126.com
- ² Department of Information and Finance Management, National Taipei University of Technology, Taipei, Taiwan
mnasia1@gmail.com
- ³ Department of Information and Finance Management, National Taipei University of Technology, Taipei, Taiwan
thomas6311@mail.ntut.edu.tw
- ⁴ National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan
wmt@wmt35.idv.tw

Abstract. HUIM refers to assisting users in discovering itemsets with high utility, a topic that has garnered significant attention in recent years because these high utility itemsets (HUI) represent higher profit values. The primary advantages of closed high utility itemsets (CHUI) lie in their conciseness and losslessness, signifying that high utility itemsets can be succinctly represented by a compact itemsets without compromising the original high utility itemsets. However, for sparse datasets with low data distribution density, the results of mining CHUI may be suboptimal due to the low repetition rate of their itemsets. Therefore, in this paper, we redefine the concept of CHUI, introduce a loose support restriction, and propose a new definition of loose closed high utility itemsets (LCHUI), which are also concise and lossless. Experiments demonstrate that LCHUI not only enhance the mining efficiency of CHUI in sparse datasets but are also applicable to dense datasets. The LCHUI determine the size of CHUI through a user defined loose-support-threshold, thereby improving the performance of mining CHUI in various datasets.

Keywords: Data mining · Closed high utility itemsets mining · loose closed high utility itemsets.

1 Introduction

For the task of data mining, frequent itemsets mining is a basic and popular data mining task, which is used to discover all the frequent itemsets in the

* Corresponding author

user transaction database, however, the frequent itemsets mining result of a dataset is often huge and not streamlined enough, based on this problem, a concise representation of frequent itemsets, i.e., the concept of the closed mode, is proposed, and the closed itemsets means that a itemsets X does not have its superset that So that the support of X itemsets and its superset is the same. If the closed itemsets is also a frequent itemsets, then it forms a frequent closed itemsets. Frequent itemsets in closed mode are not only very concise, but at the same time they are guaranteed to have no losses.

The frequent itemsets mining process only considers the frequency of occurrence of the itemsets in each transaction transaction, based on the assumption that the importance of each itemsets is equal. However, in real-life applications, the dataset contains not only the frequency of occurrence of itemsets, but also the utility of itemsets and other information, so for retailers, they tend to pay more attention to the value of itemsets, which enables them to adjust their decision-making in a timely manner to obtain more profit value through the mining results [5], so the task of high utility itemsets mining (HUIM) is gaining more and more attention in real-life practice. HUIM refers to finding itemsets from itemsets whose utility values are larger than a minimum utility threshold given by the user. By adjusting the minimum utility threshold, the size of the HUIM results can be changed, and often the high utility itemsets (HUI) are very large, so the process of generating the HUI often implies the consumption of runtime and runtime memory.

In order to represent the HUI more effectively, the mining of closed high utility itemsets (CHUI) applies the closed itemsets which are applicable to the frequent itemsets pattern to the HUI mining process, and the number of mining of CHUI is obviously smaller than the number of mining of HUI, which minimizes the mining cost [9]. However, in some sparse datasets, the number of CHUI mined is close to equal to the number of HUI mined, which means that the current closed high utility itemsets mining (CHUIM) does not perform well in some sparse datasets.

In order to effectively solve the above mentioned limitations of mining CHUI in sparse data sets, this paper redefines the concept of closed itemsets by adding a loose support threshold, which makes the conditions of itemset closure narrowed down, and thus puts forward the loose closed high utility itemsets (LCHUI), It is experimentally proved that the mining of LCHUI can make up for the limitations of CHUI in sparse dataset mining, and is also suitable for further improving the efficiency of mining dense datasets. The main contributions of this paper are as follows:

- (1) Elaboration of a novel definition for LCHUI based on the current research limitations, which mining for CHUI with loose restriction.
- (2) Introduction of the proposed algorithm: LCHUI-Miner algorithm, which mining for LCHUI, the newly defined mining patterns can effectively overcome the constraints of CHUI mining in sparse datasets.

(3) By comparing and analyzing various CHUIM algorithms, it has been demonstrated that the LCHUI-Miner algorithm adjusts the loose support threshold can obtain more concise mining results in different types of datasets.

2 Related Works

In order to solve the problem of large order of magnitude of mining results for HUI, Tseng V S et al. [11] proposed Closed+ High Utility itemset Discovery (CHUD) algorithm, which is the first two-stage algorithm for mining HUI. Then CHUI Miner algorithm [12] is a one-stage algorithm for mining CHUI, which tends to be costly in practice due to his simple pruning strategy. CLS-Miner algorithm [1] is used to mine CHUI efficiently, and he proposes three new construction strategies: chain-EUCP, LBP, pruning by coverage, which reduces the cost of mining to a large extent. The EFIM-Closed algorithm [4] is also a more efficient mining algorithm, and he proposes three strategies to discover CHUIs: closure jumping, forward closure checking and backward closure checking, and the EFIM-Closed algorithm is 71 times faster than the CHUD algorithm. However, the EFIM-Closed algorithm needs to scan the database many times, so it has limited performance in dense datasets, the HMiner-Closed algorithm [10] solves this problem by proposing a new data structure: modified compact utility list is used to minimize the number of times of scanning the database as much as possible. ECUL-Miner [13] is also an algorithm that utilizes the compact utility list structure to mine the CHUIs. The ECUL-Miner is also an algorithm that utilizes the compact utility list structure to mine CHUIs, through the Extended Compact Utility List, a back-checking method to improve the efficiency of mining CHUIs. And there are some algorithms used to mine CHUI in dynamic databases, IncCHUI algorithm [2] introduces a novel algorithm named IncCHUI to mine CHUI efficiently from incremental databases using the incremental utility-list structure. The CHUI-DS algorithm [6] implements CHUI mining in data streams, which uses the concept of a sliding window to consider all the items within the window, thus resulting in a less efficient algorithm, in order to solve this problem the FCHM-Stream algorithm [8] is proposed, this algorithm uses a resultset maintenance strategy based on the skip-list structure within the sliding window, which realizes the fast mining of CHUI within each window.

For rewriting the rules of closed itemsets, C´eline H´ebert and others [7] have proposed a method of mining Frequent δ -Free Patterns in Large Databases. By adding an increment of δ , the rules of closed itemsets are changed, which makes the pruning strategy in the mining process more optimized. In this paper, another evolutionary expression of closed itemsets is obtained by adding loose thresholds, and this study applies closed itemsets with loose restrictions to high utility pattern mining in order to better realize the mining results in different databases.

3 Problem Statement

Suppose D is a transaction database containing a set of n transactions $D = \{T_1, T_2, \dots, T_n\}$, while $I = \{i_1, i_2, \dots, i_m\}$ is a set of distinct items. and the set $X \subseteq I$ is called an itemset, and an itemset is said to be a k -itemset if it contains k items. Each item is associated with an internal utility value IU (quantity) and an external utility value EU (profit), the utility of item x_i in transaction T is denoted as $U(x_i, T)$, and the transaction utility of transaction T_n is denoted as TU_{T_n} . **Table 1** gives an example of a transaction database where the external utility values of items $\{a, b, c, d, e, f, g\}$ are $\{4, 3, 2, 1, 3, 2, 1\}$ as shown in **Table 2**.

Table 1. An example of a transaction database.

TID	Transaction	IU	Utility	TU
T_1	b, d, e	2, 2, 2	6, 2, 6	14
T_2	b, c, e, g	3, 5, 2, 6	9, 10, 6, 6	31
T_3	a, c, d, e	3, 4, 2, 2	12, 8, 2, 6	28
T_4	a, b, c, d, e, f	2, 1, 1, 5, 2, 6	8, 3, 2, 5, 6, 12	36
T_5	a, c, e, g	1, 2, 1, 2	4, 4, 3, 2	13
T_6	a, b, d, e, f	2, 1, 1, 1, 3	8, 3, 1, 3, 6	21
T_7	b, d, e	1, 1, 1	3, 1, 3	7
T_8	a, b, c, e, f	1, 2, 1, 1, 2	4, 6, 2, 3, 4	19

Table 2. An example of a transaction database.

Item	a	b	c	d	e	f	g
EU	4	3	2	1	3	2	1

Definition 1 (High utility). Let be an itemset X . if the utility of an itemset X is greater than the user defined minutility threshold $minutil$, i.e., $U(X) \geq minutil$, then X is a high utility itemset.

For example, it can be derived that $U(a) = U(a, T_3) + U(a, T_4) + U(a, T_5) + U(a, T_6) + U(a, T_8) = 12 + 8 + 4 + 8 + 4 = 36$. If there is $minutil = 30$, $U(a) \geq minutil$, then the set of terms a is a high utility term set. This is followed by a definition of high utility item mining.

Definition 2 (TWU). *The transaction-weighted utility of an itemset X in a transaction database D is called as $twu(X)$ and defined as:*

$$TWU(X) = \sum_{X \subseteq T_n \wedge T_n \in D} TU(T_n). \quad (1)$$

For example, it can be derived that $TWU(a) = T_3 + T_4 + T_5 + T_6 + T_8 = 28 + 36 + 13 + 21 + 19 = 117$. This is followed by a definition of high utility item mining.

Definition 3 (Pruning using the TWU). *Let be an itemset X . If $TWU(X) < minutil$, then X and its supersets are low utility itemsets.*

Definition 4 (Closed high utility itemset [2]). *Let be an itemset X is a high utility itemset, if there exists no proper superset $Y \supset X$ in D such that $Sc(X) = Sc(Y)$, so the itemset X is called closed high utility itemset.*

Definition 5 (Loose support count). *Assuming that the support of the itemset X is $support(X)$, the loose support of the itemset X is $loose\ support(X)$.*

$$loose\ support\ count(X) = Sc(X) \times loose\ support\ threshold \quad (2)$$

Definition 6 (Loose closed high utility itemset). *Let be an itemset X is a high utility itemset, if there exists no proper superset $Y \supset X$ in D such that $loose\ support\ count(X) \leq Sc(Y) \leq Sc(X)$, so the itemset X is called loose closed high utility itemset.*

For example, there exists $minutil = 30$, $loose - support - threshold = 0.5$. The itemset ace exists in T_3, T_4, T_5 and T_8 with support count 4. the utility of the itemset ace : $U(ace) = U(ace, T_3) + U(ace, T_4) + U(ace, T_5) + U(ace, T_8) = 62$, $U(ace) > minutil$, so the itemset ace is a high utility itemset. Among all supersets of the itemset ace , there exists no superset equal to its support count, so the itemset ace is CHUI. However, $loose\ support\ count = 2$ for the itemset ace , there exists the itemset $Y : acbe$ with support count=2, which satisfies the condition of $loose\ support\ count(X) \leq Sc(Y) \leq Sc(X)$, so the itemset ace is not LCHUI.

4 The proposed LCHUI-Miner algorithm

This section mainly introduces the mining algorithm of LCHUI, Loose closed high utility itemsets-Miner (LCHUI-Miner), which redefines the concept of closed itemset, in the algorithm implementation, it is mainly realized by **Algorithm 1**, in which **Algorithm 1** mines the LCHUI the most important two processes i.e., **Algorithm 2** searches for LCHUI and **Algorithm 3** HasBackwardExtension, which changes the precondition of searching for the closed itemsets by increasing the threshold of the loose closure support to achieve the effect of the loose closure of the itemsets in a more succinct way.

Algorithm 1 LCHUI-Miner**Require:**

D , a transaction dataset; $minutil$, user-given minimum utility threshold; $loose - support - threshold$

Ensure:

$LCHUI$: Threshold-constrained set of loose closed high utility itemsets.

```

1: Scan the dataset and calculate the  $twu$  for all  $1 - itemsets$ ;
2: Sort the items in ascending order based on  $twu$ ;
3: Scan the dataset and prune out items for which  $twu(i) < minutil$ ;
4: for each tran in dataset do
5:   if (tran is the same as any of the previous transactions) then
6:     Update the  $iTidCUL$  in  $MCULs$  that contains the items in tran;
7:   else
8:     Insert a new  $iTidCUL$  in  $MCULs$  that contains the items in tran;
9:   end if
10: end for
11: Build the  $EUCS$ ;
12:  $LCHUI = \text{Search-LCHUI}(\emptyset, 1 - MCULs, minutil, loose - support - threshold)$ ;

```

5 Experimental Results

LCHUI-Miner is the algorithm used for mining LCHUI, and its efficacy is demonstrated by experimental results. The experiments described below evaluate whether LCHUI-Miner can effectively address the limitations of current CHUI-related mining algorithms in sparse datasets. The LCHUI-Miner algorithm is implemented using the Java language. The experiments were conducted on a 3.10GHz Intel (R) Core (TM) i5 processor with 8GB of RAM on a PC running the Windows 10 operating system. The experimental dataset is downloaded from the SPMF library [3]. This experiment is conducted using the datasets Retail ($minutil$: 0.04%), Kosarak ($minutil$: 0.8%), Chess ($minutil$: 20%), and Connect ($minutil$: 30%), each exhibiting different characteristics as shown in **Table 3**.

Table 3. Features of the experimental datasets.

Datasets	Transactions	Items	Type
<i>Retail</i>	88162	16470	Sparse
<i>Kosarak</i>	990002	41270	Sparse
<i>Chess</i>	3196	75	Dense
<i>Connect</i>	67557	129	Dense

Fig. 1 represents the experimental results of LCHUI-Miner on sparse dataset, and it can be seen that the number of CHUIs mined by using the HMiner-Closed

Algorithm 2 LCHUI-Miner Procedure: Search-LCHUI**Require:**

P , the itemset prefix; $MCULs$; $minutil$, user-given minimum utility threshold;
 $loose - support - threshold$

Ensure:

$LCHUI$: Threshold-constrained set of loose closed high utility itemsets with P as the prefix.

```

1: for each position  $i$  in  $MCULs$  do
2:    $X = P \cup MCULs[i].item$ ;  $sup = Sup(X)$ ,  $loose-sup = Loose-Sup(X)$ ;
3:   if  $(U(X) + RU(X) \geq minutil$  and HasBackwardExtension $(X, sup, loose -$ 
      $sup, LCHUI)$  then
4:     if  $(MCULs[i].TidList = NULL)$  then
5:        $\beta = X \cup y = MCULs[j].item : j < MCULs.size$  and  $j > i$ ;
6:       if  $(U(\beta) \geq minutil)$  then
7:          $LCHUI \cup \beta$ ;
8:       else
9:          $exMCULs = Construct-MCUL(MCULs, i, minutility)$ ;
10:         $count = |MCUL \in exMCULs : Sup(\beta) \leq Sup(MCUL) \wedge Sup(\beta) \geq$ 
           $loose - Sup(MCUL)|$ ;
11:        if  $(count = |MCULs| - i)$  then
12:          //closure jumping;
13:          if  $(U(\beta) \geq minutil)$  then
14:             $LCHUI = LCHUI \cup \beta$ ;
15:          else
16:            if  $(count = 0$  and  $U(\beta) \geq minutil)$  then
17:               $LCHUI = LCHUI \cup \beta$ ;
18:               $Search-LCHUI(X, exMCULs, minutil, loose - support -$ 
                 $threshold)$ ;
19:            end if
20:          end if
21:        end if
22:      end if
23:    end if
24:  end if
25: end for

```

mining algorithm is not more concise compared to the original number of HUIs, and it can even be said that there is not much significant change in the number of mined. And through the mining results using the LCHUI-Miner algorithm, it is easy to see that due to the addition of the loose support threshold, it can effectively make the set of LCHUI more concise. The three sets of comparison data for each bar comparison plot are the comparison of the number of digs for HUI, CHUI, and LCHUI with loose support threshold = 0.8, loose support threshold = 0.7, and loose support threshold = 0.6, respectively.

Fig. 2 shows the mining performance of LCHUI-Miner algorithm in dense dataset, as shown in the figure, HMiner-Closed algorithm is able to mine CHUI

Algorithm 3 LCHUI-Miner Procedure:HasBackwardExtension**Require:**

X , a candidate; $sup(X)$, the support for X ; $loose - sup(X)$, the loose-support for X ; $LCHUI$.

Ensure:

whether or not X has a backward extension.

```

1:  $k = |X|$ ;
2:  $n = LCHUI.length$ ;
3: if ( $k \geq n$ ) then
4:   return false;
5: end if
6: for  $i = k + 1$  to  $n - 1$  do
7:    $vt = -1$ ;
8:    $vt = \text{Binary search in } LCHUI[i] \text{ based on the support}$ ;
9:   if ( $vt \neq -1$ ) then
10:     $prev = vt$ 
11:    while ( $prev \geq 0 \wedge sup(LCHUI[i].itemset[prev]) \geq loose - sup \wedge$ 
12:       $sup(LCHUI[i].itemset[prev]) \leq sup$ ) do
13:      if ( $LCHUI[i].itemset[prev]$  contains  $X$ ) then
14:        return true;
15:      else
16:         $prev --$ ;
17:      end if
18:    end while
19:     $next = vt + 1$ ;
20:    while ( $next \leq n \wedge sup(LCHUI[i].itemset[prev]) \geq loose - sup \wedge$ 
21:       $sup(LCHUI[i].itemset[prev]) \leq sup$ ) do
22:      if ( $LCHUI[i].itemset[prev]$  contains  $X$ ) then
23:        return true;
24:      else
25:         $prev ++$ ;
26:      end if
27:    end while
28:  end if
29: end for
30: return false;

```

in dense dataset very efficiently and concisely, but at the same time, LCHUI-Miner's loose support threshold can be valued between 0-1, then it will be able to be more concise in the mining results of CHUI by adjusting the threshold in order to achieve a different degree of simplicity of the item set. The three sets of comparison data for each bar comparison plot are the number of digs for HUI, CHUI, and LCHUI with loose support threshold = 1, loose support threshold = 0.98, and loose support threshold = 0.95, respectively. When the loose support threshold is 1, the concept of LCHUI is the same as that of CHUI, and the number of excavations is also the same.

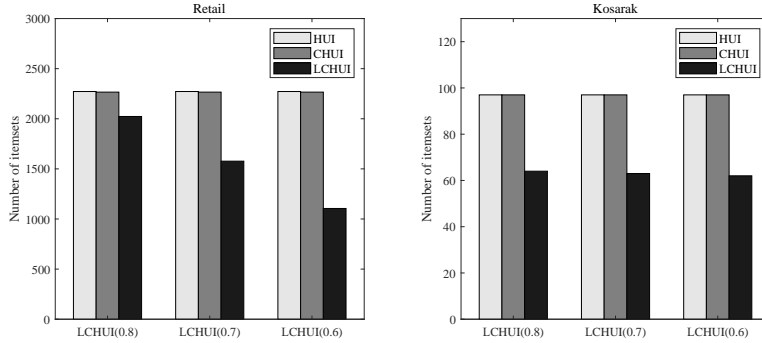


Fig. 1. Comparison of the number of mined HUI, CHUI, and LCHUI with different loose support thresholds in sparse datasets.

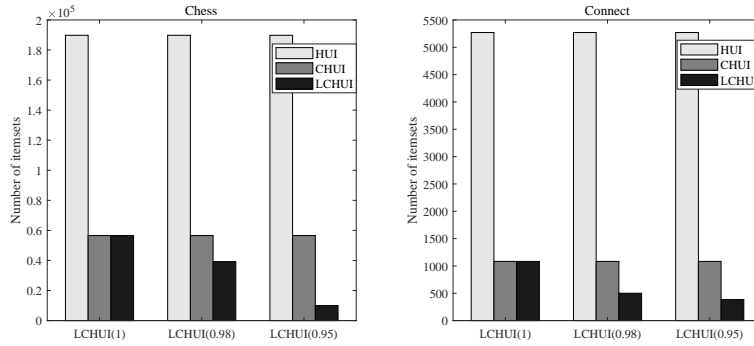


Fig. 2. Comparison of the number of mined HUI, CHUI, and LCHUI with different loose support thresholds in dense datasets.

6 Conclusion

With the advent of the information age, the explosive growth of complex data inevitably dazzle people, sparse data as one of them, does not mean that these data scattered useless, through some scientific and reasonable means of mining will be able to find useful information from these sparse data, and enable the user to obtain a higher profit value from it. In this study, we define the concept of LCHUI and propose the LCHUI-Miner algorithm to mine LCHUI efficiently, and prove that the application of the loose closure support threshold makes the mining result of LCHUI is more concise than that of CHUI under the same situation, and improves the algorithm's running time, which well solves the mining of CHUIM in sparse data set limitations, better apply CHUI to different types of static datasets.

In our future work, we hope to apply the concept of LCHUI more deeply to data mining applications, such as to dynamic datasets, to bring practical value to production and life.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Dam, T.L., Li, K., Fournier-Viger, P., Duong, Q.H.: Cls-miner: efficient and effective closed high-utility itemset mining. *Frontiers of Computer Science* **13**(2), 357–381 (2019). <https://doi.org/10.1007/s11704-016-6245-4>
2. Dam, T.L., Ramampiaro, H., Nørnvåg, K., Duong, Q.H.: Towards efficiently mining closed high utility itemsets from incremental databases. *Knowledge-Based Systems* **165**, 13–29 (2019). <https://doi.org/10.1016/j.knosys.2018.11.019>
3. Fournier-Viger, P., Lin, J.W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H.: An open-source data mining library (2016), <https://www.philippe-fournier-viger.com/spmf/index.php?link=download.php>
4. Fournier-Viger, P., Zida, S., Lin, J.C.W., Wu, C.W., Tseng, V.S.: Efim-closed: Fast and memory efficient discovery of closed high-utility itemsets. In: *Machine Learning and Data Mining in Pattern Recognition*. pp. 199–213 (2016). https://doi.org/10.1007/978-3-319-41920-6_15
5. Han, M., Cheng, H., Zhang, N., Li, X., Wang, L.: An efficient algorithm for mining closed high utility itemsets over data streams with one dataset scan. *Knowledge and Information Systems* **65**(1), 207–240 (2023). <https://doi.org/10.1007/s10115-022-01763-9>
6. Haodong, C., Meng, H., Ni, Z., Xiaojuan, L., Le, W.: Closed high utility itemsets mining over data stream based on sliding window model. *Knowledge-Based Systems* **58**(11), 2500–2514 (2021). <https://doi.org/10.7544/issn1000-1239.2021.20200554>
7. Hébert, C., Crémilleux, B.: Mining frequent δ -free patterns in large databases. vol. 3735, pp. 124–136 (2005). https://doi.org/10.1007/11563983_12
8. Li, M., Han, M., Chen, Z., Wu, H., Zhang, X.: Fchm-stream: fast closed high utility itemsets mining over data streams. *Knowledge and Information Systems* **65**(6), 2509–2539 (2023). <https://doi.org/10.1007/s10115-023-01831-8>
9. Lin, J.C.W., Djenouri, Y., Srivastava, G., Fournier-Viger, P.: Efficient evolutionary computation model of closed high-utility itemset mining. *Applied Intelligence* **52**(9), 10604–10616 (2022). <https://doi.org/10.1007/s10489-021-03134-3>
10. Nguyen, L.T., Vu, V.V., Lam, M.T., Duong, T.T., Manh, L.T., Nguyen, T.T., Vo, B., Fujita, H.: An efficient method for mining high utility closed itemsets. *Information Sciences* **495**, 78–99 (2019). <https://doi.org/10.1016/j.ins.2019.05.006>
11. Tseng, V.S., Wu, C.W., Fournier-Viger, P., Yu, P.S.: Efficient algorithms for mining the concise and lossless representation of high utility itemsets. *IEEE Transactions on Knowledge and Data Engineering* **27**(3), 726–739 (2015). <https://doi.org/10.1109/TKDE.2014.2345377>
12. Wu, C.W., Fournier-Viger, P., Gu, J.Y., Tseng, V.S.: Mining closed+ high utility itemsets without candidate generation. In: *2015 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. pp. 187–194 (2015). <https://doi.org/10.1109/TAAI.2015.7407089>

13. Yue, Z., Qiyun, X., Lin, L., Lijuan, W.: Ecul-miner: Efficiently mining high utility closed itemsets. In: 2021 33rd Chinese Control and Decision Conference (CCDC). pp. 2337–2341 (2021). <https://doi.org/10.1109/CCDC52312.2021.9601880>