Applied Intelligence (2021) 51:8629–8640
https://doi.org/10.1007/s10489-021-02357-8

Mining colossal patterns with length constraints

Tuong Le[1,2] · Thanh-Long Nguyen[3] · Bao Huynh[4] · Hung Nguyen[4] · Tzung-Pei Hong[5,6] · Vaclav Snasel[7]

Accepted: 13 March 2021 / Published online: 9 April 2021

Abstract
Mining of colossal patterns is used to mine patterns in databases with many attributes and values, but the number of instances in each database is small. Although many efficient approaches for extracting colossal patterns have been proposed, they cannot be applied to colossal pattern mining with constraints. In this paper, we solve the challenge of extracting colossal patterns with length constraints. Firstly, we describe the problems of min-length constraint and max-length constraint and combine them with length constraints. After that, we evolve a proposal for efficiently truncating candidates in the mining process and another one for fast checking of candidates. Based on these properties, we offer the mining algorithm of Length Constraints for Colossal Pattern (LCCP) to extract colossal patterns with length constraints. Experiments are also conducted to show the effectiveness of the proposed LCCP algorithm with a comparison to some other ones.

Keywords Colossal pattern · Data mining · High-dimensional database · Length constraints

1 Introduction

Frequent pattern mining (FPM) is an essential step in association rule mining [1], correlation mining [2], sequential pattern mining [3, 4], episode mining [5], classification [6], and clustering [7]. FPM has been widely applied in many sectors, such as market analysis [8], weblog analysis [9], and prediction [10]. It is not only limited to market basket analysis but also applied in other fields, especially in bioinformatics, such as DNA sequence analysis [11], gene expression [12], and protein relations complexes [13].

In recent years, many effective algorithms have been developed to resolve the challenge of pattern mining, and interest in this problem remains [14, 15], specifically in today's era of vast automatic data gathering when a new kind of database has been recognized, called high-dimensional data, which is classified by a relatively smaller number of rows compared to columns [16].

The main reason for the high level of attention paid to FPM algorithms is the high computational cost. The search area of FPM is exponential to the dimensionality of the transactions in the database. This presents challenges for itemset production when the support levels are small.

✉ Bao Huynh
hq.bao@hutech.edu.vn

Tuong Le
lecungtuong@tdtu.edu.vn

Thanh-Long Nguyen
longnt@hufi.edu.vn

Hung Nguyen
nm.hung@hutech.edu.vn

Tzung-Pei Hong
tphong@nuk.edu.tw

Vaclav Snasel
vaclav.snasel@vsb.cz

[1] Informetrics Research Group, Ton Duc Thang University, Ho Chi Minh City 700000, Vietnam

[2] Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City 700000, Vietnam

[3] Faculty of Information Technology, Ho Chi Minh City University of Food Industry, Ho Chi Minh City 700000, Vietnam

[4] Faculty of Information Technology, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City 700000, Vietnam

[5] Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung 811, Taiwan

[6] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 804, Taiwan

[7] Faculty of Electrical Engineering and Computer Science, VSB-Technical University of Ostrava, 70800 Ostrava, Czech Republic

In addition, some mining tasks in bioinformatics have generated many small frequent patterns. Regrettably, the patterns in larger sizes are more meaningful than those in shorter ones. These large patterns with lots of various items are called colossal patterns [17]. As such, it is ineffective to execute a mining algorithm that will find all sizes of frequent patterns when the ones needed are only the colossal ones. The terms mining and extracting in the paper are used interchangeably.

Some efficient algorithms to extract colossal patterns have been proposed, such as the Pattern-Fusion [17], the CPM (Colossal Pattern Miner) [18], BVBUC [16], Doubleton Pattern Mining (DPM) [19], DPMine [20], CP-Miner and PCP-Miner [21] approaches.

In many problems, users need to enter constraints to mine the patterns of interest, which accelerates the mining process, limits the number of results, and reduces the computing cost. Hence, mining with length constraints has attracted many researchers in recent years. Many methods exist for mining frequent patterns with pattern constraints [22, 23], class association rules with itemset constraints [24, 25] and class constraints [26], erasable itemsets with subset and superset constraints [27], and colossal patterns with itemset constraints [28]. The top-down strategy is suitable for pruning non-colossal patterns; however, it cannot be applied to colossal pattern mining with pattern constraints.

Since colossal pattern mining is extremely intensive, some pruning strategies based on constraints are applied to reduce computing cost. For example, PCP-Miner-Post1 and PCP-Miner-Post2 can eliminate colossal patterns that do not satisfy the length constraints in the mining processing, but their efficiency is not good enough.

The colossal pattern mining with length constraints is useful to accelerate the mining process and eliminate the number of redundant patterns. However, it is difficult to remove colossal patterns that do not satisfy the length constraints because it is hard to determine the right length.

This paper proposes an efficient technique for extracting colossal patterns with length con-straints (min-length and max-length). Firstly, we state the problem of extracting colossal patterns with length constraints. Next, theorems based on min-length and max-length are developed for the early pruning of candidates. Based on this, a length constraint-based technique for quickly extracting colossal patterns is developed. Finally, the proposed method is evaluated compared to the PCP-Miner-Post1 and PCP-Miner-Post2 methods in mining time and memory usage on various real-life databases.

The paper is structured as follows. Section 2 elaborates on some related problems, including FPM, FPM with constraints, colossal-pattern mining, and colossal-pattern mining with itemset constraints. The foundational concepts and problem statements are introduced in Section 3. In Section 4, new theorems are suggested to quickly eliminate candidates based on length constraints. An effective algorithm for mining colossal patterns with length constraints is also developed in this section. This proposed algorithm is compared to some other related methods in the following content, Section 5. The conclusions and future development trends in this area are presented in the last section.

2 Related works

Colossal patterns are useful in many applications, especially in bioinformatics fields. They were first proposed in Pattern-Fusion [17] by combining the core-patterns with a heuristic to estimate the colossal patterns by only extracting the top-K frequent patterns. From that, some methods were developed, such as the Colossal Pattern Miner (CPM) [18], which chooses the seed patterns in a smart way by separating sub-patterns of overlapping colossal patterns based on their frequencies. Doubleton Pattern Mining (DPM) [19] uses the vector intersection operator and a data structure named D-struct to find all colossal patterns in a biological database, with better performance than the CPM. Besides, an improved version of DPM called DPMine [20] was proposed based on a DPT+ tree to find Doubleton patterns. This algorithm's weakness is that a large number of mined itemsets is obtained when the *minSup* is small. Next, BVBUC [16], applying a bottom-up scheme for extracting all colossal patterns, was proposed. Although BVBUC solves some disadvantages of the previous methods, it still has a few limitations, so the next two methods, named CP-Miner and PCP-Miner [21], were proposed. CP-Miner applied early pruning methods by relying on CP-Tree to mine colossal patterns, including pruning the transactions and removing non-colossal pattern nodes, while PCP-Miner, an improved version of CP-Miner, has a sorting strategy to efficiently eliminate candidate non-colossal patterns.

Because colossal pattern mining has generated lots of redundant patterns, mining the Frequent Colossal Closed Itemsets (FCCI) has been the focus of attention recently. Some approaches for mining colossal closed patterns have thus been proposed, such as DisClose [29], based on a Compact Row-Tree (CRTree) structure with a bottom-up strategy to mine colossal closed patterns. The DSFCCIM [30] uses the Effective Improvised Pre-processing (EIP) technique and integrates a novel Rowset Cardinality Table, an Itemset Support Table, to mine FCCI, pruning the complete set of unrelated features and unrelated rows. The RARE [31] is based on the breadth-first bottom-up strategy to mine colossal closed patterns. Moreover, the DREFCCIM [32] is the first distributed method to find FCCI.

Although FCCI mining can eliminate redundant patterns for many applications in the real world, the users are only interested in a small set of FPs that satisfy certain conditions. For example, when analyzing the DNA sequences, there are many patterns in the data, but the analysts are only concerned

Table 1 A transaction database

TID	Items
1	A J K L
2	B C D E G H J
3	B C E F G H J
4	B C D E F G H J
5	B E F G H J
6	B H K
7	B D F G H
8	E F G H J
9	F G H
10	B C D E G H
11	C F K
12	A G K L

with patterns containing a number of specific items, so mining patterns with constraints is a good choice.

Many strategies for mining FPs with constraints have been developed. The first strategy is called post-processing and relies on Apriori, Eclat or FP-Growth to mine all the FPs that satisfy the constraint; however, this strategy requires a lot of time to produce and check candidates. The next strategy is called pre-processing, and first removes all the entries that do not satisfy the constraint and then mines all the FPs from the rest of the database. This last approach is known as constraint FPs, where the constraints are integrated into the extracting procedure.

Approaches for extracting patterns with constraints have been suggested for many different types of constraints regarding event occurrence, timing, data, and combinations of these, as used in real-world monitoring systems. In that, the MPP [33] uses a multi-valued decision diagram (MDD) within a prefix projection method for mining constraint sequence patterns. Next, a query-constraint-based ARM (QARM) [34] uses a Top-k Non-Redundant (TNR) technique to produce association rules for the examination of multiple, different clinical datasets. Recently, a constraint-

Table 2 Colossal patterns with length constraints

#	Colossal patterns satisfying length constraints
1	B E G H J
2	B G H
3	E G H J
4	B C E G H
5	B D G H
6	B F G H
7	E F G H J
8	F G H

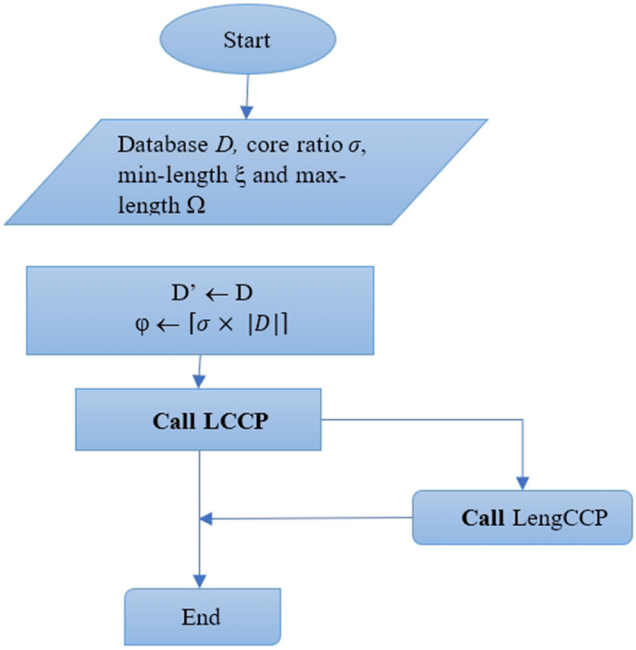

Fig. 1 Flowchart of the LCCP algorithm

programming model [35] can easily be expanded to mine different types of rules. It can be applied to any kind of user constraints. After that, two algorithms named MWAPC and EMWAPC [36] apply the dynamic bit vector structure based on the PreWAP tree to mine web access patterns. Recently, a coding method has been used to transform binary sequences into DNA-based sequences [37] that satisfy the maximum run-length and the GC-content for each sequence. The EHIL (Efficient High utility Itemsets with Length constraints) [38] methods to mine HUIs by incorporating length constraints reduce the number of candidates. The NetDAP [39] technique for pattern matching with gap constraints is based on a structure named an approximate single-leaf Nettree.

In addition, a first method named CPCP-Miner [28] based on the CPCP-Tree structure for mining colossal patterns with itemset constraints was developed. This algorithm rapidly eliminates non-colossal pattern candidates with constraints.

3 Background

Consider the set of items $X = \{e_1, e_2, e_3, \ldots, e_m\}$, where e_i is an item ($1 \le i \le m$). A transaction $T = \langle x_1, x_2, x_3, \ldots, x_n \rangle$ is the set of itemsets, where $x_j \subseteq X$ ($1 \le j \le n$) is an itemset. A transaction database $D = \{t_1, t_2, t_3, \ldots, t_{|D|}\}$, where $|D|$ is the number of

Table 3 Items and their support

Item	A	B	C	D	E	F	G	H	J	K	L
Support	2	7	5	4	6	7	9	9	6	4	2

Table 4 A transaction database after filtering items

TID	Items
1	J K
2	B C D E G H J
3	B C E F G H J
4	B C D E F G H J
5	B E F G H J
6	B H K
7	B D F G H
8	E F G H J
9	F G H
10	B C D E G H
11	C F K
12	G K

transactions in the database D and t_i ($1 \le i \le |D|$) is a tuple in form $\langle tid, t \rangle$, where *tid* is the identifier of transaction t_i. Table 1 shows an example transaction database D.

Given a pattern Q, an itemset $P \subseteq Q$ is a σ-core pattern of Q if and only if $\text{supp}(Q)/\text{supp}(P) \ge \sigma$, $\sigma \in (0,1]$ is called the core ratio [17], where $\text{supp}(Q)$ is denoted the support of pattern Q. A pattern P is called a colossal pattern in a set of frequent patterns if and only if there does not appear an itemset Q such that $P \subset Q$ and P is a σ-core pattern of Q.

Given a transaction database D, a core ratio σ, a *min-length threshold* ξ, and a *max-length threshold* Ω, the problem of extracting colossal patterns with length constraints is stated in Definitions 1–3.

Definition 1 (Colossal pattern with min-length constraint) Extracting colossal patterns with min-length constraint in database D is to extract all samples of colossal pattern P in D with $|P| \ge \xi$.

Definition 2 (Colossal pattern with max-length constraint) Mining colossal patterns with max-length constraint in database D extracts all samples of colossal pattern P in D with $|Q| \le \Omega$.

Based on Definitions 1 and 2, the problem of extracting colossal patterns with length constraints is as given in Definition 3.

Definition 3 (Colossal pattern with length constraints) Extracting colossal patterns with length constraints means finding all the patterns in D that satisfy the max-length constraint and min-length constraint.

Example 1: For Table 1 with $\sigma = 0.25$, $\xi = 3$ and $\Omega = 5$, we have colossal patterns with length constraints as shown in Table 2.

Table 2 is created from Table 1 by removing the items that do not satisfy the support threshold φ and deleting the transactions smaller than the min-length constraint ξ. The support threshold $\varphi = \lceil \sigma \times |D| \rceil = 3$ ($\lceil 0.25 \times 12 \rceil$). In Table 1, the support threshold of two items A and L is 2 less than $\varphi = 3$, so they are removed, and the number of items in transactions 1 and 12 is 2, less than $\xi = 3$, so these two transactions are removed. Similarly, item K is deleted because its support is 2. After K is deleted, transactions 6 and 11 are also removed, and we have the outcomes shown in Table 2.

Table 5 New database after removing the transactions

TID	Items
2	B C D E G H J
3	B C E F G H J
4	B C D E F G H J
5	B E F G H J
6	B H K
7	B D F G H
8	E F G H J
9	F G H
10	B C D E G H
11	C F K

Table 6 Items and their support after removing transactions

Item	B	C	D	E	F	G	H	J	K
Support	7	5	4	6	7	8	9	5	2

4 LCCP: A fast algorithm for mining colossal patterns with length constraints

In this section, two theorems are given to ensure the theoretical foundation of the proposed algorithm, which is cost-effective in mining colossal patterns with length constraints. Theorem 2 is used to quickly omit candidate patterns that

Table 7 A transaction database after removing transactions and items and rearranging the *TIDs*

TID	Items
1	B C D E G H J
2	B C E F G H J
3	B C D E F G H J
4	B E F G H J
5	B D F G H
6	E F G H J
7	F G H
8	B C D E G H

Fig. 2 LCCP algorithm for mining colossal patterns with length constraints

```
Input: Transaction database D, core ratio σ, min-length ξ and max-length Ω.
Output: All colossal patterns ℂ that satisfy length constraints.
Method:
1. Calculate support threshold φ ← [σ × |D|];
2. Let D' be the database after removing all items less than φ and removing transactions less than ξ;
3. Initial the [ℑ]← {{t1,1}, {t2,1}, ..., {tm,1}} from D';
4. ℂ ← [ℑ];
5. Call procedure LCCP([ℑ], ℂ, φ, ξ, Ω);
Procedure LCCP ([ℑ], ℂ, φ, ξ, Ω)
6. Sort-Pattern-Order([ℑ]);
7. If [ℑ.supp]= φ-1 then
8.     For each t in [ℑ] do
9.         If (|t.pattern| ≤ Ω and isColossal(t.pattern)) then
10.            Insert ℂ ← t.pattern and ℂ ← t.supp;
11. Else
12.    For each xi in [ℑ] do
13.        If (xi.supp + |[ ℑ]| - i ≥ φ) then
14.            [ℑ1] ← ∅;
15.            For each yj in [ℑ] with j > i do
16.                z.pattern ← xi.pattern ∩ yj.pattern;
17.                z.supp ← xi.supp +1;
18.                If (|z.pattern| = |yj.pattern|) then
19.                    [ℑ] = [ℑ] \ yj;
20.                If (|ℑ.pattern| ≥ ξ) then
21.                    [ℑ1] = [ℑ1] ∪ z.pattern;
22.            If (|xi.pattern| ≤ Ω) then
23.                Call procedure LengCCP([ℑ1], ℂ, φ, ξ);
24.            Else
25.                LCCP([ℑ1], ℂ, φ, ξ, Ω);
Procedure LengCCP ([ℑ], ℂ, φ, ξ)
26. Sort-Pattern-Order([ℑ]);
27. If [ℑ.supp]= φ-1 then
28.    For each t in [ℑ] do
29.        If (isColossal(t.pattern)) then
30.            Insert ℂ ← t.pattern and ℂ ← t.supp;
31. Else
32.    For each xi in [ℑ] do
33.        If (xi.supp + |[ ℑ]| - i ≥ φ) then
34.            [ℑ1] ← ∅;
35.            For each yj in [ℑ] with j > i do
36.                z.pattern ← xi.pattern ∩ yj.pattern;
37.                z.supp ← xi.supp +1;
38.                If (|z.pattern| = |yj.pattern|) then
39.                    [ℑ] = [ℑ] \ yj;
40.                If (|ℑ.pattern| ≥ ξ) then
41.                    [ℑ1] = [ℑ1] ∪ z.pattern;
42.            LeCCP([ℑ1], ℂ, φ, ξ);
```

cannot be guaranteed to satisfy the min-length constraint. Theorem 3 is evaluated to be effective in quickly checking candidates with the max-length constraint.

According to [21], a CP-Tree includes two kinds of elements (vertex and arc), which can be briefly described as follows:

- Each vertex includes three attributes: a set of transaction IDs, the pattern included in the set of transaction IDs, and the support of a pattern.

For example, the nodes {BCDEGHJ} and {BCEFGHJ} is represented by 1 × 0111101101, 2 × 0110111101, and its support is 2.

- An arc connects a node P at level k to a node Q at level $(k+1)$ if the *TID* of P is the prefix of that of Q.

For example, the node 1 × 0111101101 connects to the node 2 × 0110111101, creating a node 12 × 0110101101 because *TID* 011 is the prefix of *TID* 2.

Definition 4 (Subsuming of a pattern) Given two patterns P_x and P_y in the CP-tree, if $P_x \subseteq P_y$, then P_x is subsumed by P_y.

Theorem 1 [27] If a pattern P of a node is subsumed by any colossal pattern, then all the patterns generated from this node cannot be colossal patterns.

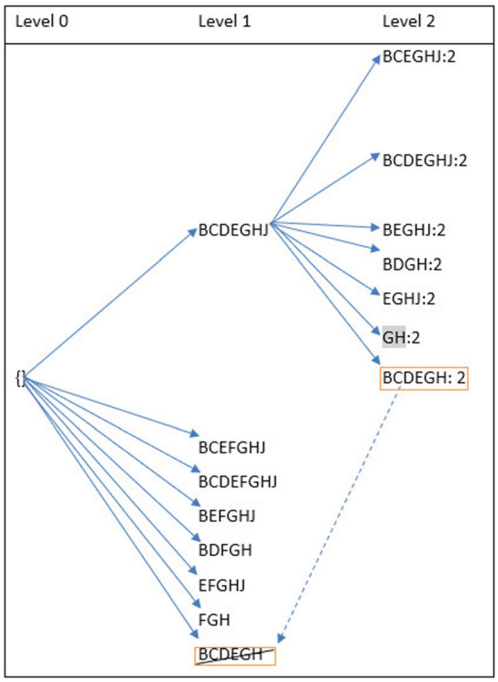

Fig. 3 CP-tree after expanding node {BCDEGHJ}

Theorem 2 (min-length constraint-based for pruning candidates) If a node n in the CP-tree does not satisfy the min-length constraint, then all its child nodes cannot satisfy the min-length constraint.

Proof According to Definition 1, n does not satisfy the min-length constraint, i.e. $|n.\text{pattern}| < \xi$. Besides, n_1 is a child node of n then $|n_1.\text{pattern}| \leq |n.\text{pattern}|$. Therefore, $|n_1.\text{pattern}| < \xi$ or n_1 cannot satisfy the min-length constraint.

Based on Theorem 2, we need not expand a node if it does not satisfy the min-length constraint. Therefore, we can prune the exploration area in the mining process.

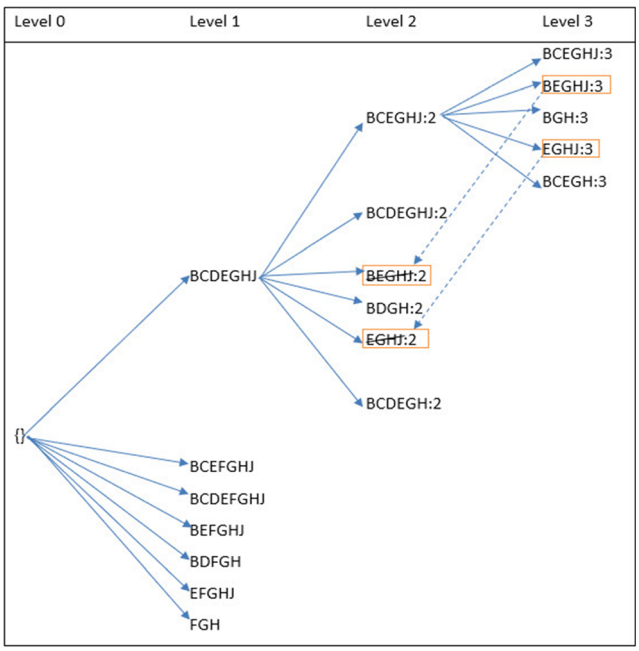

Fig. 4 CP-tree after expanding node {BCDEGHJ}

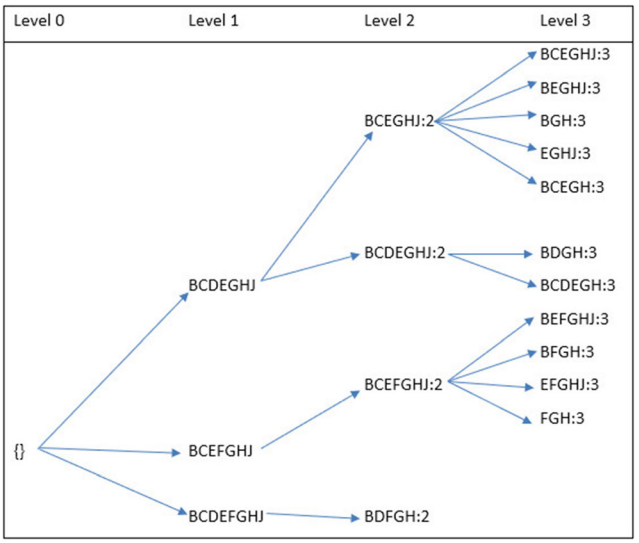

Fig. 5 Final CP-tree for mining colossal patterns with length constraints

Theorem 3 (max-length constraint-based for checking candidates) If a node n in the CP-tree satisfies the max-length constraint, then all its child nodes satisfy the max-length constraint.

Proof According to Definition 2, n satisfies the max-length constraint, i.e. $|n.\text{pattern}| \leq \Omega$. Besides, n_1 is a child node of n then $|n_1.\text{pattern}| \leq |n.\text{pattern}|$. Therefore, $|n_1.\text{pattern}| \leq \Omega$ or n_1 satisfies the max-length constraint.

Based on Theorem 3, we need not check all the child nodes of a node n if it satisfies the max-length constraint, and this saves time as we do not need to check many nodes in CP-Tree.

4.1 Proposed algorithm

In this section, we develop an efficient method, named LCCP, for quickly extracting colossal patterns with length constraints

Table 8 Characteristics of the experimental databases

Database	# of items	# of transactions
Accidents	468	340,183
Connect	130	67,557
Chess	75	3196
Mushroom	120	8124

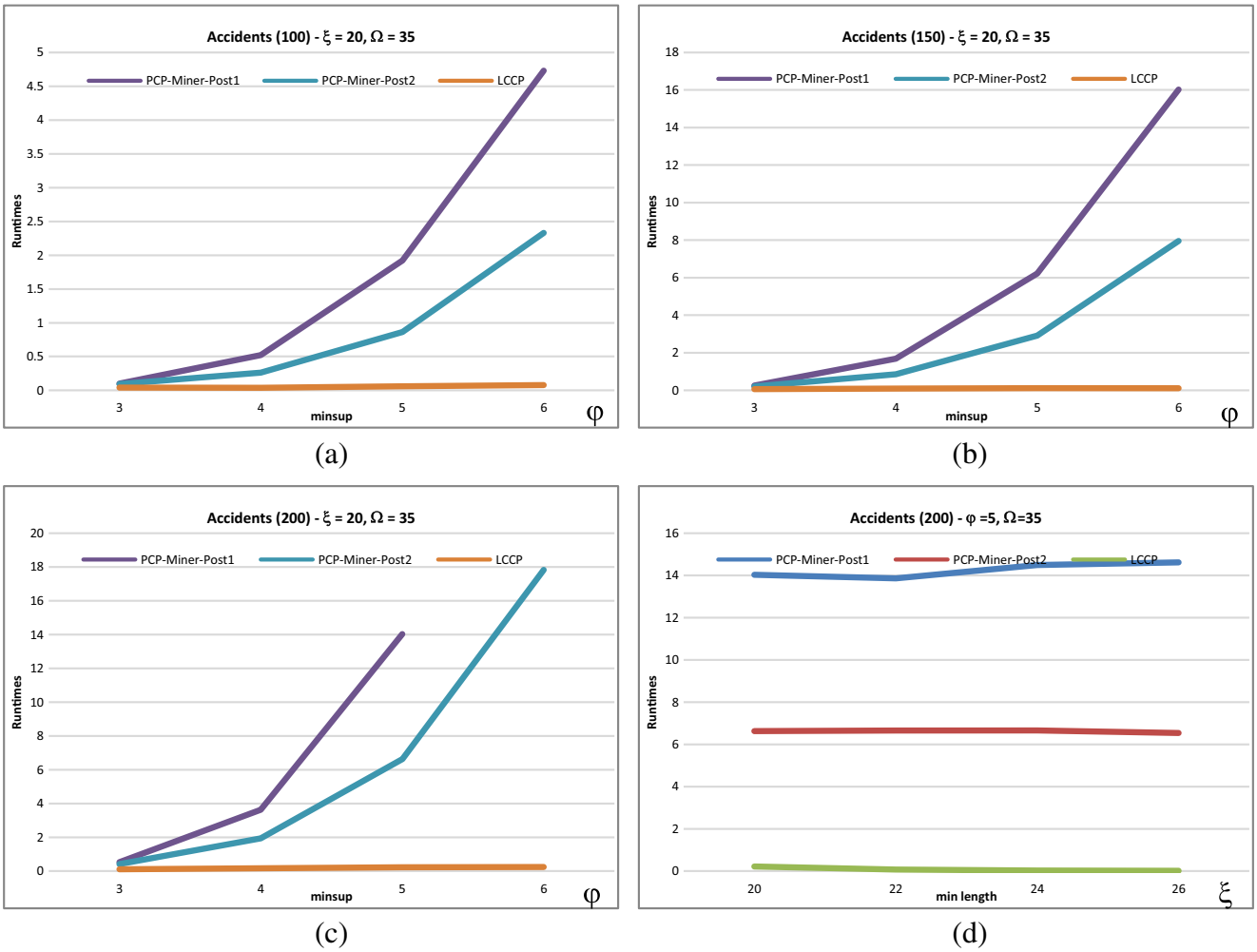

Fig. 6 Various threshold φ and transactions, fixed ξ = 20 and Ω = 35 on Accidents dataset. **a** Run with 100 transactions, **b** Run with 150 transactions, **c** Run with 200 transactions, **d** Run with 200 transactions

(min-length and max-length). LCCP relies on PCP-Miner [21] to efficiently generate and prune colossal pattern candidates. It is also based on Theorem 2 to prune candidates that cannot meet the min-length constraint, and based on Theorem 3 to eliminate candidates that do not need to have the max-length constraint checked.

Figure 1 shows the LCCP approach for extracting colossal patterns with length constraints. It is expanded from PCP-Miner [21] to efficiently generate candidates. Theorem 2 is used to prune candidates (Line 2, Lines 20–21 and 40–41). In Line 2, LCCP uses Theorem 2 to remove all transactions with a length less than ξ. Lines 20 and 40 check the condition of Theorem 2. If the condition is true, then it prunes all the child nodes of *T* (Lines 21 and 41). Line 22 checks the condition of Theorem 3; if it is true, LCCP is called (Line 23) to mine colossal patterns with the min-length constraint (this means we need not check the max-length constraint). The rest of the LCCP and LengCCP algorithms are the same as PCP-Miner [21].

4.2 An example

Example 2: We illustrate the LCCP algorithm based on the information given in Example 1. Firstly, the support of each item in Table 1 is calculated, and the results are as shown in Table 3.

With φ = 3 (=⌈0.25 × 12⌉), we can see that A and L do not satisfy φ, so they are filtered. The database in Table 1 after filtering items that do not satisfy φ is shown in Table 4.

The number of items in transactions 1 and 12 is 2, which is smaller than ξ (according to Example 1, ξ = 3), so these two transactions are removed. The results are shown in Table 5.

Table 6 presents the support of items after removing transactions.

K is deleted because its support is 2. After that, transactions 6 and 11 are removed, and we have the outcomes shown in Table 7.

LCCP uses the database in Table 7 to build the CP-Tree, as shown in Figs. 2, 3 and 4.

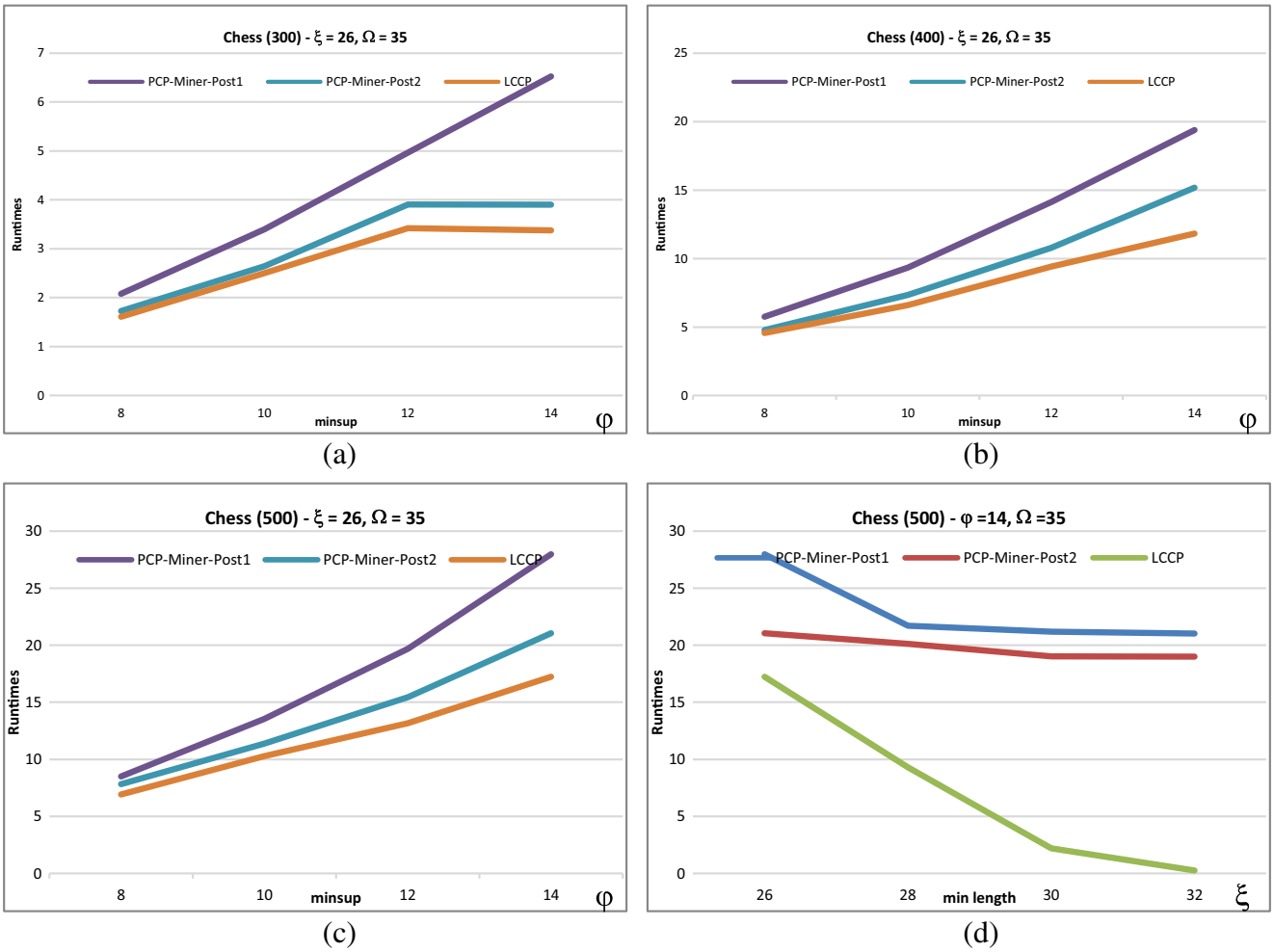

Fig. 7 Various threshold φ and transactions, fixed ξ = 26 and Ω = 35 on Chess dataset. **a** Run with 300 transactions, **b** Run with 400 transactions, **c** Run with 500 transactions, **d** Run with 500 transactions

Firstly, level 1 of the CP-tree will be created from the transactions in Table 7, as shown in Fig. 3. Considering node {BCDEGHJ}, it will join with the node following it to create new nodes in the tree as follows.

- With node {BCEFGHJ}: create node {BCEGHJ} at level 2 (support = 2) ⇒ [*T1*] = {{BCEGHJ}}.
- With node {BCDEFGHJ}: create node {BCDEGHJ} at level 2 (support = 2) ⇒ [*T1*] = {{BCEGHJ}, {BCDEGHJ}}.
- With node {BEFGHJ}: create node {BEGHJ} at level 2 (support = 2) ⇒ [*T1*] = {{BCEGHJ}, {BCDEGHJ}, {BEGHJ}}.
- With node {BDFGH}: create node {BDGH} at level 2 (support = 2) ⇒ [*T1*] = {{BCEGHJ}, {BCDEGHJ}, {BEGHJ}, {BDGH}}.
- With node {EFGHJ}: create node {EGHJ} at level 2 (support = 2) ⇒ [*T1*] = {{BCEGHJ}, {BCDEGHJ}, {BEGHJ}, {BDGH}, {EGHJ}}.
- With node {FGH}: create node {GH} at level 2 (support = 2), because |GH| = 2 < ξ = 3 ⇒ GH} does not add to [*T1*] (by Theorem 2).
- With node {BCDEGH}: create node {BCDEGH} at level 2 (support = 2) ⇒ [*T1*] = {{BCEGHJ}, {BCDEGHJ}, {BEGHJ}, {BDGH}, {EGHJ}, {BCDEGH}}. Because the pattern of the new node is the same as {BCDEGH} ⇒ Remove {BCDEGH} from the Level 1 (by Theorem 1).

After creating [*T1*] = {{BCEGHJ}, {BCDEGHJ}, {BEGHJ}, {BDGH}, {EGHJ}, {BCDEGH}} for node {BCDEGHJ}, the algorithm will be called recursively to expand the child nodes of the nodes in [*T1*]. Consider node {BCEGHJ} with all the nodes following it in [*T1*] (Fig. 4):

- With node {BCDEGHJ}: create node {BCEGHJ} at level 3 (support = 3) ⇒ [T1] = {{BCEGHJ}}.
- With node {BEGHJ}: create node {BEGHJ} at level 3 (support = 3) ⇒ [*T1*] = {{BCEGHJ}, {BEGHJ}}.

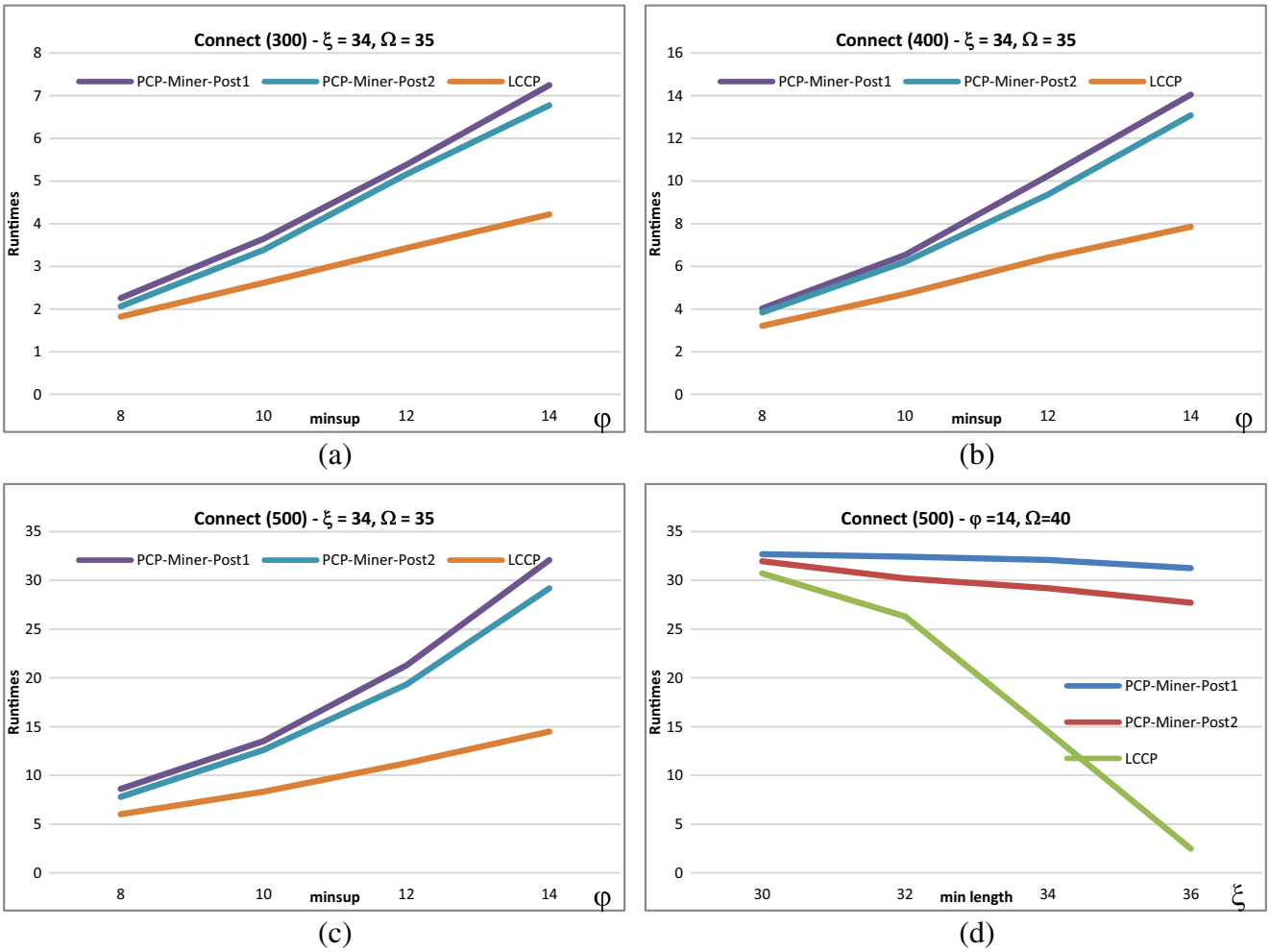

Fig. 8 Various threshold φ and transactions, fixed ξ = 26 and Ω = 35 on the Connect dataset. **a** Run on 300 transactions, **b** Run on 400 transactions, **c** Run on 400 transactions, **d** Run on 400 transactions

Because the pattern of the new node is the same as {BEGHJ} ⇒ Remove {BEGHJ} from level 2 (by Theorem 1).

- With node {BDGH}: create node {BGH} at level 3 (support = 3) ⇒ [*T1*] = {{BCEGHJ}, {BEGHJ}, {BGH}}.
- With node {EGHJ}: create node {EGHJ} at level 3 (support = 3) ⇒ [*T1*] = {{BCEGHJ}, {BEGHJ}, {BGH}, {EGHJ}}. Because the pattern of the new node is the same as {EGHJ} ⇒ Remove {EGHJ} from level 2 (by Theorem 1).
- With node {BCDEGH}: create node {BCEGH} at level 3 (support = 3) ⇒ [*T1*] = {{BCEGHJ}, {BEGHJ}, {BGH}, {EGHJ}, {BCEGH}}.

Figure 5 shows the final CP-tree after expanding all nodes and using Theorems 1 and 2 to prune nodes. After creating level 3, from the patterns in level 3, we get the result of {{BEGHJ}, {BGH}, {EGHJ}, {BCEGH}, {BDGH}, {BFGH}, {EFGHJ}, {FGH}}.

5 Experimental study

5.1 Environment and databases

This section compares the runtime of PCP-Miner-Post1 and LCCP to confirm the proposed method's effectiveness. PCP-Miner-Post1 uses PCP-Miner to mine colossal patterns and chooses the patterns that satisfy the mined pattern's length constraints; PCP-Miner-Post2 uses PCP-Miner to mine colossal patterns but removes the patterns that do not satisfy the length constraints in the mining process. All the experiments exhibited in this section were performed on a PC with an Intel Core i5 3.2 GHz, and 4GB of RAM, running on Windows 10, with the Visual C# 2017. We did experiments on four databases, as shown in Table 8, which were downloaded from http://fimi.cs.helsinki.fi/data/.

The Accident database contains anonymized traffic accident data. The three remaining databases are prepared based on the UCI datasets. The detail of each database is shown in

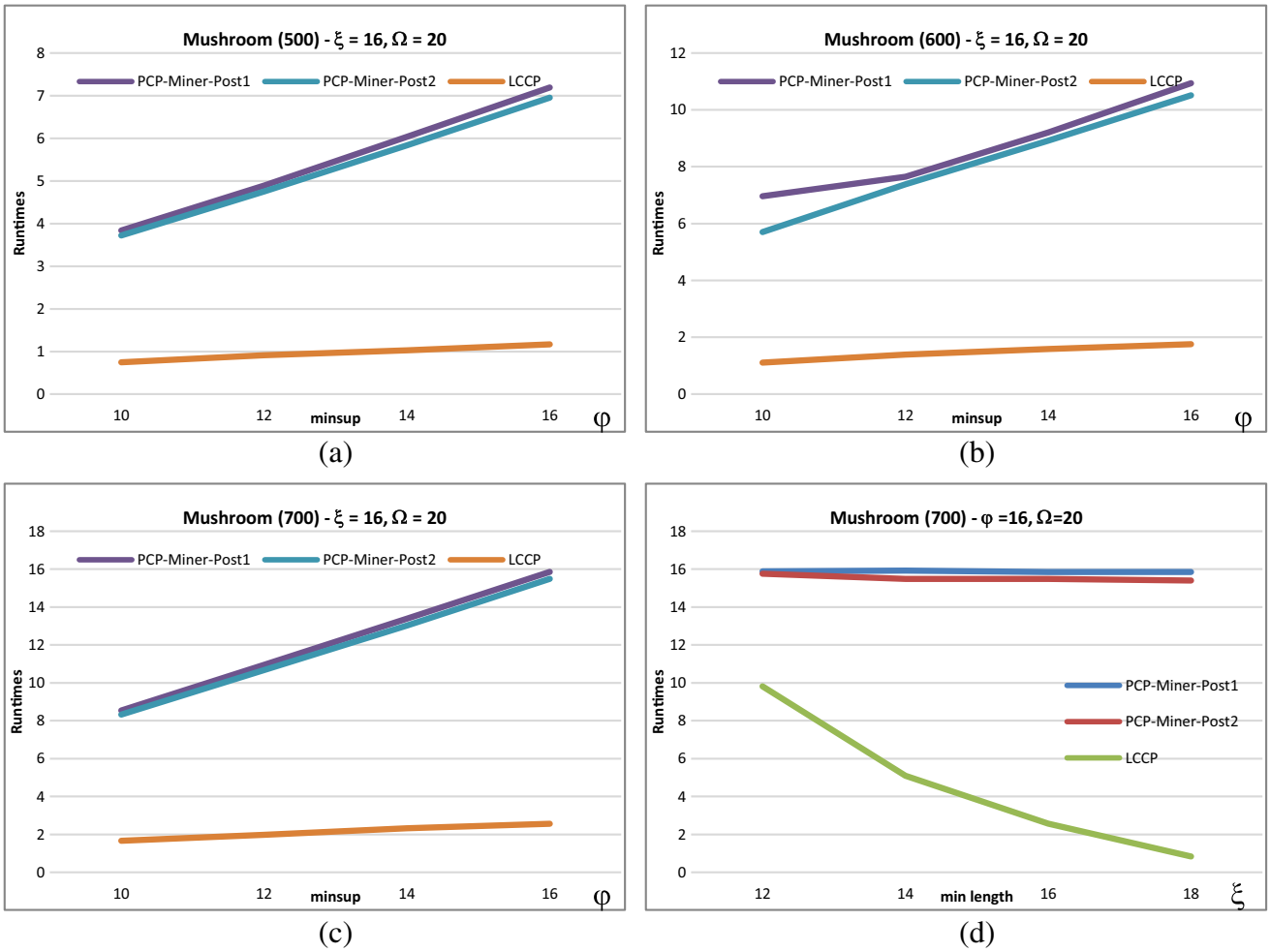

Fig. 9 Various threshold φ, fixed ξ = 16 and Ω = 20 on the Mushroom database. **a** Run on 500 transactions, **b** Run on 500 transactions, **c** Run on 700 transactions, **d** Run on 700 transactions

Table 8. We choose these datasets because they have been normalized and widely applied in various experiments by many famous researchers in the data mining field.

5.2 Comparison of the runtime

We compare the runtime of PCP-Miner-Post1, PCP-Miner-Post2 and LCCP on the four databases in Table 8 with various settings. In the PCP-Miner-Post1 algorithm, the set of colossal patterns was mined first. After that, we selected the patterns that satisfy the length constraints. The PCP-Miner-Post2 algorithm is based on the PCP-Miner algorithm. It checked the length constraints when putting a pattern into the result set. The LCCP algorithm is a newly proposed method, and this applied Theorems 2 and 3 to eliminate the patterns that do not satisfy the constraints during the mining process. We changed the number of transactions incrementally when executing the algorithms for each database to evaluate the proposed method's effectiveness.

Figure 6a–d shows the results of the experiments for the Accidents database. We fix min- and max-length constraints (ξ = 20, Ω = 35), change φ from 3 to 6, and easily find that the runtime of LCCP is less than that of PCP-Miner-Post1 and PCP-Miner-Post2. When increasing the φ value, the LCCP increased very slightly, while both PCP-Miner-Post1 and PCP-Miner-Post2 linearly increased fast.

In Fig. 6d, we fix φ = 5, Ω = 35 and change the min-length constraint ξ from 20 to 26. The runtime of LCCP is six times and fourteen times faster than those of PCP-Miner-Post2 and PCP-Miner-Post1, respectively. Also, the runtime of LCCP decreased while those of PCP-Miner-Post1 and PCP-Miner-Post2 showed almost no decrease.

Similarly, for the Chess database, we fix the min- and max-length constraints in Fig. 7a–d with ξ = 26 and Ω = 35, and change φ from 8 to 14, and the results show that LCCP is always the best algorithm.

Next, in Fig. 7d, we fix φ = 14, Ω = 35, change the min-length constraint ξ from 26 to 32, and the LCCP is still better

than the PCP-Miner-Post1 and PCP-Miner-Post2 algorithms. In particular, when we increase the min-length constraint, the runtime of LCCP decreases significantly, while the PCP-Miner-Post1 and PCP-Miner-Post2 both only decrease slightly. The results in Fig. 6d show the efficiency when using Theorems 2 and 3 compared to without using them (i.e., when using LCCP compared to PCP-Miner-Post2).

Figure 8a–d perform the same tests for the Connect database. We find that the runtimes of PCP-Miner-Post1 and PCP-Miner-Post2 are nearly the same, while the runtime of LCCP is much better than that of both PCP-Miner-Post1 and PCP-Miner-Post2.

When we fix $\varphi = 14$, $\Omega = 35$, and change ξ from 30 to 36, the runtime of LCCP is significantly decreased while those of PCP-Miner-Post1 and PCP-Miner-Post2 only fall slightly. Thus, the experiments confirmed that the LCCP is the best algorithm of all three methods for the Connect database.

Figure 9a–d compares the runtimes of LCCP, PCP-Miner-Post1 and PCP-Miner-Post2 for the Mushroom database with various settings. We fix the min- and max-length constraints as $\xi = 16$, $\Omega = 20$, and LCCP is always the best algorithm for mining colossal patterns with length constraints for the Mushroom database. In particular, when we change φ, the time gaps between the runtime of LCCP and those of PCP-Miner-Post1 and PCP-Miner-Post2 are very large.

In Fig. 9d, we fix $\varphi = 16$ and $\Omega = 20$ and change the min-length constraint from 12 to 18, and the results show that the runtime of LCCP is much less than those of PCP-Miner-Post1 and PCP-Miner-Post2. Therefore, LCCP outperforms PCP-Miner-Post1 and PCP-Miner-Post2 for this dataset.

In short, through the above experiments, we can conclude that LCCP outperforms PCP-Miner-Post1 and PCP-Miner-Post2 for mining colossal patterns with length constraints.

6 Conclusions and future work

This paper proposed two new theorems and a method named LCCP for mining colossal patterns with length constraints. Firstly, the problem of mining colossal patterns with length constraints was presented. Secondly, two theorems were introduced and used for quickly determining whether a colossal pattern satisfies the length constraints. Based on these, an efficient algorithm for mining colossal patterns with length constraints was proposed, with the theorems having eliminated those patterns that do not satisfy the length constraints to improve the mining times. Finally, experiments were conducted to verify the proposed approaches. The experimental results show that the LCCP algorithm is better than the PCP-Miner-Post1 and PCP-Miner-Post2 algorithms.

This paper focused on min-length and max-length constraints for mining colossal patterns. In the future, the combination of many constraints will also be studied. Colossal pattern mining with constraints will be researched and implemented in a distributed or parallel environment.

References

1. Telikani A, Gandomi A, Shahbahrami A (2020) A survey of evolutionary computation for association rule mining. Inf Sci 524:318–352
2. Shao Y, Liu B, Wang S, Li G (2020) Software defect prediction based on correlation weighted class association rule mining. Knowl Based Syst 196:105742
3. Alibasa M, Calvo R, Yacef K (2019) Sequential pattern mining suggests wellbeing supportive behaviors. IEEE Access 7:130133–130143
4. Huynh B, Trinh C, Huynh H, Van T, Vo B, Snásel V (2018) An efficient approach for mining sequential patterns using multiple threads on very large databases. Eng Appl of AI 74:242–251
5. Fournier-Viger P, Yang Y, Yang P, Lin J, Yun U (2020) TKE: Mining Top-K frequent Episodes, in IEA/AIE 2020: 832–845
6. Smedt J, Deeva G, Weerdt J (2020) Mining behavioral sequence constraints for classification. IEEE Trans Knowl Data Eng 32(6): 1130–1142
7. Zou H (2020) Clustering algorithm and its application in data mining. Wirel Pers Commun 110(1):21–30
8. Astrova I, Koschel A, Lee S (2020) Using market basket analysis to find semantic duplicates in ontology. ICCSA 4:197–211
9. Hagen M, Stein B (2018) Weblog analysis, in Encyclopedia of Social Network Analysis and Mining. 2nd Ed.
10. Littmann M, Goldberg T, Seitz S, Bodén M, Rost B (2019) Detailed prediction of protein sub-nuclear localization. BMC Bioinformatics 20(1):205:1–205:15
11. Dessouky M, Taha E, Dessouky M, Eltholth A, Hassan E, El-Samie F (2019) Non-parametric spectral estimation techniques for DNA sequence analysis and exon region prediction. Comput Electric Eng 73:334–348
12. Kumar D, Sharma D (2019) Deep learning in gene expression modeling, in Handbook of Deep Learning Applications, pp. 363–383
13. Bachman J, Gyori B, Sorger P (2018) FamPlex: a resource for entity recognition and relationship resolution of human protein families and complexes in biomedical text mining. BMC Bioinform 19(1):248:1–248:14
14. Deng N, Chen X, Li D, Xiong C (2019) Frequent patterns mining in DNA sequence. IEEE Access 7:108400–108410
15. Lin J, Yang L, Fournier-Viger P, Hong T (2019) Mining of skyline patterns by considering both frequent and utility constraints. Eng Appl AI 77:229–238
16. Sohrabi M, Barforoush A (2012) Efficient colossal pattern mining in high dimensional datasets. Knowl-Based Syst 33:41–52
17. Zhu F, Yan X, Han J, Yu P, Cheng H (2007) Mining colossal frequent patterns by core pattern fusion. ICDE'07, pp. 706–715
18. Dabbiru M, Shashi M (2010) An efficient approach to colossal pattern mining. Int J Comput Sci Network Security 6:304–312
19. Prasanna K, Seetha M (2015) A doubleton pattern mining approach for discovering colossal patterns from biological dataset. Int J Comput Appl 119(21)
20. Prasanna K, Seetha M (2015) Efficient and accurate discovery of colossal pattern sequences from biological datasets: A doubleton pattern mining strategy (DPMine). IMCIP 54:412–421
21. Nguyen T, Vo B, Snásel V (2017) Efficient algorithms for mining colossal patterns in high dimensional databases. Knowl-Based Syst 122:75–89

22. Van T, Vo B, Le B (2018) Mining sequential patterns with itemset constraints. Knowl Inf Syst 57(2):311–330
23. Le T, Nguyen A, Huynh B, Vo B, Pedrycz W (2018) Mining constrained inter-sequence patterns: a novel approach to cope with item constraints. Appl Intell 48(5):1327–1343
24. Nguyen D, Nguyen L, Vo B, Pedrycz W (2016) Efficient mining of class association rules with the itemset constraint. Knowl-Based Syst 103:73–88
25. Bessiere C, Lazaar N, Lebbah Y, M. M. (2018) Users constraints in itemset mining, CoRR abs/1801.00345
26. Nguyen D, Nguyen L, Vo B, Hong T (2015) A novel method for constrained class association rule mining. Inf Sci 320:107–125
27. Vo B, Le T, Pedrycz W, Nguyen G, Baik S (2017) Mining erasable itemsets with subset and superset itemset constraints. Expert Syst Appl 69:50–61
28. Nguyen T, Bay V, Huynh B, Snasel V, Nguyen L (2017) Constraint-based method for mining colossal patterns in high dimensional databases, in Information Systems Architecture and Technology - ISAT, Advances in Intelligent Systems and Computing, pp. 195–204
29. Zulkurnain N (2012) DisClose : discovering colossal closed itemsets from high dimensional datasets via a compact row-tree. University of Manchester
30. Vanahalli M, Patil N (2019) An efficient parallel row enumerated algorithm for mining frequent colossal closed itemsets from high dimensional datasets. Inf Sci 496:343–362
31. Zaki FM, Zulkurnain N (2018) RARE: mining colossal closed itemset in high dimensional data. Knowl-Based Syst 161:1–11
32. Vanahalli M, Patil N (2018) Distributed mining of significant frequent colossal closed itemsets from long biological dataset. ISDA 1:891–902
33. Hosseininasab A, Hoeve W-J, Ciré A (2019) Constraint-based sequential pattern mining with decision diagrams. AAAI 33:1495–1502
34. Abeysinghe R, Cui L (2018) Query constraint based mining of association rules for exploratory analysis of clinical datasets in the National Sleep Research Resource. BMC Med Inf Decis Making 18(S-2):89–100
35. Belaid M, Bessiere C, Lazaar N (2019) Constraint programming for association rules. SDM:127–135
36. Van T, Yoshitaka A, Le B (2018) Mining web access patterns with super-pattern constraint. Appl Intell 48(11):3902–3914
37. Song W, Cai K, Zhang M, Yuen C (2018) Codes with run-length and GC-content constraints for DNA-based data storage. IEEE Commun Lett 22(10):2004–2007
38. Singh K, Bhaskar Biswas B (2019) Efficient algorithm for mining high utility pattern considering length constraints. Int J Data Warehous Min 15(3):1–27
39. Wu Y, Fan J, Li Y, Guo L, Wu X (2020) NetDAP: (δ, γ) - approximate pattern matching with length constraints. Appl Intell 50(11): 4094–4116