J. Shanghai Jiao Tong Univ. (Sci.), 2019, 24(2): 243-253
https://doi.org/10.1007/s12204-019-2058-6

Research on Two Main Construction Methods of Concept Lattices

*DONG Ying** (董　颖), *WU Yue* (吴　悦), *LIU Zongtian* (刘宗田)
(School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China)

Abstract: Because of the completeness of concept lattices, the time complexity of constructing concept lattices has become the main factor affecting the application of formal concept analysis (FCA). The key problems in the research of concept lattices are how to improve the generation efficiency and how to reduce the space and time complexity of constructing concept lattices. So far, reviews on lattice construction algorithms have not been comprehensive. In view of this situation, we give a detailed review on two categories of construction algorithms: batch methods and incremental methods. The first category is a formal context that cannot be updated once the concept lattice has been constructed; the second category is a formal context that can be updated after a new object being added to the formal context. We briefly introduce classical and improved construction methods, illustrate the deficiencies of some algorithms and point out the improvements of the follow-up algorithms. Furthermore, we compare and discuss several key algorithms, and also pay attention to the application of concept lattices. Finally, two further research directions of concept lattices are proposed, including parallel construction methods of concept lattices and research of heterogeneous data concept lattices.
Key words: formal concept analysis (FCA), concept lattice, batch processing, incremental processing
CLC number: TP 301.6　　**Document code:** A

0 Introduction

The concept is introduced from philosophy into the computer science initially[1]. A concept in philosophy is defined as an ideological unit which is composed of two parts: extent and intent. Based on philosophy, formal concept analysis (FCA) was introduced by Rudolf Wille in the early 1980s.

Like a concept in philosophy, a formal concept in FCA is composed of two parts (extent and intent) too. The extent of a formal concept is a set of objects which belong to the formal concept. The intent of a formal concept is a set of attributes which are shared by the objects. As the underlying core structure of FCA, concept lattices essentially describe the relation between objects and attributes and illustrate the generalization and universal instance relationship among concepts. The Hasse graph of a concept lattice realizes the visualization of data[2]. For the last two decades, as a powerful tool in data discovery, concept lattices have played an important role extensively in various disciplines, such as knowledge discovery, data mining and software engineering. However, with the rapid development of computer technology, the amount of data on the Internet increases exponentially, which results in the dramatic increase of the time and space complexity. Therefore, many improved construction methods of concept lattices have been proposed. It is regretful that previous reviews about construction methods of concept lattices are not comprehensive.

Two categories of concept lattice construction methods, including batch generation and incremental generation methods, are given in this paper. We introduce classical algorithms and point out the improvements of the follow-up algorithms. Besides, we briefly explain the deficiencies of some algorithms, compare and discuss several key algorithms, and also pay attention to the application of concept lattices. Finally, two specific ideas about future development directions of concept lattices construction algorithms are proposed.

1 Preliminaries

In this section, we introduce basic FCA notions and conventions. All definitions and propositions are referred to Refs. [3-4], where readers can get more detailed description.

Definition 1 A formal context is a triple of sets, $K(U, A, I)$, where U is the set of objects, A is the set of attributes and $I \subseteq U \times A$ represents a binary relation

Received date: 2018-08-23
Foundation item: the National Natural Science Foundation of China (No. 61273328), and the National Key Foundation for Exploring Scientific Instrument of China (No. 51227803)
∗**E-mail:** grumpy@shu.edu.cn

between U and A; $(x, a') \in I$ or xIa' indicates that the object x has the attribute a'.

A matrix can be seen as a formal context, where a row is an object and a column is an attribute. An example of a formal context can be seen as follows:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \left[\begin{array}{cccc} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{array}\right] \end{array}.$$

Definition 2 For a set of objects $X \in P(U)$ and a set of attributes $B \in P(A)$, the following mapping relationships exist between X and B:

$$f(X) = \{a' | a' \in A, \forall x \in X, xIa'\},$$
$$g(B) = \{x | x \in U, \forall a' \in B, xIa'\}.$$

A formal concept of the formal context is a pair of (X, B), if $f(X) = B$ and $g(B) = X$. Here, X is the extent of the formal concept and B is the intent of the formal concept. The extent X represents a collection of all objects covered by the concept, and the intent B represents a collection of shared attributes by X. Each concept describes a set of objects and their public attributes. The concept, which contains all objects in the formal context, is called a whole concept and the concept with no objects is called a null concept.

Definition 3 The set of all concepts of $K(U, A, I)$ is denoted by $\mathrm{CS}(K)$. Let $C_1(X_1, B_1)$ and $C_2(X_2, B_2)$ be two formal concepts of a formal context. If $A_1 \subseteq A_2$ or $B_1 \supseteq B_2$, then C_1 is called a subconcept of C_2 and C_2 is called a superconcept of C_1. This relationship between C_1 and C_2 is denoted by $(X_1, B_1) \leqslant (X_2, B_2)$; (X_1, B_1) is called a lower neighbor (or a child) of (X_2, B_2) and (X_2, B_2) is called an upper neighbor (or a parent) of (X_1, B_1) if there is no concept $C_3(X_3, B_3)$ satisfying $(X_1, B_1) \leqslant (X_3, B_3) \leqslant (X_2, B_2)$. All formal concepts of the formal context K with the subconcept-superconcept relation constitute a complete lattice, which is called the concept lattice of K. The concept lattice of K is denoted by $L(K)$.

Definition 4 Let (X_1, B_1) and (X_2, B_2) be two concepts of a concept lattice. The upper and lower bounds derived from (X_1, B_1) and (X_2, B_2) are defined as follows:

$$(X_1, B_1) \vee (X_2, B_2) = (g(B_1 \cap B_2), (B_1 \cap B_2)),$$
$$(X_1, B_1) \wedge (X_2, B_2) = ((X_1 \cap X_2), f((X_1 \cap X_2))).$$

Proposition Let $K(U, A, I)$ be a formal context, $\forall X$, $X_1, X_2 \subseteq U$, $\forall B$, $B_1, B_2 \subseteq A$. Part propositions are given as follows:

$$X_1 \subseteq X_2 \Rightarrow f(X_2) \subseteq f(X_1),$$
$$B_1 \subseteq B_2 \Rightarrow g(B_2) \subseteq g(B_1),$$
$$f(g(f(X))) = f(A), \quad g(f(g(B))) = g(B),$$
$$(g(f(X)), f(A)) \in L(K),$$
$$(f(B), f(g(B))) \in L(K).$$

A Hasse graph of $L(K)$ is connected by the concepts of the concept lattice.

Figure 1 represents the concept lattice of the formal context in Definition 1.

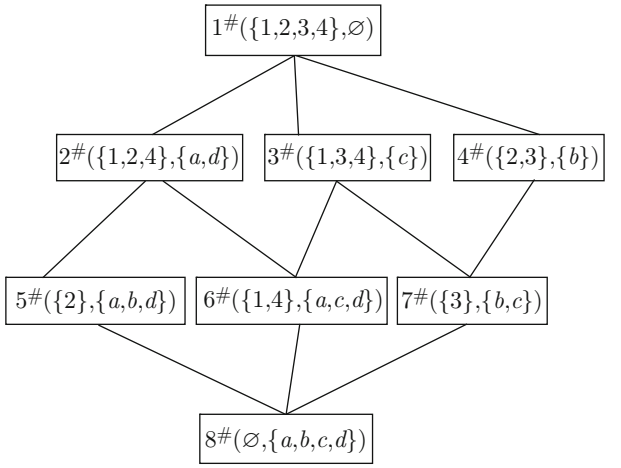

Fig. 1 Concept lattice of the formal text in Definition 1

2 Construction Methods of Concept Lattices

The methods of constructing concept lattices are divided into two categories: batch methods and incremental methods. A batch generation method is a one-time process, because changes of a formal context lead to reconstructing the concept lattice. An incremental generation method can accept changes of the formal context. Incremental methods with flexibility are the focal point of research of construction methods for concept lattices.

In this section, classical batch methods and incremental methods including their improved algorithms are elaborated.

2.1 Batch Construction Methods

Batch methods, which include two tasks: generating formal concepts and finding sub-concept/super-concept relation between concepts, are the earliest proposed methods. According to the task's priority, batch methods are divided into two kinds of models, including task-partition models and task-cross models. According to the differences in construction process, batch methods can be divided into three classes: top-down methods, such as Bordat's method[5]; bottom-up methods, such

as Chein's method[6] and enumeration methods, such as Ganter's method[7].

A task-partition model generates formal concepts firstly and builds relation between concepts next, while a task-cross model builds the relation when generating formal concepts.

In this section, we proceed with discussing classical and improved methods of task-partition and task-cross models firstly.

Figure 2 represents two classification methods of batch construction methods.

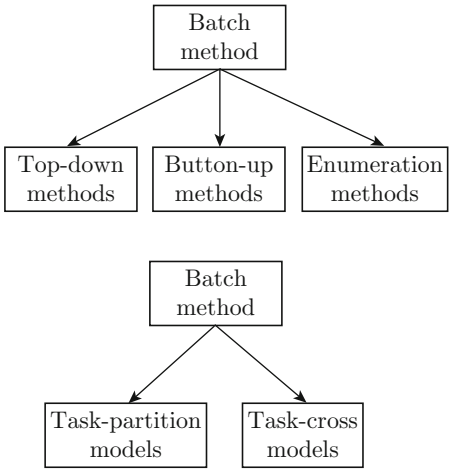

Fig. 2 Two classification methods of batch construction methods

2.1.1 *Task-Partition Model*

Classical task-partition models including NextClosure[7] are proposed by Chein[6], Ganter and Wille[7], Stumme et al.[8], Nourine and Raynaud[9], Godin et al.[10], and so on. Among above algorithms, NextClosure[7], Titanic[8] and Chein's method[6] just generate concepts without building relation between concepts.

NextClosure is the most famous algorithm among all the construction algorithms. In order to represent all formal concepts of the formal context, the algorithm uses a dictionary to sort all concepts without generating the Hasse graph. The bit vector is used to represent the intent of a concept. The vector with an attribute is tagged to 1, and the vector without an attribute is tagged to 0. The size of the bit vector reflects the ranking of the attribute set of a concept in the dictionary. The attribute set of each formal concept can be enumerated because of its uniqueness. From another perspective, NextClosure is also based on closure operations to construct concept lattices[11-12]. The algorithm does not need to save all generated concepts. It only needs to calculate certain subsets of object sets with judging whether the subsets are normal. NextClosure constructs concept lattices by enumeration. The time complexity of the algorithm is $O(|G|^2|M||L|)$, where $|G|$ is the number of objects, $|M|$ is the number of attributes and $|L|$ is the number of connections between objects and attributes.

Titanic optimizes the process for generating formal concepts with a technology to compute frequent item sets. The technology can be used to acquire rules in data mining. It is similar to calculating frequent sets that the algorithm generates all formal concepts in the top-down order layer by layer. Titanic calculates closed sets with a weight function instead of the given closure operator. Firstly, it tags the weight of the largest concept to 1. Then it finds all concepts with a weight of 1, and generates the concepts iteratively layer by layer. Compared with NextClosure, Titanic does not need to enumerate every concept, so its time performance is better than NextClosure. Considering that the storage capacity of the lattice structure is quite large, Titanic only generates all concepts without building subconcept/super-concept relation between concepts.

Chein's method adopts a bottom-up construction method. Firstly, Chein's method calculates the intersection of attribute sets for every conceptual pair in the current layer from bottom. Then, each intersection will be judged whether it is included in the previous concept. If not, the intersection will be generated as the intent of a new concept. Like above two algorithms, Chein's method just generates concepts with relation not being built.

Unlike above discussed three algorithms, as an algorithm of task-partition models, Alaoui's method[10] is just responsible for establishing parent-child relation, not for generating formal concepts. Therefore, Alaoui's method can be seen as a complement to algorithms which just generate concepts. Obviously, it takes $O(|L|^2)$ time to finish connecting between concepts. Nourine's method[9], which constructs a complete concept lattice, not only generates concepts but also establishes parent-child relation between concepts. Parent-child relation can be indexed by searching a dictionary sort tree. The tree is constructed as follows[12]: the complementary sets of concepts' extents can be seen as a basic operating unit. Let $F = \{\cup_{I\in K} I|K \subset B\}$ be a union of subsets of B which is the complement sets of extents for each individual attribute. Here, F generated from B is represented by the tree. Each path of the tree represents an element in F, and the operation with the same prefix shares partial path. After generating the dictionary sort tree, the covering graph of F can be given by deriving conditions which satisfy the coverage relation defined between two elements in F. Then, the corresponding algorithm is given. The time complexity to generate concepts of Nourine's method is $O((|G| + |M|)|G|)$, and the time complexity to establish links is $O((|G| + |M|)|G|)$ too. Nourine's method is superior to NextClosure and Chein's method. Table 1 shows the comparison of above algorithms.

Table 1 The comparison of task-partition models

Task-partition model	Construction method	Whether concepts constructed	Whether relation built	Whether Hasse graph built	Time complexity
NextClosure	Enumeration	Yes	No	No	$O(\|G\|^2\|M\|\|L\|)$
Titanic	Top-down	Yes	No	No	—
Chein's method	Bottom-up	Yes	No	No	$O(\|G\|^3\|M\|)$
Nourine's method	—	Yes	Yes	No	$O((\|G\|+\|M\|)\|G\|\|L\|)$
Alaoui's method	—	No	Yes	—	$O(\|L\|^2)$

2.1.2 *Task-Cross Model*

Task-cross models include Bordat's method[5], Lindig's method[11], and so on. There are two key subprocesses of task-cross models[13]. One is to calculate all the largest extended nodes of a concept; the other is to determine whether the largest extended nodes are generated firstly. According to different orders to generate sub-concepts, task-cross models can be divided into depth-first algorithms and breadth-first algorithms. Depth priority, which is conducive to the local part of data, can make full use of the cache to improve the speed of program execution.

Bordat's method[5] belongs to the top-down construction algorithms. It starts from the top concept, iteratively finds children of the current concept and establishes relation between parents and children. The main idea of the algorithm is given as follows[12]. Let (X_1, X_2) be the current concept. If $B_2 \subseteq B - B_1$ which is a subset of the attribute set maintaining the properties of a complete binary group in X_1, $B_1 \cup B_2$ will be the intent of a subset of (X_1, X_2). The problem of the algorithm is that each binary group will be generated more than one time. What's more, in order to avoid duplication, checking whether each concept has been generated can lead to time overhead. In the end, a Hasse graph is constructed. The time complexity of the algorithm is $O(|G||M|^2)$.

Lindig's method[11] is a more efficient algorithm and can be seen as a "bottom-up Bordat". It generates the bottom concept firstly. All upper concepts of a concept, which is not generated before, will be generated. In order to determine whether a concept is not generated quickly, an index tree of all generated concepts is constructed. The efficiency of searching concepts in Lindig's method is higher than the efficiency of Bordat's method because Bordat's method lacks the index tree to determine whether a concept has been generated. From the point of view of time complexity, searching concepts takes $O(|G||M|)$ time. In view of determining and generating parents, the time complexity of Lindig's method is $O(|G|^2|M|)$. The time performance of the discussed two algorithms of task-cross models depends on the number of objects and attributes in a formal context. Lindig's method is superior to Bordat's method if the number of attributes is larger than that of objects; otherwise, Bordat's method is superior to Lindig's method.

2.1.3 *Improved Batch Construction Methods*

With the development of concept lattices, many improved batch construction methods have been proposed. Xie Run et al.[14] proposed a hierarchical algorithm based on the number of objects belonging to concepts. The concepts in the same layer have the same number of objects. The concepts of a temporary sub-lattice are generated layer by layer, with a width priority, to avoid generating duplicate concepts and to speed up constructing concept lattices. Finally, true sub-lattice concepts are selected. However, this method can just connect two adjacent layers in the Hasse graph. Building relation between concepts not in adjacent layers should be considered separately.

Xie Zhipeng[15] believed that task-cross models are more efficient than task-partition models because information acquired in the process of generating concepts can be used. He took advantage of extended equivalence classes to generate concept lattices in batches quickly. Information of the extended equivalence class for each concept is utilized. The definition of extended equivalence classes is given as follows[15].

Definition 5 For a concept C of a concept lattice, attributes $b_1, b_2 \notin \text{Intent}(C)$. Here, b_1 and b_2, which are extended equivalently for C, are denoted by $(b_1, b_2) \in \text{EQU\_EX}(C)$, under the condition of $g(\text{Intent}(C) \cup \{b_1\}) = g(\text{Intent}(C) \cup \{b_2\})$.

An extended equivalence class of concept C is a division of $B - \text{Intent}(C)$ by the extended equivalence relation EQU_EX(C).

For a concept C, each of its largest extended equivalence classes corresponds to each child of C. All direct and indirect extended concepts of the child concept C_1 are obtained by all indirect and direct extended concepts of its parent C. It is similar to Lindig's method that an index tree is constructed to index the existing concepts in the second step. The node in the tree corresponding to the existing concept will be returned. If the concept does not exist, the corresponding node in the index tree will be inserted. Compared with Bordat's method and Lindig's method, the algorithm reduces the time for finding direct extended nodes. During building relation between concepts, an edge is finished in the

time $O(|A|)$ with a tree structure as the index. Considering that the largest number of edges in a concept lattice is $|L||A|$, the time complexity of constructing the complete concept lattice is $O(|A|^2|L|)$.

In contrast with NextClosure, Kuznetsov[16] proposed a more efficient concept construction method, Cbo (close-by-one), based on closure operation. On the basis of Cbo, fast Cbo[17] and parallel close-by-one (PCbo)[18] have been developed. Cbo[16] adopts a formal test and subset selection method, like NextClosure[7]. The main improvement of Cbo is that it introduces an intermediate structure to generate concepts more effectively. The intermediate structure can be a "tree" to save the set of concepts, and update and construct the Hasse graph. A "stack" can replace the "tree" without the need of constructing a Hasse graph.

Whether it is NextClosure or Cbo, in the case of sparse data, generating a closure and detecting its regularity at each step, and coupling with the calculation of the closure can bring about much time wasting. So, Qi et al.[19] proposed an algorithm by narrowing search space of closure gradually to construct concept lattices. This method sets the power set of attribute sets as the search space of initial closure. The initial separation of the search space is defined as follows.

Definition 6 Let $P(\mathrm{CR})$ be the power set of CR; $\mathrm{CORE} \subseteq m$, $\mathrm{CR} \subseteq M$ and $\forall b_i \in \mathrm{CR}$, $\max(\mathrm{CORE}) < b_i$; $\mathrm{SS}(\mathrm{CORE}, \mathrm{CR}) = \{\mathrm{ss}|\mathrm{ss} = \mathrm{CORE} \cup \mathrm{cr}, \mathrm{cr} \in P(\mathrm{CR})\}$ is denoted by a search space, where CORE is a core set of SS, and CR is a candidate set of SS[19].

The search space is further divided into smaller sub-search space.

Definition 7 A search space SS(CORE, CR) is given; $\mathrm{SS}_i(\mathrm{CORE}_i, \mathrm{CR}_i) = \{\mathrm{ss}|\mathrm{ss} = \mathrm{CORE}_i \cup \mathrm{cr}, \mathrm{cr} \in P(\mathrm{CR}_i)\}$ is a sub-search space defined by an attribute b_i ($b_i \in \mathrm{CR}$). Here, $\mathrm{CORE}_i = \mathrm{CORE} \cup \{b_i\}$ is the core set of SS_i; $\mathrm{CR}_i = \{b'|b_i < b', b' \in \mathrm{CR}\}$ is a candidate set. Specially, SS_0 is defined as $\mathrm{SS}_0(\mathrm{CORE}_0, \mathrm{CR}_0) = \{\mathrm{CORE} \cup \varnothing\}$[19].

The union of regular closed sets of all sub-search spaces is equivalent to the regular closed sets of the original search space. The validity of a sub-search space can be determined by judging whether the core set of the sub-search space can generate a regular closed set, and their results are consistent.

The main steps of the algorithm is given as follows[19].

Search space partition based concepts generation (SSPCG)
Input: a formal concept $K(U, A, I)$
Output: all concepts of $K(U, A, I)$
Begin
  Output the largest concept $(U, \varnothing)$
  For ($i = 1$; $i \leqslant$ the number of sub-search space; i++)
    Mark attributes which can confirm the sub-search space in CR_i
  If (SS_i is a valid sub-search space)
    Generate the largest concept of SS_i
    Futher shrink SS_i until it cannot be divided; output the smallest concept of SS_i
  End
  Output the smallest concept $(\varnothing, A)$
End

There are some ways to improve batch methods from other special points of view. Zhang et al.[20] proposed an algorithm based on the extension of inter-relevant successive trees to construct concept lattices. The algorithm takes advantage of the order of attributes to construct an inter-relevant successive tree for a formal context, with each attribute set being symbolized as a string. The concepts are generated by searching the inter-relevant tree, while the Hasse graph cannot be generated. Cheng[21] found that a binary formal context can be seen as a Boolean matrix. He introduced the rank of attributes. The concepts are generated from the attribute set B. The remaining concepts are generated by the intersection of those concepts, parents of which are recorded. The Hasse graph is generated finally. The purpose of the intersection operation is to narrow the object sets and to expand the attribute sets. At last, this method is stratified according to the cardinality of the object set for each concept. Since the parents are recorded, this method, which not only builds parent-child relation between adjacent layers but also builds relation between concepts not in adjacent layers, forms a complete lattice structure. However, the method has obvious shortcomings. Firstly, Cheng[21] did not programme to implement this method. Secondly, in the case of large amount and uneven distribution of data, almost every concept has to do intersection operation with other concepts, which results in high labor costs. Compared with Cheng's method[21], Zhang et al's method[20] can be applied more widely, because inter-relevant successive trees are constructed on the basis of texts which can be expressed in various forms. Applied to formal contexts, inter-relevant successive trees are not limited to binary relation formal contexts. In the latest research, Li Xin et al.[22] constructed concept lattices based on irreducible attributes. This method not only generates concepts which are also obtained by doing intersection operation, but also builds relation between concepts. What is different from Cheng[21] is that Li Xin et al. thought each object to do the intersection operation represents each concept in a concept lattice and all concepts in the set P^J consist of a set of irreducible attribute concepts[22]. In this process, new concepts are connected with the concepts in the original concept lattice. The time performance is optimized because a new concept is only

constructed by its higher "neighbors".

2.2 Incremental Construction Methods

Among proposed batch methods, only a few methods can generate Hasse graphs. Even some of them just generate concepts without building parent-child relation between concepts. It is undeniable that parent-child relation between concepts is vital for mining association rules. Moreover, a formal context is not static usually, which leads to batch algorithms being not suitable for constructing concept lattices in change. Therefore, it is imperative to develop incremental construction methods for concept lattices.

2.2.1 *Classical Incremental Construction Methods*

The basic idea of incremental methods is given as follows[10]. The object to be inserted is intersected with all the concepts in the original concept lattice, and different operations are performed according to the results of the intersection. There are three main problems to be solved during incremental construction[15]: judging the type of concepts in the original concept lattice, avoiding generating duplicate concepts, and updating edges.

Some typical incremental algorithms are given as follows[12]: Godin's method[10], GALOIS[23], and AddIntent[24]. They are briefly described below.

Godin's method is the most classical incremental algorithm. The concepts in the original concept lattice $L(X, B, R)$ are divided into three categories when a new object x^* is inserted into $L(X, B, R)$. The new concept lattice is denoted by $L^* = (X \cup \{x^*\}, B, R^*)$. The definitions of the three types of concepts in $L(X, B, R)$ are given below[15].

Definition 8 (Modified Nodes) If a node C satisfies $\text{Intent}(C) \subseteq f(\{x^*\})$, C is a modified concept. If C is a modified node, it will be updated to $(\text{Extnet}(C) \cup \{x^*\}, \text{Intent}(C))$ in L^*.

Definition 9 (Generator Nodes) If a node $C_1 = (X_1, B_1)$ satisfies: ① Itersection $= B_1 \cap f(x^*)$ and for any node C_2 in L, $\text{Intent}(C_2) \neq$ Intersection; ② for any node C_3 satisfies $C_3 > C_1$ in L, $\text{Intent}(\text{C}_3) \cap B_1 \neq$ Intersection, where C_1 is a generator node. The nodes which are $(\text{Extent}(C_1) \cup \{x^*\}$, $\text{Intent}(C_1) \cap f(x^*))$ in L^* generated by generator nodes are called new nodes. There is a one-to-one correspondence relationship between a generator node and a new node.

Definition 10 (Old Nodes) Nodes in L except generator nodes and modified nodes are called old nodes.

Table 2 shows a summary of the modifications resulting from the updating process with respect to the classification of nodes.

Table 2 A summary of modifications resulting from updating process according to the classification of nodes

Type of node	Extent	Intent	Condition
Modified node C	$\text{Extent}(C) \cup \{x^*\}$	No change	$\text{Intent}(C) \subseteq f(\{x^*\})$
Generator node C	No change	No change	For any node C_1 in $L, C_2 > C$, Intent $(C_1) \neq$ Intersection, Intent $(C_2) \cap B \neq$ Intersection
Old node C	No change	No change	Nodes except modified nodes and generator nodes in L
New node generated by C	$\text{Extent}(C) \cup \{x^*\}$	Intent $(C) \subseteq f(\{x^*\})$	—

Godin also gave an improved algorithm. It is not necessary to check all concepts in the concept lattice. The algorithm just needs to check nodes which have common attributes with new instances. On the basis of Godin's method, Xie Run[25] gave the judgment condition of generator nodes: $B_0 \overset{\cup}{_{x \in I}} f(x)$ after full investigation of the concept lattice, where B_0 is the attribute set of the added object. The improved method reduces the search range of generator nodes and speeds up the updating process of the lattice.

GALOIS[23] is similar to the basic idea of Godin's method. The main differences are that when making connections, GALOIS finds the minimum upper bound and the largest lower bound of the new node and deletes edges between them, and finally builds links with the new node.

AddIntent traverses the concept lattice to search generator nodes from bottom up[24]. It traverses the neighbor nodes of each node iteratively until finding a generator node C whose neighbor nodes do not contain the intent of the new node Y. The largest time complexity of AddIntent is $O(|G|^2|M|)$. However, a large number of objects will lead to time waste. Moreover, repeated traversing can happen when Y is the intent of some nodes.

2.2.2 *Improved Incremental Construction Methods*

To judge the type of the node to be inserted quickly, Xie and Liu[26] proposed a method to construct concept lattices through a tree structure. This method can constrain the search range of new nodes and their children. The left modifications are the same as Godin's method. Figure 3 shows the index tree corresponding to the concept lattice in Fig. 1.

Considering previous incremental algorithms just adding objects gradually, Li Yun et al.[27] proposed an algorithm based on adding attributes. Like adding

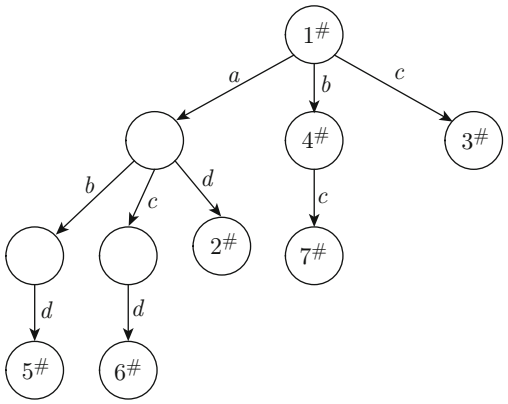

Fig. 3 The index tree corresponding to the concept lattice in Fig. 1

objects, the algorithm also divides the nodes in the new concept lattice L^* into three types. ① Old node: the intersection of the extent of such a node and the added node is empty. ② Modified node: the extent of such a node is included in the extent of the added node. The intent of such a node will be updated to the union of the intent of the original node and the added node. ③ New node: when the intersection of the extent to be inserted and the extent of a node in the original concept lattice does not appear, a new node should be added into the lattice. The intersection is the extent of the new node. The time and space complexity of this algorithm is almost equal to that of Godin's method. When the number of objects is much larger than that of attributes in the formal context, this method is superior to Godin's method.

The most important step in incremental constructing concept lattices is to find generator nodes. It is known that parents of a new node must be a new node or a modified node in Godin's method. At the same time, a generator node must be a child of the new node corresponding to it. So, searching generator nodes can be transformed into looking for the relationship between generator and new nodes or generator and modified nodes.

In Godin's method, in order to search parents and children of a new node, the algorithm needs to traverse all the current node's upper and lower nodes. On the basis of Godin's method, Shen et al.[28] took advantage of the uniqueness of the primary key in a database to judge whether the new node has existed. Moreover, the depth-first traversal of the graph can avoid generating edges that do not satisfy the above conditions. The complexity of traversal is higher than that of Godin's method. After the number of objects growing to a certain number, the time complexity of the algorithm is significantly slower than that of Godin's method. Zhi[29] discovered that generating new nodes is only related to newly generated nodes and has nothing to do with the large number of previously generated concepts. Therefore, the search range will be narrowed to generate new nodes[30]. In order to avoid repeated traversal, the new nodes are constructed to a sub-lattice. The fast incremental construction method proposed by Xie and Liu[26] still needs to search generator nodes, while Zhi's method can determine whether new nodes are generated from the closet parent nodes directly. Thus, Zhi's method is superior to Godin and Xie Zhipeng's method theoretically.

FastAddIntent proposed by Zou et al.[31] can make two improvements on the basis of AddIntent. ① FastAddIntent does not repeat traversing the nodes whose intent is Y. ② The search range to look for parents is narrowed to $C_d = \{(g(D), f(g(D)))|D \in M_d\}$, where $M_d = \{F|B \subset F, |F| = |B| + 1\}$. The most time AddIntent takes for searching generator nodes in a diagram graph is $O(|G|^2|M|)$, while the most time taken by Zhi's method is $O(|G||M|^2)$. Considering that the number of objects is always much larger than that of attributes in general, FastAddIntent has a greater ascension to AddIntent. The whole time FastAddIntent takes for constructing concept lattices is $O(|G||M|^2|L|)$ while that of AddIntent is $O(|G|^2|M||L|)$.

There are other improvements to incremental construction algorithms from other angles. Liu et al.[32] classified the objects with the same attributes into one category on the basis of the classification idea of rough sets. Due to this, the formal matrix of a formal context has been put forward. Based on the formal matrix, a concept lattice's construction method is given theoretically. It is a pity that the method has not been implemented and the idea of classification is also relatively simple. Incremental construction methods mentioned previously just handle problems of adding one object or attribute. In view of this situation, Zhan and Liu[33] put forward a method that can process multiple added objects or attributes incrementally at a time. They built two patter trees, called C-tree and T-tree. C-tree represents the new added transactions and T-tree represents the formal context corresponding to the transaction database. Firstly, this method removes each single-branch mode in T-tree, and then projects C-tree to obtain the projection tree. At last, the C-tree reorganized according to the projection tree is acquired. In the above process, each single-branch mode disassembled from the T-tree contains more than one transaction, so the algorithm realizes batch processing in incremental construction methods.

Whether it is incremental or batch construction, concept lattices are all constructed from starch with the completed partial lattice being the basis for the calculation of the added node. Outrata[34] proposed an efficient algorithm for updating the concept lattice, i.e., calculating the change of the concept lattice directly from the change of the input data[34]. The algorithm is

used for computing the set of all concept lattices. It is significantly faster than recomputing all concepts or the whole concept lattice over again. Moreover, it performs comparably to existing incremental algorithms[34].

Incremental construction algorithms usually update concept lattices for adding objects or attributes, while sometimes some objects or attributes in the formal context need to be deleted in reality. In this case, incremental methods for deleting objects or attributes are put forward. It is worthwhile to point out that there is a fundamental difference between algorithms for deleting objects gradually and attributes' reduction theory[35]: attributes' reduction theory concerns the smallest subset which belongs to the set of attributes of the original concept lattice, so the concept lattice represented by the smallest subset is isomorphic to the original concept lattice. Deleting objects gradually concerns the changes of the concept lattice after deleting some objects or attributes. Therefore, the changed concept lattice is not necessarily isomorphic to the original one.

Carpineto and Romano[36] proposed RemoveObject, and Zhang et al.[37] put forward DeleteObject. In DeleteObject, formal concepts of the original concept lattice are divided into three categories. ① Old concept: the extents of old concepts do not include those of concepts to be removed. ② Modified concept: concepts of this class are removed from the concept lattice. ③ Removed concept: concepts of this class are similar to the new concepts in incremental algorithms for adding objects. After removing them, edges associated with these concepts are readjusted. Main steps of DeleteObject are given as follows.

DeleteObject
Input: the original concept lattice; the object to be removed
Output: the lattice $L(K|_{-\{g\}})$ after removing g
Begin
  Find the lowest concept containing g, and then add it into the candidate set CS_c
  For ($i = 1$; $i \leqslant$ the number of elements in CS_c; $i++$)
    For the element C in CS_c, all unvisited parents of C are added to CS_c
    If (C is a concept to removed)
      Remove C
      Adjust edges associated to C
    Else
      Mark all parents of C as modified concepts
    End if
  Output the lattice $L(K|_{-\{g\}})$ removing g
End

The main steps of RemoveObject are similar to those of DleteObject. But in the first step, RemoveObject traverses the whole concept lattice instead of looking for the lowest concept containing g. When judging whether a concept is a deleted concept or not, both DeleteObject and RemoveObject need to traverse all lower concepts of the removed concept (X, B) to look for the destructor concept in the lattice L^* (corresponding to the generator concept in incremental algorithms for adding objects). This process compares X-$\{g\}$ with the extent of each lower concept of (X, B). In the worst case, this process takes the time $O(|G||M|)$. After deleting concepts, it is a much time consuming procedure for the intents being compared several times during adjusting the relationship of edges.

3 Application of Concept Lattices

Since the concept lattice was proposed, its application has focused on software engineering, data mining, information retrieval, etc. In software engineering, concept lattices provide theoretical support for object-oriented programming, software re-engineering, software reuse, etc., achieving a series of achievements. For example, the interface hierarchy is very important for coders to understand and analyze class libraries in the field of object-oriented programming which is very popular[15]. As a part of the IGLOO, Godin et al.[38] inserted new classes one by one to generate a concept lattice or other form of structure which can help coders better understand the hierarchy of the library based on the specification of the class library. In software re-engineering, code needs to be modular. Lindig and Snelting[39] constructed a concept lattice by analyzing the relationship between the process and global variables, and explained how to get the structure of module in the lattice structure which can be used to evaluate the cohesion and coupling between module candidates. In data mining, extracting association rules from data is one of its core contents. Godin and Missaoui[40] proposed an algorithm for extracting implication rules from concept lattices based on the theory of concept lattices, and used the theory of dependencies on functions in relation databases to deal with implications of rules. Pasquier et al.[41] proposed Duquelme-Guigue basis used for extracting deterministic association rules. They also proposed proper basis and structural basis for extracting approximating association rules. This work was based on the discovery of all frequent item sets. What's more, information retrieval is also an important area for success application of concept lattices. Krohn et al.[42] applied concept lattices to document retrieval. Whenever a keyword is input, the documents related to it are sorted. The first 10 results form a document-term formal context, and then the weight matrix where each element represents the weight of each term in a particular document is assigned to the formal context. Finally, a concept lattice containing search results is formed. The concept lattice as well as the weight matrix

is useful to guide the order and direction of information retrieval. With the deepening of FCA and concept lattice research, application fields of concept lattices expand. Cole and Eklund[43] applied concept lattices to analyze and visualize a medical database with 1 962 attributes and 4 000 prescription summaries. Kent and Bowman[44] built a system called Nebula and corresponding interfaces which are based on concept lattices for digital library.

It is worth noting that it is necessary to extend concept lattices so as to better apply concept lattices to some specific fields. A weighted concept lattice is formed when different weights are assigned to the attributes because the degree of attention to attributes is different. Some applications pay more attention to attribute sets rather than object sets. In this case, the object set can be replaced by its potential. The resulting concept lattice is called quantitative concept lattice. There are still many similar situations, which will not be listed here.

However, no matter how wide the application scope of concept lattices is, time and space complexity of constructing concept lattices will grow exponentially once the scale of the formal context in the application is expanded. How to construct concept lattices efficiently is still the main factor affecting the development of application. The utilization of new methods and techniques to construct concept lattices is the premise of applying concept lattices to larger data. In addition to the two types of concept lattice construction methods studied in this paper, Valtchev and Missaoui[45] proposed a method that constructs a concept lattice by combining its sub-lattices whose formal contexts are split from the original formal context. How to construct concept lattices by combining sub-lattices in a parallel environment which can be efficient provides researchers with new ideas. At the same time, in practice, affairs are always in a changing process. These changes include updating of concepts in a formal context, generation of new concepts, and adjustment of relationships. How to maintain the concept lattice timely and dynamically to cope with these changes also affects the application of concept lattices. Therefore, the application of concept lattice still needs long-term work.

4 Future Prospects

However, some experiments to evaluate the performance of algorithms are not included in the paper. In the future, we are going to select some representative classical and improved algorithms to carry out experiments. The space and time complexity will be compared to verify whether improved algorithms are effective. At the same time, experimental results will be analyzed to judge whether the algorithm has room for improvement and verified experiments will also be done.

For a long time, the development of concept lattices has been relatively slow for the reason that construction algorithms for concept lattices are always restricted by the complexity of time and space, especially for a large amount of data. However, it should not be overlooked that concept lattices show the most essential relationship between concepts, while machine learning mainly draws information from large amounts of data. In contrast with concept lattices, information obtained through machine learning is relatively superficial. Moreover, the structure of a concept lattice is quite stable. As long as the formal context does not change, the lattice structure will not change regardless of the construction method adopted. The range of application of concept lattices is quite broad. In addition to playing an important role in data mining and software engineering, concept lattice can also be applied to web log mining, web text management medium, etc. The research combined with the ontology has become a major direction of concept lattice research. The construction of concept lattices is the premise and foundation of all applications of concept lattices, so it is necessary to continue developing efficient construction algorithms. So, we give two further research directions on the construction methods of concept lattices:

(1) Parallel construction of concept lattices. The general concept lattice construction algorithms are implemented in the environment with one CPU. With the massive increase of the amount of data, the time and space complexity of the algorithms will greatly increase. With the rapid development of multi-CPU computers, parallel construction of concept lattices has room for improvement. There are differences between parallel construction and distributed construction. The distributed construction is still in the environment with a single CPU, while the parallel construction divides construction tasks to many CPUs to complete, effectively improving the efficiency of construction. At present, there are some parallel construction methods, such as the concept lattice construction based on search space partition proposed by Qi et al[19]. Because the divided search space is relatively independent, it is very beneficial to parallel construction. But the research of parallel construction also has the following problems: how to coordinate subtasks, how to ensure the interaction between subtasks, how to divide the task granularity, etc. On the basis of parallel construction, multi-process can be a further research point.

(2) Research on the concept lattices for heterogeneous data. Most of the data in practice is heterogeneous, while the data in the concept lattices is always Boolean. Data structure in formal contexts is single. In order to meet the construction requirements of concept lattices, data usually needs to be preprocessed in real applications. Whether the data is manual preprocessed or computer-pre-processed, it will increase

the time consumption inevitably. Some of research has focused on other structural data, such as multivalued data and interval data. However, concept lattices' construction for heterogeneous data has never made a big breakthrough. Therefore, the study of the construction of concept lattices for heterogeneous data will play an important role in practical application.

5 Conclusion

Concept lattices has not only made great progress in construction methods in more than 30 years since it was put forward, but also developed other research directions, such as attributes' reduction, and rule acquisition. At the same time, it has been widely used in machine learning, pattern recognition, computer network and other fields. No matter how the concept lattices develop, the lattice construction methods must be the foundation of the concept lattice research, but the overview of the concept lattice construction algorithms is not comprehensive. Therefore, this article has done the work of the following several aspects.

(1) Firstly, we introduce some basic theory knowledge of concept lattices, and then introduce two kinds of algorithms in detail: classical batch and incremental construction algorithms. At the same time, we compare several classical batch algorithms from several aspects.

(2) We introduce the main improved batch and incremental methods, and expound some important algorithms in detail. For a more intuitive representation of advantages and disadvantages of these methods, we analyze and compare the time complexity, improvement principle and improvement ideas of these algorithms. Furthermore, construction algorithms for removing objects which have not got enough attention are also involved in this paper.

(3) We also pay attention to the application of concept lattices. We not only enumerate some concrete examples of main application areas of concept lattices, but also introduce several extended concept lattices for adapting to specific application areas. At last, we present some requirements of the application on concept lattices in addition to high time-and-space complexity of construction methods of concept lattices.

References

[1] WILLE R. Restructing lattice theory: An approach based on hierarchies of concepts [C]//*Formal Concept Analysis*. Dordrecht: Reidel Publishing Company, 1982: 314-339.

[2] XIE Z P, LIU Z T. Concept lattice and association rule discovery [J]. *Journal of Computer Research and Development*, 2000, **37**(12): 1415-1421 (in Chinese).

[3] HAN J W, CAI Y D, CERCONE N. Knowledge discovery in databases: An attribute-oriented approach [C]//*Proceedings of the 18th International Conference on Very Large Data Bases*. Vancouver, Canada: Morgan Kaufmann, 1992: 547-559.

[4] SOBIESKI S, ZIELIŃSKI B. Modelling role hierarchy structure using the formal concept analysis [C]//*Annales UMCS Informatica AI X*. [s.l.]: Versita, 2010: 143-159.

[5] BORDAT J P. Calcul pratique du treillis de Galois d'une correspondance [J]. *Mathématiques Et Sciences Humaines*, 1986, **96**: 31-47.

[6] CHEIN M. Algorithme de recherche des sou-matrices premières d'une matrice [J]. *Bull Math Soe*, *Sci*, 1969, **13**: 21-25.

[7] GANTER B, WILLE R. Formal concept analysis: Mathematical foundations [M]. New York, USA: Springer-Verlag, 1997.

[8] STUMME G, TAOUIL R, BASTIDE Y, et al. Fast computation of concept lattices using data mining techniques [C]//*Proceedings of 7th International Workshop on Knowledge Representation Meets Databases*. Berlin, Germany: [s.n.], 2000: 129-139.

[9] NOURINE L, RAYNAUD O. A fast algorithm for building lattices [J]. *Information Processing Letters*, 1999, **71**(5/6): 199-204.

[10] GODIN R, MISSAOUI R, ALAOUI H. Incremental concept formation algorithms based on Galois (concept) lattices [J]. *Computational Intelligence*, 1995, **11**(2): 246-267.

[11] LINDIG C, GBR G D. Fast concept analysis [EB/OL]. [2018-08-23]. https://www.researchgate.net/publication/2812391_Fast_Concept_Analysis

[12] QI H. Knowledge discovery methods research based on formal concept analysis [D]. Changchun, China: College of Computer Science and Technology, Jilin University, 2005 (in Chinese).

[13] JIN Y. Research on algorithms for sequential pattern mining based on concept lattice [D]. Changchun, China: College of Computer Science and Technology, Jilin University, 2007 (in Chinese).

[14] XIE R, LI H X, MA J, et al. Hierarchic construction of concept lattice [J]. *Journal of Southwest Jiaotong University*, 2005, **40**(6): 837-841 (in Chinese).

[15] XIE Z P. Research on knowledge discovery based on concept lattice model [D]. Hefei, China: School of Computer Science and information Engineering, Hefei University of Technology China, 2001 (in Chinese).

[16] KUZNETSOV S O. A fast algorithm for computing all intersections of objects in a finite semi-lattice [J]. *Automatic Documentation and Mathematical Linguistics*, 1993, **27**(5): 11-21.

[17] OUTRATA J, VYCHODIL V. Fast algorithm for computing fixpoints of Galois connections induced by object-attribute relational data [J]. *Information Sciences*, 2012, **185**(1): 114-127.

[18] KRAJCA P, OUTRATA J, VYCHODIL V. Parallel algorithm for computing fixpoints of Galois connections [J]. *Annals of Mathematics and Artificial Intelligence*, 2010, **59**(2): 257-272.

[19] QI H, LIU D Y, HU C Q, et al. An algorithm for generating concepts based on search space partition

[J]. *Journal of Software*, 2005, **16**(12): 2029-2035 (in Chinese).

[20] ZHANG K, HU Y F, WANG Y. An IRST-based algorithm for construction of concept lattices [J]. *Journal of Computer Research and Development*, 2004, **41**(9): 1493-1499 (in Chinese).

[21] CHENG K. Construction method of concept lattice [D]. Chengdu, China: School of Mathematics, Southwest Jiaotong University, 2012 (in Chinese).

[22] LI X, SHAO M W, ZHAO X M. Constructing lattice based on irreducible concepts [J]. *International Journal of Machine Learning and Cybernetics*, 2017, **8**: 109-122.

[23] CARPINETO C, ROMANO G. GALOIS: An order-theoretic approach to conceptual clustering [C]//*Proceedings of the 10th International Conference on Machine Learning*. Amherst, USA: [s.n.], 1993: 33-40.

[24] MERWE D V D, OBIEDKOV S, KOURIE D. Addintent: A new incremental algorithm for constructing concept lattices [C]//*International Conference on Formal Concept Analysis*. Berlin, Germany: Springer, 2004: 372-385.

[25] XIE R. Study on constructing algorithms of concept lattice [D].Chengdu, China: School of Mathematics, Southwest Jiaotong University, 2006 (in Chinese).

[26] XIE Z P, LIU Z T. A fast incremental algorithm for building concept lattice [J]. *Chinese Journal of Computers*, 2002, **25**(5): 490-496 (in Chinese).

[27] LI Y, LIU Z T, CHEN L, et al. Attribute-based incremental formation algorithm of concept lattice [J]. *Mini-Micro Systems*, 2004, **25**(10): 1768-1771 (in Chinese).

[28] SHEN X J, HAN D J, LIU Z T, et al. Improvement on constructing algorithms of concept lattices [J]. *Computer Engineering and Applications*, 2004, **40**(24): 100-103 (in Chinese).

[29] ZHI H L. Research on key technologies in constructing and application of concept lattice [D]. Shanghai, China: School of Computer Engineering and Science, Shanghai University, 2010 (in Chinese).

[30] ZHI H L, ZHI D J. Theory and algorithms of concept lattice incremental construction based on attributes [J]. *Computer Engineering and Applications*, 2012, **48**(26): 17-21 (in Chinese).

[31] ZOU L G, ZHANG Z P, LONG J. A fast incremental algorithm for constructing concept lattices [J]. *Expert Systems with Applications*, 2015, **42**(9): 4474-4481.

[32] LIU L F, WU M D, WANG D, et al. Building concept lattices based on attribute reduction [J]. *Computer Engineering and Science*, 2007, **29**(6): 140-142 (in Chinese).

[33] ZHAN L Q, LIU D X. Study on FCI mining algorithms based on concept lattice [J]. *Journal of Harbin Engineering University*, 2007, **28**(2): 194-197 (in Chinese).

[34] OUTRATA J. A lattice-free concept lattice update algorithm [J]. *International Journal of General Systems*, 2016, **45**(2): 211-231.

[35] ZHANG L, ZHANG H L, YIN L H, et al. Theory and algorithms of attribute decrement for concept lattice [J]. *Journal of Computer Research and Development*, 2013, **50**(2): 248-259 (in Chinese).

[36] CARPINETO C, ROMANO G. Concept data analysis: Theory and applications [M]. London, UK: John Wiley, 2004.

[37] ZHANG L, ZHANG H, SHEN X, et al. An incremental algorithm for removing object from concept lattice [J]. *Journal of Computational Information Systems*, 2013, **9**: 3363-3372.

[38] GODIN R, MILI H, MINEAU G W, et al. Design of class hierarchies based on concept (Galois) lattices [J]. *Theory and Practice of Object Systems*, 1998, **4**(2): 117-133.

[39] LINDIG C, SNELTING G. Assessing modular structure of legacy code based on mathematical concept analysis [C]//*Proceedings of the 19th international conference on Software engineering*. Boston, USA: ACM, 1997: 349-359.

[40] GODIN R, MISSAOUI R. An incremental concept formation approach for learning from databases [J]. *Theoretical Computer Science*, 1994, **133**: 387-419.

[41] PASQUIER N, BASTIDE Y, TAOUIL R, et al. Closed sets based discovery of small covers for association rules [C]//*BDA'1999 International Conference on Advanced Databases*. Bordeaux, France: [s.n.], 1999: 361-381.

[42] KROHN U, DAVIES N J, WEEKS R. Concept lattices for knowledge management [J]. *BT Technology Journal*, 1999, **17**(4): 108-116.

[43] COLE R, EKLUND P W. Scalability in formal concept analysis [J]. *Computational Intelligence*, 1999, **15**(1): 11-27.

[44] KENT R E, BOWMAN C M. Digital libraries, conceptual knowledge systems, and the Nebula interface [R]. Conway, USA: University of Arkansas, 1995.

[45] VALTCHEV P, MISSAOUI R. Building concept (Galois) lattices from parts: Generalizing the incremental methods [C]//*Conceptual Structures*: *Broadening the Base*. Berlin, Germany: Springer, 2001: 1-2.