ICIC Express Letters
Volume **10**, Number **11**, November **2016**

 ISSN 1881-803X
pp. **2623–2629**

ADDCONCEPT: A VERTICAL UNION ALGORITHM OF CONCEPT LATTICES

Hongtao Liang[1,2], Jianliang Xu[1] and Ke Xu[3]

[1]School of Information Science and Engineering
Ocean University of China
No. 238, Songling Road, Laoshan District, Qingdao 266100, P. R. China
hongtaoliang@126.com; xjl9898@ouc.edu.cn

[2]School of Information Engineering
Qingdao University of Technology
No. 236, South Fuzhou Road, Jiaozhou Disctrict, Qingdao 266300, P. R. China

[3]School of Computer Science and Engineering
Beihang University
No. 37, Xueyuan Road, Haidian District, Beijing 100191, P. R. China
kexu@nlsde.buaa.edu.cn

Received March 2016; accepted June 2016

Abstract. *Assembling same-fields concept lattices into a new concept lattice is considered as a novel research method in parallel constructing concept lattice and ontology merging. Its basic idea is to horizontally or vertically divide a large formal context into a set of smaller sub-contexts that share objects or attributes, and then assemble the corresponding lattices of the sub-contexts. Different from the existing concept lattices union algorithms that all the concepts should be compared, this paper presents a new union approach of concept lattices, in which only the new and updated concepts generated in last insertion are compared. Thus, we can drastically reduce the comparison times of the concepts by traversing the structures of the original lattices. Experiments show that the proposed algorithm can improve efficiency obviously, compared with other concept lattice merging algorithms.*

Keywords: Concept lattice, Vertical union, Formal concept analysis, Bottom-up, Concepts inserting

1. **Introduction.** Concept lattice [1] is the core data structure of Formal Concept Analysis and has found its applications in information retrieval, data mining, machine learning and other areas [2, 3, 4]. Constructing concept lattice from formal context is always an important research spot of formal concept analysis. Especially, incremental algorithms have been paid attention to widely for their good maintainability, such as Godin algorithm [5] and AddIntent algorithm [6]. Due to its own completeness of concept lattice, time complex of constructing concept lattice is one of the most limiting factors in the application of formal concept analysis. In [7], an idea of parallel building is proposed, in which formal context is split into several sub-contexts, and then the corresponding sub-lattices are constructed respectively and merged into a complete concept lattice. This idea is also called as distributed processing model of concept lattice [8]. Merging concept lattices is the core work of distributed processing model. According to the splitting fashions, the union algorithms can be divided into horizontal union and vertical union. In vertical union, the concept lattices have the same attribute field and the different object field, which is in contrast to horizontal union.

[9, 10] defined the operation from vertical and horizontal union respectively, and then presented vertical union and horizontal union algorithms to distributed constructing concept lattice. The process of distributed constructing concept lattice is building each sub-concept lattice first, and then inserting all the concepts of one sub-lattice into another sub-lattice in the ascending order of intension (extension) to complete the union of two sub-lattices. In this process, the improved incremental algorithm is used to avoid repeated comparisons of update concepts and new concepts generated by inserting concept.

Based on the vertical union algorithm of [9], a linear index structure is introduced into union algorithm of concepts in [11], where all the sub-concepts are updated quickly by searching for the concepts with the same kind (i.e., concepts with the same extension) between two lattices with the same field. The algorithm takes advantage of the character that synonymous concepts can only generate update concepts, so as to reduce the number of concepts comparison and improve the computation efficiency. The structure of the original concept lattice is utilized and the improved AddIntent algorithm is adopted; [12] provides a top-down union algorithm, which can reduce the repeated comparisons between concepts. However, in all literature, the influences of the parent/child relation between the concepts to be inserted on the change of concept lattice structure have not been considered.

The algorithm presented in this paper is also based on the idea of inserting the concepts of one lattice into another lattice one by one. Compared with the existing concept lattice union algorithms, we make full use of the relations between parent/child and generator/new concepts to simplify the iteration process. Thus, the number of comparison concepts is reduced greatly and experiments show that our approach has significant advantages in efficiency.

The remaining of this paper is organized as follows. Basic definitions of concept lattice and vertical union are given in the next section. Section 3 presents the vertical union algorithm. Comparison experiments are demonstrated in Section 4 to verify the effectiveness of the proposed algorithm. Finally, we conclude this paper in Section 5.

2. **Basic Definitions of Concept Lattice and Vertical Union.** In this section, we will introduce some basic definitions in formal concept analysis and vertical union [1, 8, 10].

Definition 2.1. *In formal concept analysis, formal context is a triple $K = (G, M, I)$, where G is object set, M is attribute set and I is the binary relation between G and M. For an object $g \in G$ and an attribute $m \in M$, gIm means object g has attribute m.*

Definition 2.2. *In a formal context (G, M, I), a formal concept is a pair $C = (O, D)$ satisfying $O = g(D)$, $D = f(O)$, and two following properties:*

$$f(O) = \{m \in M | \forall g \in O, gIm\}, \quad g(D) = \{g \in G | \forall m \in D, gIm\}.$$

Here, $O \in P(G)$ is called as extension of C, and $D \in P(M)$ is called as intension of concept C.

Definition 2.3. *The partial order on concepts is defined as follows. Let $C_1 = (O_1, D_1)$ and $C_2 = (O_2, D_2)$, $C_1 \leq C_2 \Leftrightarrow O_1 \subseteq O_2$ $(D_1 \supseteq D_2)$. If there does not exist C_3 satisfying $C_1 \leq C_3 \leq C_2$, then we call $C_1 \prec C_2$, where C_1 is regarded as child and C_2 is regarded as parent. The parent and child are linked by an edge. The lattice induced by partial order relation $\leq$ is called as concept lattice of formal context K, denoted by $L(K)$.*

Definition 2.4. *Let formal contexts $K = (G, M, I)$, $K_1 = (G_1, M, I_1)$ and $K_2 = (G_2, M, I_2)$ satisfy $G_1 \subseteq G$, $G_2 \subseteq G$, if for arbitrary $m \in M$ and $g \in G_1 \bigcap G_2$ satisfying $gI_1m \Leftrightarrow gI_2m$, then K_1 and K_2 are called as formal contexts with the same attribute field and they are both child contexts of K. Concept lattices $L(K_1)$ and $L(K_2)$ are called as concept lattices with the same attribute field and they are both child lattices of $L(K)$.*

Definition 2.5. *Given two formal contexts K_1 and K_2 with the same attribute field; if $C_1 = (O_1, D_1) \in L(K_1)$, $C_2 = (O_2, D_2) \in L(K_2)$, then we define the vertical addition operation between concepts as $C_3 = C_1 \mp C_2 = (O_1 \cup O_2, D_1 \cap D_2)$.*

In the union process of concept lattices $L(K_2)$ and $L(K_1)$, in order to identify the change of concepts of $L(K_2)$ in lattice $L(K_1)$, we will introduce the following definitions. Let two concept lattices $L(K_1)$ and $L(K_2)$ be with the same attribute field, $C_1 = (O_1, D_1) \in L(K_1)$ and $C_2 = (O_2, D_2) \in L(K_2)$.

Definition 2.6. *When concept C_2 is inserted into $L(K_1)$, for $C_1 \in L(K_1)$, vertical addition operation $C_1 \mp C_2$ is needed. Here, C_2 is called as inserted concept and C_1 is called as compared concept.*

Definition 2.7. *If there exists $D_1 \subseteq D_2$, then compared concept C_1 is called as updated concept of inserted concept C_2. When concept C_2 is inserted into $L(K_1)$, C_1 will be updated as (O_1O_2, D_1), i.e., updated concept C_1 is the result of $C_1 \mp C_2$.*

Definition 2.8. *For $C_2 \in L(K_2)$, if there exists concept $C_1 \in L(K_1)$ satisfying: (1) there does not exist any concept $C = (O, D) \in L(K_1)$ satisfying $D = D_1 \bigcap D_2$; (2) for any parent $C_3 = (O_3, D_3)$ of concept C_1, relation $D \bigcap D_3 = D \bigcap D_1$ does not hold, then C is called as new concept, and compared concept C_1 is called as generator of C.*

In fact, condition (2) of Definition 2.8 is used to ensure that generator concept is infimum concept of new concepts, i.e., the maximum concept in all the concepts satisfies condition (1). When C_2 is inserted into $L(K_1)$, vertical addition operation with concept C_1 will generate new concept $C = C_1 \mp C_2 = (O_1 \bigcup O_2, D_1 \bigcap D_2)$. According to condition (2), compared concept C_1 (generator concept) is the child of C. The concept lattice union algorithm proposed in this paper aims at two concept lattices with the same attribute field. For brevity, it will not be repeated in the following.

3. Main Idea and Theory Basis.

The following theorems provide theory basis for the algorithm of this paper.

Theorem 3.1. *For $C_1 = (O_1, D_1)$, $C_2 = (O_2, D_2) \in L(K_2)$ satisfying $C_1 \leq C_2$, if they have been inserted into $L(K_1)$ and $C = (O, D) \in L(K_1)$ is the generator concept or updated concept of C_1 in $L(K_1)$, then C must also be the new or updated concept of C_2 in $L(K_1)$.*

According to Theorem 3.1, all the generator and updated concepts of parent C_1 must be a subset of new and updated concepts of child C_2. Therefore, child C_2 should be inserted into $L(K_1)$ ahead of parent C_1, and it is enough for parent to compare with only new and updated concepts of child. This will greatly reduce range of concept search and comparison.

Theorem 3.2. *Given $C_1 = (O_1, D_1)$, $C_2 = (O_2, D_2) \in L(K_2)$ satisfying $C_1 \leq C_2$, $C_3 = (O_3, D_3) \in L(K_1)$ satisfying $D_3 \bigcap D_2 \subseteq D_1$; if C_3 and its all higher concepts are generator, new or updated concepts of C_1, then they must be generator, new or updated concepts of C_2.*

Based on Theorem 3.1, Theorem 3.2 shows that when concepts of $L(K_2)$ are vertically added with concepts of $L(K_1)$, child can replace parent for comparison. By Theorem 3.2, if the intension of child C_1 includes the intersection of intensions of parent C_2 and compared concept C_3, then when vertical addition is operated with C_3 and all its higher concepts, parent C_2 can be replaced by child C_1.

Theorem 3.3. *For $C_1 = (O_1, D_1)$, $C_2 = (O_2, D_2) \in L(K_2)$ satisfying $C_2 \leq C_1$, if $C = (O, D) \in L(K_1)$ is the common updated concept in $L(K_1)$ of C_1 and C_2, then $O_2 \subseteq O_1 \subseteq O$.*

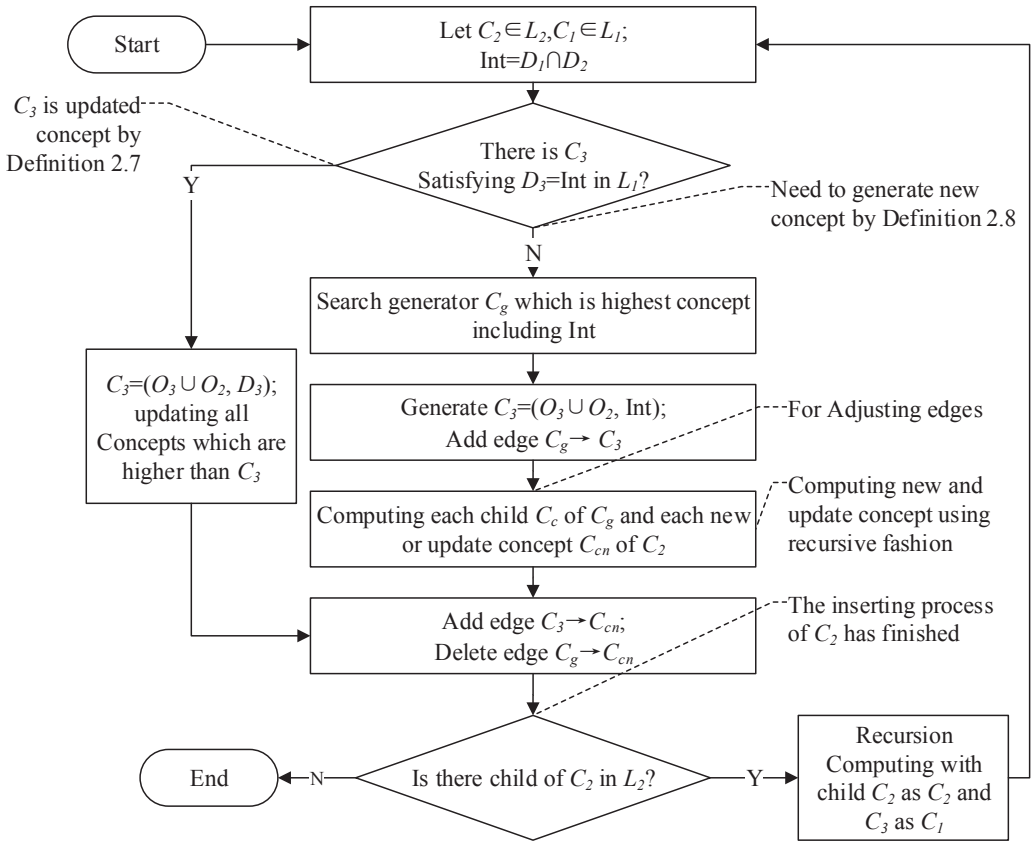

FIGURE 1. Flow chart of main idea of vertical union of concept lattices

According to Theorem 3.3, for the common updated concept C of parent C_1 and child C_2, extension of new concept obtained by updating C after C_2 being inserted should include that after C_1 is inserted. Therefore, when concept C is updated, if child C_2 is used to update C first, updating C by father C_1 can be skipped.

The main idea of vertical union algorithm is shown in Figure 1. For two concept lattices L_1 and L_2 to be merged, beginning from infimum concept of lattices L_1 and L_2, two concepts C_1 and C_2 from L_1 and L_2 carried out vertical addition operation.

If there exists concept C_3 in lattice L_1 whose intension equals the intersection of intensions of C_1 and C_2, by Definition 2.6, the resulting concept of vertical addition of C_1 and C_2 is C_3. Intensions of all the concepts higher than C_3 are subsets of intension of C_2. By Definition 2.7, C_3 and concepts higher than C_3 are all updated concepts, and all of their extensions are added with extension of C_2 to update.

If there does not exist any concept whose intension equals the intersection of intensions of C_1 and C_2, by Definition 2.8, a new concept needs to be generated. The highest concept including intension intersection of C_1 and C_2 should be found first. According to condition (2) of Definition 2.8, this concept is generator by which new concept C_3 can be generated. According to Definition 2.3, when new concept is generated, edges need to be modified. Since generator is the parent of new concept, edge between generators C_g and C_3 need to be established. Since parent of new concept must be the resulting concept of vertical addition operation of C_2 and the concept higher than generator, vertical addition of C_2 and parent of C_g can be recursively computed to obtain these new or updated concepts. New concepts may lie between some parent-child pairs. According to Definition 2.3, the parent-child relation will not exist between these concepts. Hence, these edges should be deleted. After vertical addition of C_1 and C_2 is calculated, the inserting process of C_2 is completed. Then children of C_2 are continued to be inserted into L_1. According to Theorem 3.1, children only need to be vertically added with new or updated concepts.

Therefore, new or updated concept C_3 is recursively inserted as new C_1, and child of C_2 is recursively inserted as new C_2.

4. **Experiment.** To confirm the algorithm's effectiveness, comparison experiments of the proposed algorithm (VUACL), algorithm in [10] (UAMCL), algorithm in [11] (VUACLSC), and algorithm in [12] (FVMCL) are realized by Delphi programming in a computer with 3GB DRAM and 2.30G Hz CPU. Experimental data are random formal contexts, where two parameters are fixed and one parameter is changed in each experiment. Every formal context is vertically split into two formal cotexts with the same size. Then, two lattices are built and merged by the three algorithms respectively. Finally, running times are recorded.

In the first experiment, attribute number of the formal context is fixed at 100, the relative probability of objects and attributes is fixed at 20%, the number of object is changed from 10 to 150 with step 10, and there are 15 formal contexts altogether. The results are shown in Figure 2.

In the second experiment, object number of the formal context is fixed at 100, the relative probability of objects and attributes is fixed at 20%, the number of attributes is changed from 10 to 80 with step 5. The results are shown in Figure 3.

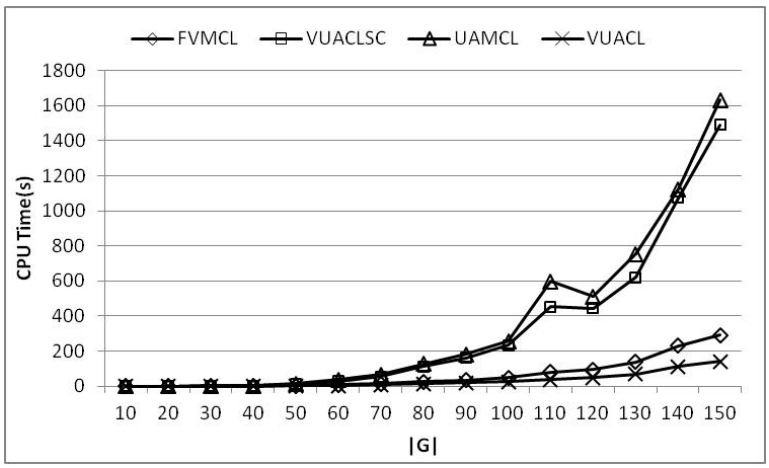

FIGURE 2. The experiment results when $|M| = 100$ and relative probability $= 20\%$

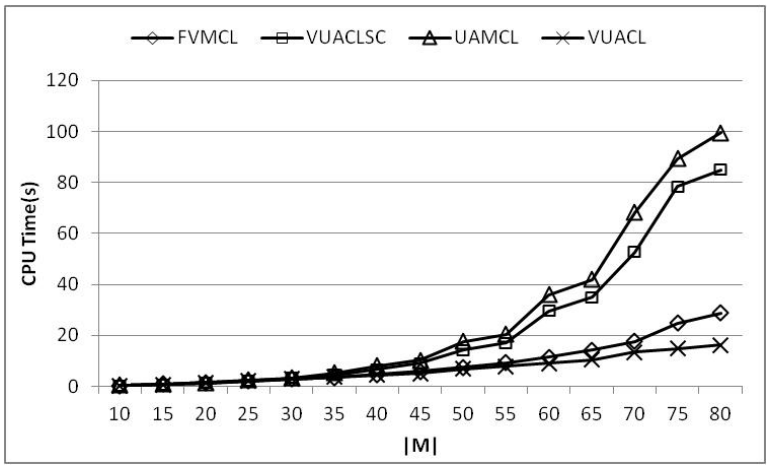

FIGURE 3. The experiment results when $|G| = 100$ and relative probability $= 20\%$

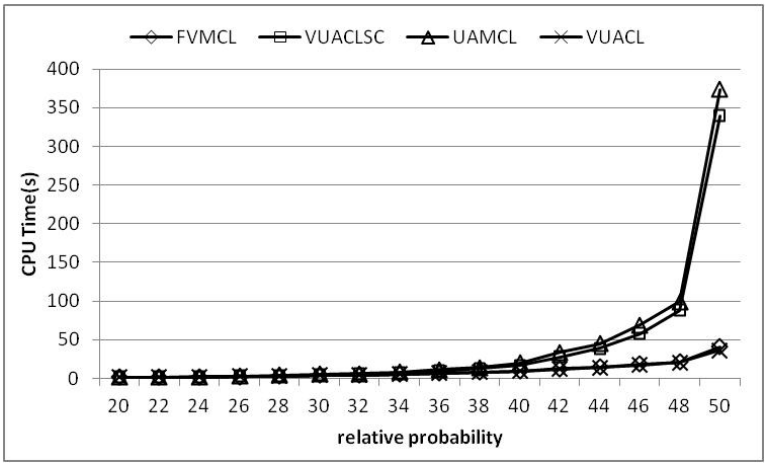

FIGURE 4. The experiment results when $|G| = 100$ and $|M| = 20$

In the third experiment, object number of the formal context is fixed at 100, attribute number is fixed at 20, and the relative probability of objects and attributes is changed from 20% to 50% with step 2%. Figure 4 records the corresponding results via four different algorithms. Obviously, this experiment results demonstrate the effectiveness of our proposed algorithm.

The experiment results show that, with the increase of the size and relative probability of the formal context, the proposed algorithm has obvious advantage on efficiency compared with the algorithms in the existing literature. This can resort to the theory basis built in this paper. Especially, by Theorems 3.1 and 3.2, numbers of operation can be greatly decreased.

5. **Conclusion.** This paper presents an algorithm of assembling concept lattices for parallel constructing concept lattice. The influence of the parent-child relation of the concepts to be inserted on new concepts and updated concepts is analyzed. A bottom-up way is adopted to insert concepts of one lattice into another lattice, resulting in significant reduction of comparison range of concepts. Furthermore, by bottom-up updating extension of concepts, a lot of comparison time is saved. The proposed algorithm can assemble the concept lattices according to the dataset, avoiding to constructing concept lattices from scratch. Finally, the experimental results and analysis show that the bottom-up vertical union algorithm of concept lattices is very quick and effective.

Acknowledgment. This work is partially supported by Science and Technology Project of Shandong Province University (No. J13LN77), and Shandong Province Colleges and Universities Young Teachers Visiting Scholar Fund. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*, Springer-Verlag, Berlin, 1999.

[2] M. E. Cornejo, J. Medina and E. Ramrez-Poussa, Attribute reduction in multi-adjoint concept lattices, *Information Sciences*, vol.294, pp.41-56, 2015.

[3] M. E. Cornejo, J. Medina and E. Ramrez-Poussa, On the use of irreducible elements for reducing multi-adjoint concept lattices, *Knowledge-Based Systems*, vol.89, pp.192-202, 2015.

[4] P. Butka, J. Pócs and J. Pócsová, On equivalence of conceptual scaling and generalized one-sided concept lattices, *Information Sciences*, vol.259, pp.57-70, 2014.

[5] R. Godin, R. Missaoui and H. Alaui, An incremental concept formation algorithm based on Galois (concept) lattices, *Computational Intelligence*, vol.11, pp.246-267, 1995.

[6] D. van der Merwe, S. Obiedkov and D. G. Kourie, AddIntent: A new incremental algorithm for constructing concept lattices, *Proc. of the ICFCA*, LNAI, pp.205-206, 2004.
[7] Y. Li, Z. Liu, X. Shen et al., Theoretical research on the distributed construction of concept lattice, *Proc. of the International Conference on Machine Learning and Cybernetics*, pp.474-479, 2003.
[8] H. Zhi, D. Zhi and Z. Liu, Theory and algorithm of concept lattice union, *Acta Electronica Sinica*, vol.38, pp.455-459, 2010.
[9] Z. Liu, L. Li and Q. Zhang, Research on a union algorithm of multiple concept lattices, *Proc. of the 9th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, Berlin, pp.533-540, 2003.
[10] Y. Li and Z. Liu, Horizontal union algorithm of multiple concept lattices, *Acta Electronica Sinica*, vol.32, pp.1849-1854, 2004.
[11] L. Zhang, X. Shen and D. Han, Vertical union algorithm of concept lattices based on synonymous concept, *Computer Engineering and Applications*, vol.43, pp.95-98, 2007.
[12] L. Zhang, H. Zhang, X. Yu and L. Yin, A fast algorithm for vertically merging concept lattices, *Proc. of PCSPA*, 2011.