Hindawi Publishing Corporation
Scientific Programming
Volume 2015, Article ID 910281, 6 pages
http://dx.doi.org/10.1155/2015/910281

Research Article

Research of Improved FP-Growth Algorithm in Association Rules Mining

Yi Zeng, Shiqun Yin, Jiangyue Liu, and Miao Zhang

Faculty of Computer and Information Science, Southwest University, Chongqing 400715, China

Correspondence should be addressed to Shiqun Yin; qqqq-qiong@163.com

Received 17 September 2014; Revised 22 January 2015; Accepted 22 January 2015

Academic Editor: Oleg V. Gendelman

Association rules mining is an important technology in data mining. FP-Growth (frequent-pattern growth) algorithm is a classical algorithm in association rules mining. But the FP-Growth algorithm in mining needs two times to scan database, which reduces the efficiency of algorithm. Through the study of association rules mining and FP-Growth algorithm, we worked out improved algorithms of FP-Growth algorithm—Painting-Growth algorithm and N (not) Painting-Growth algorithm (removes the painting steps, and uses another way to achieve). We compared two kinds of improved algorithms with FP-Growth algorithm. Experimental results show that Painting-Growth algorithm is more than 1050 and N Painting-Growth algorithm is less than 10000 in data volume; the performance of the two kinds of improved algorithms is better than that of FP-Growth algorithm.

1. Introduction

Data mining is a process to obtain potentially useful, previously unknown, and ultimately understandable knowledge from the data [1]. Association rules mining is one of the important portions of data mining and is used to find the interesting associations or correlation relationships between item sets in mass data [2]. Discovering frequent item sets is a key technology and step in the applications of association rules mining [3]. The most famous algorithm is Apriori put forward by Agawal in the algorithms of discovering frequent item sets [4]. Apriori algorithm through continuous connection scans the database removing unfrequented item sets to find all the frequent item sets in data. But the Apriori algorithm repeatedly scans the database in mining process and produces a large number of candidate item sets, which influence the running speed of mining [5].

FP-Growth (frequent-pattern growth) algorithm is an improved algorithm of the Apriori algorithm put forward by Jiawei Han and so forth [6]. It compresses data sets to a FP-tree, scans the database twice, does not produce the candidate item sets in mining process, and greatly improves the mining efficiency [7]. But FP-Growth algorithm needs to create a FP-tree which contains all the data sets. This FP-tree has high requirement on memory space [8]. And scanning the database twice also makes the efficiency of FP-Growth algorithm not high.

In this paper, we worked out two kinds of improved algorithms—N Painting-Growth algorithm and Painting-Growth algorithm. N Painting-Growth algorithm builds two-item permutation sets to find association sets of all frequent items and then digs up all the frequent item sets according to the association sets. Painting-Growth algorithm builds an association picture based on the two-item permutation sets to find association sets of all frequent items and then digs up all the frequent item sets according to the association sets. Both of the improved algorithms scanning the database only once, improving the overhead of scanning database twice in traditional FP-Growth algorithm, and completing the mining only according to two-item permutation sets, thus, have the advantages of running faster, taking up small space in memory, having low complexity, and being easy to maintain. It is obvious that improved algorithms provide a reference for next association rules mining research.

2. The System Model of Association Rules Mining

2.1. Frequent Item Sets. Set $I = \{i_1, i_2, \ldots, i_n\}$ as a collection of all different items in the database, each transaction T is a subset of I, that is, $T \subseteq I$, and database D is a collection of transactions. For a given transaction database D, the total number of transactions it contains is N. Define the support count(X) of item set $X(X \subseteq I)$ as the number of transactions T in D making $X \subseteq T$ and the support support(X) of item set X as count $(X)/N$ [9]. The number of items in an item set is called dimension or length of this item set, if the length of the item set is k, called k-item set [1].

Definition 1. For a given minimum support, minsup, if the item set meets support(X) $\geq$ minsup, item set X is called a frequent item set and conversely item set X is called an infrequent item set. A set shows association between a frequent item with other items, calling this set a frequent item association set. The minimum support count, minCount, meets minCount = minsup$*N$. When count(X) $\geq$ minCount, one says support(X) $\geq$ minsup [9].

Definition 2. When the length of the item set X is k and support(X) $\geq$ minsup, one calls item set Xk-item frequent set. If $k \geq 3$, one can call item set X multi-item frequent set.

Nature. All nonempty subsets of frequent item sets must be frequent.

2.2. FP-Growth Algorithm. FP-Growth algorithm [10] compresses the database into a frequent pattern tree (FP-tree) and still maintains the information of associations between item sets. Then the compressed database is divided into a set of condition databases (a special type of projection database). Each condition database is dug, respectively, and associates with a frequent item. Transaction database is in Table 1 (support count is 2); mining process using FP-Growth algorithm is shown in Table 1.

Scanning the database for the first time, we can obtain a set of frequent items and their support count. The collection of frequent items is ordered by decreasing sequence of support count. The result set or list writes for L. In this way, we have L = [C:4, D:3, E:3, A:2, B:2].

Building FP-Tree. First, the algorithm creates the root node of the tree, with the tag "null." Then it scans the database for the second time. Each item in a transaction is ordered by the sequence of L. Later it creates a branch for each transaction. For example, the first transaction "001:A, B, C, D, E" contains five items {C, D, E, A, B} according to the sequence of L, generating the first branch $\langle$(C:1), (D:1), (E:1), (A:1), (B:1)$\rangle$ for building FP-tree. The branch has five nodes. In it, C is the children link of root, D links to C, E links to D, A links to E, and B links to A. The second transaction "002:B, C, E" contains three items {C, E, B} according to the sequence of L, generating a branch. In it, C links to the root, E links to C, and B links to E. This branch shares the prefix $\langle$C$\rangle$ with the existing path of transaction "001." In this way, the algorithm makes the count of node C increase by 1 and creates two new nodes $\langle$(E:1), (B:1)$\rangle$ as a link of (C:2). Generally, the algorithm considers increasing a branch for a transaction and when each node follows common prefix, its count increases by 1; algorithm creates node for the item following the prefix and linking.

Table 1: Transaction database.

TID	Items
001	A, B, C, D, E
002	B, C, E
003	C, E, D
004	A, C, D

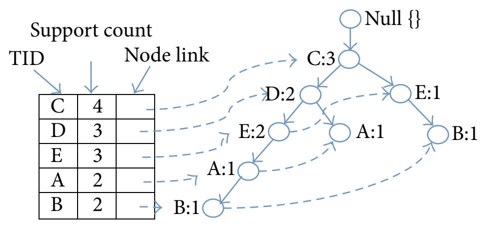

Figure 1: Generating FP-tree.

For convenience of tree traversal, the algorithm creates an item header table. Each item through a node link points to itself in FP-tree. After scanning all transactions, we get the FP-tree displayed in Figure 1.

FP-tree Mining Processing. The algorithm starts by the frequent patterns' length of 1 (initial suffix pattern) and builds its conditional pattern base (a "subdatabase," consisting of the prefix path set which appears with the suffix pattern). Then, algorithm builds a (conditional) FP-tree for the conditional pattern base and recursively digs the tree. The achievement of pattern growth gets through the link between frequent patterns generating by conditional FP-tree and suffix pattern. The mining of FP-tree is summarized in Table 2.

2.3. System Model. Algorithms of frequent patterns mining have been applied in many fields. Researching their system model can facilitate a better understanding of them. Figure 2 is a system model of the improved algorithms in this paper.

The user can get needed knowledge which passes data mining through the data mining platform. Data mining platform includes data definition, mining designer, and pattern filter. Through the data definition, we can do a pretreatment for data and make incomplete data usable; through the mining designer, we can use the improved algorithms to dig data and get useful patterns (here are frequent item sets); through the pattern filter, we can select interesting patterns from obtained patterns.

3. Improved Algorithms Based on the FP-Growth Algorithm

FP-Growth algorithm requires scanning database twice. Its algorithm efficiency is not high. This paper puts forward two improved algorithms—Painting-Growth algorithm and N Painting-Growth algorithm—which use two-item permutation sets to dig. Both algorithms scan database only once to obtain the results of mining.

Table 2: Dig FP-tree through creating conditional subpattern base.

Item	Conditional pattern base	Conditional FP-tree	Frequent pattern
B	{(C D E A:1), (C E:1)}	⟨C:2, E:2⟩	C B:2, E B:2, C E B:2
A	{(C D E:1), (C D:1)}	⟨C:2, D:2⟩	C A:2, D A:2, C D A:2
E	{(C D:2), (C:1)}	⟨C:2, D:2⟩	C E:2, D E:2, C D E:2
D	{(C:2)}	⟨C:2⟩	C D:2

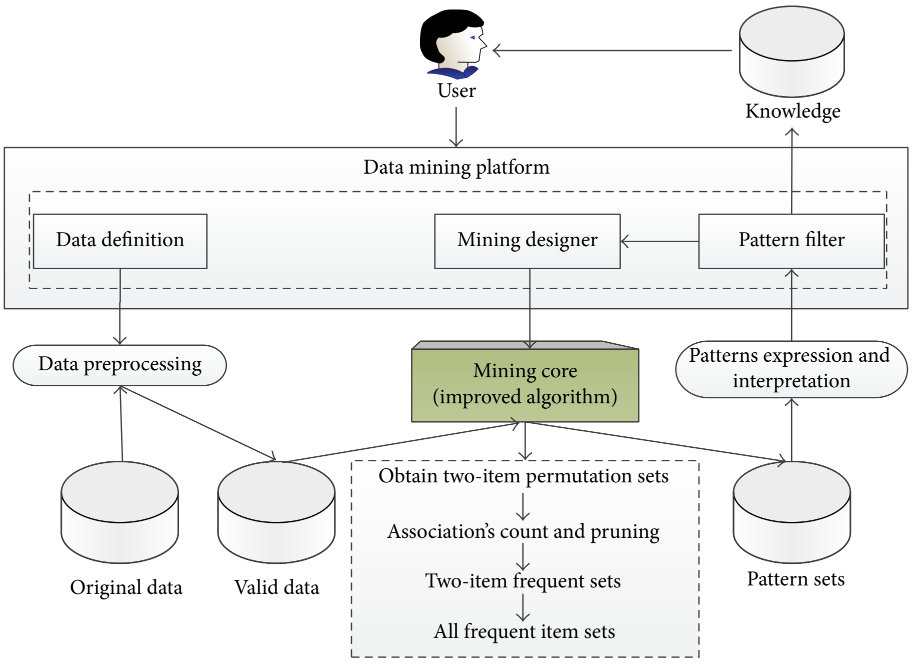

Figure 2: Association rules mining system model.

3.1. Painting-Growth Algorithm. Taking the transaction database in Table 1 as an example, the mining process with Painting-Growth algorithm is as follows.

(1) The algorithm scans the database once, obtains two-item permutation sets of all transactions, and paints peak set (the peak set is a set of all different items in transaction database). Here we take the first transaction as an example.

The first transaction is {A, B, C, D, E}.

Two-item permutation sets after scanning the first transaction are {(A, B),(A, C),(A, D), (A, E),(B, A),(C, A), (D, A),(E, A),(B, C), (B, D),(B, E),(C, B),(D, B),(E, B),(C, D), (C, E),(D, C),(E, C), (D, E),(E, D)}.

Other transactions are similar to the first transaction. The peak set after scanning database is {A, B, C, D, E}.

(2) After obtaining the peak set and two-item permutation sets of all transactions, the algorithm paints the association picture according to two-item permutation sets and peak set. It links the two items appearing in each two-item permutation. When the permutation appears again, the link count increases by 1. The association picture is shown in Figure 3.

(3) According to the association picture, algorithm exploits the support count to remove unfrequented associations. We can get the frequent item association sets as follows: {A(C:2,D:2);B(C:2,E:2); C(A:2,B:2,D:3,E:3); D(A:2,C:3, E:2); E(B:2,C:3,D:2)}.

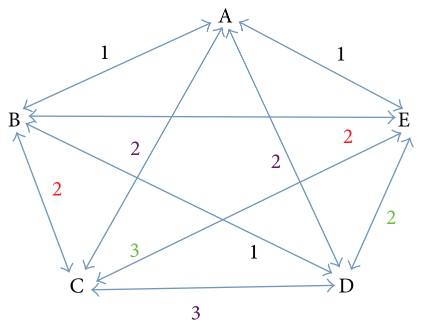

Figure 3: The association picture.

Here we take the item A as an example. A(C:2,D:2) shows that the support count of two-item set (A C) is 2 and the support count of two-item set (A D) is 2. Other items are similar to item A.

(4) According to the frequent item association sets, we can get all two-item frequent sets of this transaction database: {(A,C):2;(A,D):2;(B,C):2;(B,E):2;(C,D):3;(C,E):3;(D,E):2}.

(5) According to the frequent item association sets {A(C:2,D:2);C(A:2,B:2,D:3,E:3);D(A:2,C:3,E:2)}, we can get a three-item frequent set {(A,C,D):2}.

And according to the frequent item association sets {A(C:2,D:2);C(A:2,B:2,D:3,E:3);D(A:2,C:3,E:2)}, we also can get a three-item frequent set {(B,C,E):2}.

Similarly, according to the frequent item association sets {C(A:2,B:2,D:3,E:3);D(A:2,C:3,E:2);E(B:2,C:3,D:2)}, we get a three-item frequent set {(C,D,E):2}.

(6) At this point, we get all frequent item sets.

The algorithm pseudocode is as follows.

Algorithm 3 (Painting-Growth).

Input. Transaction database, minimum support count: 2

Output. All frequent item sets

```
(1) HashMap⟨String, integer⟩ hm0; //define a HashMap set hm0
(2) List⟨String⟩ list,list0; //define the List set list,list0
(3) List⟨String⟩ permutation(); //scan the transaction database, execute two-item arranging to each transaction, return list
(4) paint(Graphics g) //painting method
(5) String[] s=null, x=null; //define String[] s, x
(6) String z, y;
(7) HashMap⟨String, HashMap⟨String, integer⟩⟩ hm=null; //define a HashMap set hm
(8) For (int i=0; i<list. size(); i++)
(9) {
(10) s = list.get(i).split(“,”); //let list.get(i) to a String[]
(11) drawLine(s[0].x, s[0].y, s[1].x, s[1].y); //draw a line between s[0] and s[1]
(12) HashMap⟨String, HashMap⟨String, integer⟩⟩ count(drawLine()); //count the drawing line and return the item associations to hm
(13) }
(14) Iterator it = hm.keySet().iterator; //define key set iterator of hm
(15) z = it.next(); //let the key in key set of hm to z
(16) Iterator it0 = hm.get(z). keySet(). iterator; //define the key sets iterator in value sets of hm
(17) y = it0.next(); //let the key in key sets of value sets of hm to y
(18) if(hm.get(z).get(y)<minsup∗N) //if the value in value sets of hm less than minimum support count
(19) {it0.remove();} //remove the unfrequented item sets
(20) List⟨String⟩ combination(hm.get(z).keySet()); //combination the key sets in value sets based on key z of hm, return list0
(21) for(int j=0; j<list0.size();j++)
(22) {
(23) x = list0.get(j).split(“,”);
(24) if(count(hm.contain(z+“,”+list0.get(j))==1+x.length)) //if the count of item sets in hm equal with the length of the item sets(first consider the key of hm in the item sets or not)
(25) {hm0.put(z+“,”+list0.get(j),value)};//save the item sets and its support count in hm0
(26) }
(27) return hm0;//gain all frequent item sets
(28) super.paintComponents(g); //execute painting method.
```

3.2. N Painting-Growth Algorithm. The thought of N Painting-Growth algorithm is similar to the Painting-Growth algorithm, but with different implementation method. N Painting-Growth algorithm removes the painting steps. The mining process of N Painting-Growth is as follows.

(1) The algorithm scans the database once and gets two-item permutation sets of all transactions.

(2) Then, the algorithm counts each permutation in two-item permutation sets getting all item association sets.

(3) Later, the algorithm removes infrequent associations according to the support count and gets frequent item association sets.

(4) Finally, it gets all frequent item sets according to the frequent item association sets. Mining ends.

From the above processes it can be seen that the N Painting-Growth algorithm is the removing of painting steps version of Painting-Growth. The implementation methods are different: Painting-Growth algorithm imports java.awt and javax.swing, implementing mining through calling super.paintComponents(g); N Painting-Growth algorithm only passes instantiation of a class in main function to implement.

4. Experimental Results Analysis

To improved algorithms—Painting-Growth and N Painting-Growth algorithm—the biggest advantage is reducing database scanning to once. Comparing with scanning database twice of FP-Growth algorithm, it has improved time efficiency.

Another advantage is that improved algorithms are simple, completing all mining only needing transactions' two-item permutation sets. Although the FP-Growth algorithm is also getting FP-tree to complete mining, the FP-tree builds complexly and requires memory overhead largely. Relatively, the two-item permutation sets can be obtained easily.

Of course, improved algorithms have disadvantages. In Painting-Growth algorithm, the algorithm needs to build the association picture, leading to a large memory overhead. In N Painting-Growth algorithm, the implementation method is less vivid than Painting-Growth algorithm. When using the two improved algorithms to dig multi-item frequent sets, they scan the frequent item association sets repeatedly for count. This reduces the time efficiency.

In order to verify the two kinds of improved algorithms relative to the FP-Growth algorithm existing superiority, we use the Java language, in eclipse development environment, Windows 7 64-bit operating system, implementing the Painting-Growth algorithm, N Painting-Growth algorithm, and FP-Growth algorithm. The data in experiments come from Data Tang—research sharing platform. Transactions in database, respectively, are 1050, 5250, 10500, 21000, 31500, 42000, and 52500.

In experiments, three kinds of algorithms accept the same original data input and support parameter. The algorithms run 20 times in each bout, calculating the mean as a result.

Figure 4 is an execution time comparison figure for Painting-Growth algorithm, N Painting-Growth algorithm, and FP-Growth algorithm under the condition of different transactions. From the figure, on the one hand, starting from 1050 transactions, the execution time of N Painting-Growth algorithm is less than FP-Growth algorithm; at 31500 transactions, the execution time of N Painting-Growth algorithm and FP-Growth algorithm is very close. Afterwards, the time efficiency is not as good as FP-Growth algorithm.

On the other hand, from 1050 transactions, the execution time of Painting-Growth algorithm is a little bit more than FP-Growth algorithm. But with the increase in number of transactions, the execution time is less than the FP-Growth algorithm significantly. Thus it can be seen, from the transactions-execution time comparing, that Painting-Growth algorithm is more stable and efficient than FP-Growth algorithm.

Another, the implementation method of Painting-Growth algorithm and N Painting-Growth is different. The performance is also different. Although N Painting-Growth algorithm omits the painting steps, only around 1050 transactions to 10500 transactions, the execution time of N Painting-Growth algorithm is a little less than Painting-Growth algorithm. Then, with the increase of transaction amount, the performance of Painting-Growth algorithm is far better than N Painting-Growth algorithm. This shows that the implementation method of N Painting-Growth has large memory consumption which leading the execution time of N Painting-Growth grows faster.

Figure 5 is execution time's increase rate comparing of different transaction stages for Painting-Growth algorithm, N Painting-Growth algorithm, and FP-Growth algorithm. There are seven transaction stages; stage 1: 0–1050 transactions, stage 2: 1050–5250 transactions, stage 3: 5250–10500 transactions, stage 4: 10500–21000 transactions, stage 5: 21000–31500 transactions, stage 6: 31500–42000 transactions, and stage 7: 42000–52500 transactions.

From Figure 5, firstly, to Painting-Growth algorithm at initial stage 1, the execution time's increase rate of Painting-Growth algorithm is high. But then, from stage 2 to stage 7, the fluctuation of execution time's increase rate is gentle, stable performance. And at stage 2 to stage 6, the execution time's increase rate of Painting-Growth algorithm is lower than FP-Growth algorithm, superior performance.

Secondly, to N Painting-Growth algorithm at the first three stages, the execution time's increase rate of N Painting-Growth algorithm is lower than FP-Growth algorithm, performing well. But later, the increase rate of N Painting-Growth algorithm is almost higher than FP-Growth algorithm and Painting-Growth algorithm. It also explains why the execution time of N Painting-Growth is rising rapidly.

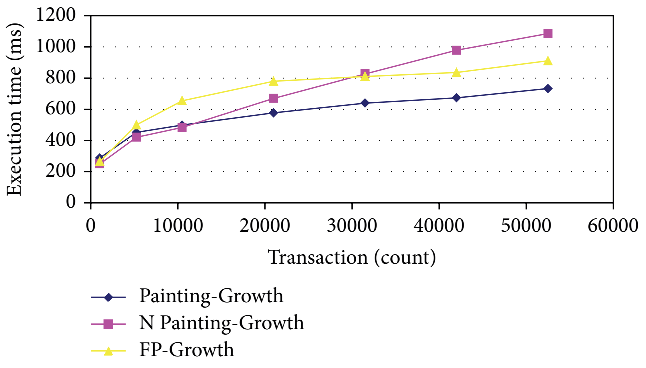

FIGURE 4: Three-algorithm transactions-execution time comparison.

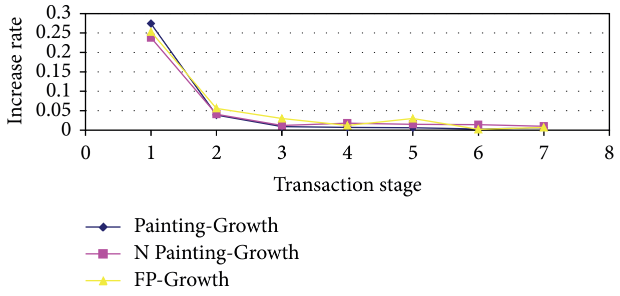

FIGURE 5: The increase rate of three algorithms in different transaction stages.

Finally, to FP-Growth algorithm, although the whole change trend of increase rate is similar to improved algorithms, it has more clear change than improved algorithms in stage 2 and stage 5. So, the FP-Growth algorithm is less stable than improved algorithms.

From what is above it can be concluded that our Painting-Growth algorithm has an obvious breakthrough in data analysis. Unhesitatingly, when the data size is suitable, we can consider adopting improved algorithms to achieve further performance. Carefully, the transactions are less than 10000 and we can consider N Painting-Growth algorithm. In other cases, the Painting-Growth algorithm performs better and we can consider adopting it.

5. Conclusions

In this paper, we put forward improved algorithms—Painting-Growth algorithm and N Painting-Growth algorithm. Both algorithms get all frequent item sets only through the two-item permutation sets of transactions, being simple in principle and easy to implement and only scanning the database once. So, at appropriate transactions, we can consider using the improved algorithms. But we also see the problems of improved algorithm: in large data, the performance of the N Painting-Growth is disappointing. Considering how to make the performance of the improved algorithms more stable, make the removal of unfrequented item associations efficient, and make the mining of multi-item frequent sets quick will be our future work.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the Fundamental Research Funds for the Central Universities (XDJK2009C027) and Science & Technology Project (2013001287).

References

[1] P. Yang and Z. Song, "An improvement to FP-growth algorithm," *Journal of Anhui Institute of Mechanical & Electrical Engineering: Natural Science*, vol. 17, no. 3, pp. 8–13, 2005.

[2] D. Fengyi and L. Zhenyu, "An ameliorating FP-growth algorithm based on patterns-matrix," *Journal of Xiamen University (Natural Science)*, vol. 44, no. 5, pp. 629–633, 2005.

[3] Y. Yang and Y. Luo, "Improved algorithm based on FP-Growth," *Computer Engineering and Design*, no. 7, pp. 1506–1509, 2010.

[4] Q. Ruan, Y. Li, and X. Liu, "A hash table and linear based improved FP-Tree algorithm," *Journal of Yangtze University (Natural Science Edition): Science & Engineering*, vol. 1, pp. 76–79, 2010.

[5] X. Luo and J. Chen, "An improvement algorithm for FP-growth," *Journal of Xi'an University of Science and Technology*, vol. 29, no. 4, pp. 491–494, 2009.

[6] L. Zhichun and Y. Fengxin, "An improved frequent pattern tree growth algorithm," *Applied Science and Technology*, vol. 35, no. 6, pp. 47–51, 2008.

[7] C. Jun and G. Li, "An improved FP-growth algorithm based on item head table node," *Information Technology*, vol. 12, pp. 34–35, 2013.

[8] B. Zheng and J. Li, "An improved algorithm based on FP-growth," *Journal of Pingdingshan Institute of Technology*, vol. 17, no. 4, pp. 9–12, 2008.

[9] N. Xinzheng and S. Kun, "Mining maximal frequent item sets with improved algorithm of FPMAX," *Computer Science*, vol. 40, no. 12, pp. 223–228, 2013.

[10] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, China Machine Press, Beijing, China, 2001, translated by: F. Ming, M. Xiaofeng.