Advanced Engineering Forum *Online: 2011-12-22*
ISSN: 2234-991X, Vols. 2-3, pp 211-216
doi:10.4028/www.scientific.net/AEF.2-3.211
© 2012 The Author(s). Published by Trans Tech Publications Ltd, Switzerland.

Optimization Mode Research of Weighted Undirected Graph

Wei Fan[1,a], Hong Wang[2,b] , Feng Liu[1,c], Qian Xing[1,d]

[1] College of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400054, China
[2] College of Pharmaceutical and Biological Engineering, Chongqing University of Technology, Chongqing 40054, China

[a]13883183559@qq.com, [b]wanghong@cqut.edu.cn, [c]liufeng@cqut.edu.cn, [d]xingqian@cqut.edu.cn

Keywords: undirected graph, weighted, multiple-nodes information version，optimization mode

Abstract. An algorithm is described for constructing a weighted undirected graph structure, based on the principle of directed acyclic graph, used the improved Floyd algorithm to improve the multi-node version of the evolution of information in order to release as an indicator of difference, to improve the reliability of multiple versions of data storage and multi-version storage and query efficiency. Theory and examples show that efficiency is improved using the optimization model in multi-node information system.

1 Introduction

It is very important to save historical information to nodes in multiple-node information system because information on history moments need be analyzed, compared, optimized, integrated and recorded, no matter in collaborative work or gathering multiple-node important information in an interactive environment. At present, researches about multiple-node information are focused on multiple-node information version and the evolving mechanism, version comparison [1], version storage mode etc. [2]. However, in this paper, while characteristics and inheritance of version information storage in a Directed Acyclic Graph (DAG) are analyzed, an optimization strategy of complete version and distribution incremental version is given in order to improve the multi-node information storage and query performance by calculating the difference between versions.

2 Model structures of multiple versions

The Directed Acyclic Graph (DAG), who shows the multi-node version changes in the processing of multiple-nodes information, is an important form of multiple-nodes information management. Incrementally storing saves storage space greatly, because change between adjacent versions is smaller than that between complete storage versions, as there is an inheritance between multiple versions in multiple-node system. On the other hand, if inheritance depth of incremental version is too large, the access speed will be greatly reduced, and the subsequent versions cannot recover because of dependencies between versions, once the basic version is damaged.

The method combined with the integrity and incremental, while some versions use complete storage and some use incremental storage, which takes the advantages of the two storages, is a common storage mode in multi-version system management. In DAG, the primary problem is to choose one from the complete version storage and incremental version storage.

Fig.1 shows the evolution of multiple versions in multiple-node information system. Version copy, which is designed to digestion conflict results branch versions. However, the mergering of branch versions are needed when branch versions reach agreement. As the vision process progressing, more evolved version can be expressed as a directed acyclic graph.

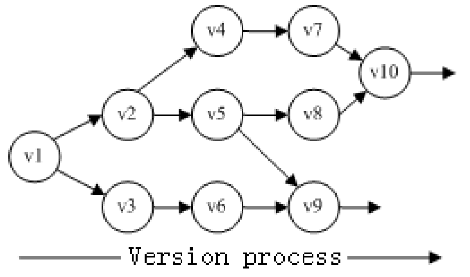

Fig. 1. Directed Acyclic Graph of multiple versions in multiple-node information system

For convenient illustration, we introduce the following definition.

Definition 1: Given Weighted Undirected Graph of multiple version G= （***V*，*E*，*W***）, where ***V*** is a set of multiple versions, *v*1， …， *vm*∈***V***, ***E*** is edge between versions, *e*1,*e*2,…,*en*∈***E***, ***W*** is weight matrix of edge between nodes of version, ***W***=(ω_{ij}) *m*×*m*， ω_{ij} means the weight of edge between versions of v_i and v_j .

Definition 2: Complete version, or the complete storage version, also called the basic version (basic version), $V_i = \bigcup (Obj_i^j, Value(Obj_i^j)), j = 1,...,k$,which records all information with same graph edit format and quickly read speed, is denoted as middle version which is complete version beside basic version.

Definition 3: Incremental version is a kind of version which records the difference between adjacent versions. We define $\Delta V_j = \bigcup (Nid, Sid, Oty, Attr, OldVal, NewVal)_j$, $j = 1...h$, where *Nid* is Collaboration Site Identification(CSI), *Sid* is Primitive Number, *Oty* is Operation code Type, Attr is Attribution operated , *OldVal* is the Value of Primitive attribution before amendment, *NewVa*l is the Modified Value of Primitive attribution, *h* is the number of incremental operation.

Definition 4: The recovery cost of incremental version is defined as operation steps of reduction process based on a complete version. The smaller the recovery cost of incremental version, the higher of the version extraction efficiency.

This definition of incremental version defined by this paper（definition 3） records the change between adjacent versions. When incremental information is small（save as Oldval）, recovery of version can be conduct in double directions. The advantages are：1）The version restoration efficiency is improved because a directed graph is changed into an undirected graph without considering the directions of version recovery and version restored by the shortest path cost can be further reduced. 2) Reliability of the version data is increased because we can choose ccorresponding complete version to restore when some versions have been destroyed.

Definition 5: Multiple-version distribution model *f*（***B*，*D***）. Multiple version node set ***V*** consists the complete version set ***B*** and incremental version set ***D***, ***V*=*B*+*D***.

Definition 6: The property of multiple versions in multiple-node system. The property of multiple versions in multiple-node is generally evaluated from those factors--the total stored information of version, the version's generation rate and data security and reliability of the version. To store information in versions as small as possible, to generate recovery versions as fast as possible, and to increase data security and reliability as high as possible, are the goals of this article.

Versions of the distribution patterns affect the property of multi-version: 1) it is good for saving storage space and increasing the efficiency of network transmission when reducing complete base version. 2) The efficiency of version query is improved because of more choices of the recovery path of the incremental versions and less cost of recovery when more complete versions are set.

This optimization strategy is to determine the integrity of the base version number first, and then choose appropriate versions of the distribution according to version data redundancy (information storage)

3 Optimized algorithm of multiple version distribution model

3.1 Basic algorithms

A directed graph is changed into an undirected graph in multiple version system because of the double directions recovery incremental version. Version which has the shortest summation of distance among those of all versions, are set to complete versions.

The shortages of Floyd algorithm are: 1）enormous amount of computation O（m^3）, especially when increasing nodes, the amount of calculation will increase drastically. 2）standard Floyd algorithm can't deal with the problem while intermediate nodes are in the weighted graph. We should improve the standard Floyd algorithm to compute the shortest path of version recovery according to the properties of multi-version undirected graph.

3.2 The process of optimized algorithm

（1） initialize distance matrix $\boldsymbol{D}^{(0)}=(d_{ij})\ m\times m$，$(i,j=1,\ldots,m)$,where m is the version number. d_{ij} is the weight between version v_i and v_j. There is no direct relationship between v_i and v_j, $d_{i,j}=\infty$. If $i=j$, $d_{ij}=0$. And $d_{ij}=d_{ji}$ because of the undirected graph.

（2）initialize base version set $\boldsymbol{B}=\{v_0\}$.

（3）according to the base version set $\boldsymbol{B}$ and $\boldsymbol{D}^{(0)}$,using improved Floyd algorithm (chapter 3.3) ,compute the distance matrix $\boldsymbol{D}^{(k)}$.

（4）sum row in distance matrix $\boldsymbol{D}$, $C(v_i)=\sum_{j=1}^{m} d_{ij}$ （$i=1,\ldots,m$）.

（5）choose version v_k, from C（v_k）=min{C（v_i）}，$i=1,\ldots,m$.

（6）$\boldsymbol{B}\leftarrow v_k$, if find specified base versions, end the algorithm, otherwise, return to step 3.

3.3 The improvement of the standard Floyd algorithm

This paper has made the following three improvements to the standard Floyd algorithm [3]: ①In the first iteration, there is not any intermediate base versions in based version set, and the distance matrix is symmetrical along the diagonal, so we only need to compute half of elements in the distance matrix. ②When calculating the shortest distance between v_i and v_j ,we reduce the amount of addition operation without comparing the length of the d_{ik}、d_{kj} and d_{ij} because if $d_{ik}>d_{ij}$ or $d_{kj}>d_{ij}$, it means the path v_i pass v_k to v_j is not shorter than inserting (before and after) node v_k. ③After the first iteration, we need to consider their impact on calculation of shortest path because there are intermediate nodes in the network graph (the intermediate base version). Figure 3 shows the process of Improved Floyd Algorithm.

3.4 Example

We show an example of multiple versions in multiple-node system in Fig.2. The original directed acyclic graph turns into an undirected graph structure because of the incremental definition of double directions restore version. The connections between versions in the figure denote that relationship of inheritance existed. We can get a new version after n steps of restoration through another version.

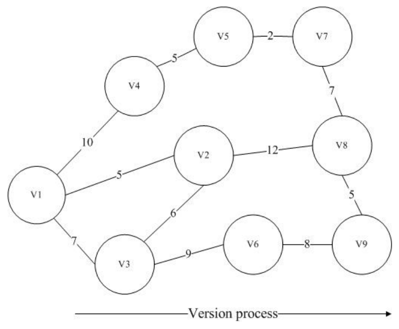

Fig. 2. An example of multiple versions in multiple-node system

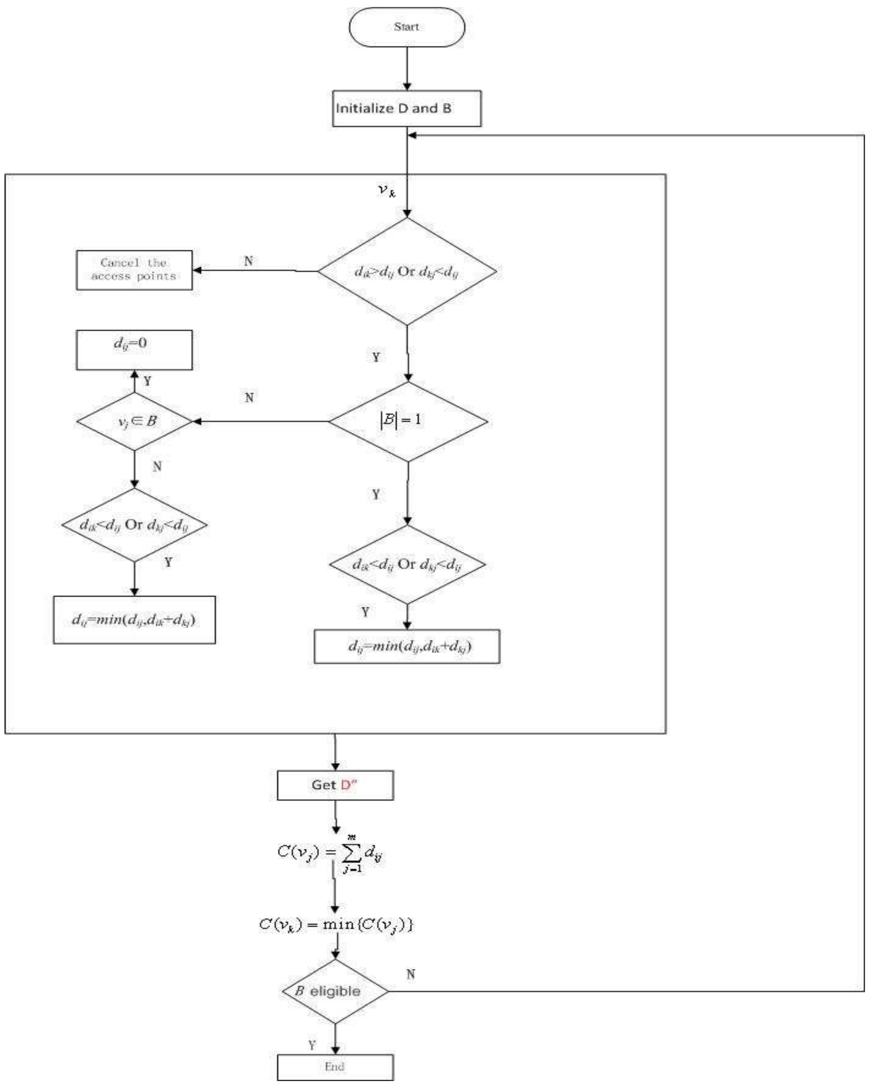
Start
Initialize D and B
v_k
$d_{ik}>d_{ij}$ Or $d_{kj}<d_{ij}$
N
Cancel the access points
Y
$|B|=1$
N
$v_j\in B$
Y
$d_{ij}=0$
N
$d_{ik}<d_{ij}$ Or $d_{kj}<d_{ij}$
Y
$d_{ij}=min(d_{ij},d_{ik}+d_{kj})$
Y
$d_{ik}<d_{ij}$ Or $d_{kj}<d_{ij}$
Y
$d_{ij}=min(d_{ij},d_{ik}+d_{kj})$
Get D''
$C(v_j)=\sum_{j=1}^{m} d_{ij}$
$C(v_k)=\min\{C(v_j)\}$
B eligible
N
Y
End

Fig. 3. The process of weights of Improved Floyd algorithm

The initial adjacency matrix

$$D^{(0)}=(w_{ij})_{9X9}=\begin{vmatrix} 0 & 10 & 7 & \infty & \infty & \infty & \infty & \infty & \infty \\ 10 & 0 & \infty & 5 & 5 & \infty & \infty & \infty & \infty \\ 7 & \infty & 0 & \infty & 6 & 9 & \infty & \infty & \infty \\ \infty & 5 & \infty & 0 & \infty & \infty & 2 & \infty & \infty \\ \infty & 5 & 6 & \infty & 0 & \infty & \infty & 12 & \infty \\ \infty & \infty & 9 & \infty & \infty & 0 & \infty & \infty & 8 \\ \infty & \infty & \infty & 2 & \infty & \infty & 0 & 7 & \infty \\ \infty & \infty & \infty & \infty & 12 & \infty & 7 & 0 & 5 \\ \infty & \infty & \infty & \infty & \infty & 8 & \infty & 5 & 0 \end{vmatrix}$$

We can get $\boldsymbol{D}^{(0)}$, k=1,2,…,9, by computing $\boldsymbol{D}^{n}$ from $\boldsymbol{D}^{n-1}$ through multiple iterations .

The sum of row 2nd is minimum among all 9 rows after adding elements of each rows in matrix $D^{(9)}$, so we choose v2 as basic version after the first iterative .

Based on the new set of base version$\{v0，v2\}$, repeat algorithm(3)~(5),

$$D^{(9)} = (d_{ij}^{(9)})_{9X9} = \begin{vmatrix} 0 & 10 & 7 & 15 & 13 & 16 & 17 & 24 & 24 \\ & 0 & 11 & 5 & 5 & 20 & 7 & 14 & 9 \\ & & 0 & 16 & 6 & 9 & 18 & 18 & 17 \\ & & & 0 & 10 & 22 & 2 & 9 & 14 \\ & & & & 0 & 15 & 12 & 12 & 17 \\ & & & & & 0 & 20 & 13 & 8 \\ & & & & & & 0 & 7 & 12 \\ & & & & & & & 0 & 5 \\ & & & & & & & & 0 \end{vmatrix}$$

$$D^{(9)} = (d_{ij}^{(9)})_{9X9} = \begin{vmatrix} 0 & 5 & 6 & 10 & 0 & 15 & 12 & 12 & 17 \\ 10 & 0 & 6 & 5 & 0 & 15 & 7 & 12 & 17 \\ 7 & 5 & 0 & 10 & 0 & 9 & 12 & 12 & 17 \\ 13 & 5 & 6 & 0 & 0 & 15 & 2 & 9 & 14 \\ 13 & 5 & 6 & 10 & 0 & 15 & 12 & 12 & 17 \\ 13 & 5 & 6 & 10 & 0 & 0 & 12 & 12 & 8 \\ 13 & 5 & 6 & 2 & 0 & 15 & 0 & 7 & 12 \\ 13 & 5 & 6 & 9 & 0 & 13 & 7 & 0 & 5 \\ 13 & 5 & 6 & 10 & 0 & 8 & 12 & 5 & 0 \end{vmatrix}$$

We choose v8 as basic version after the second iterative because the sum of row 8th is minimized in matrix D $^{(9)}$.

Considering the factors of stored data of versions and using 2 intermediate base versions, we end computing after two iterative calculations.

3.5 Algorithm Analysis

1) Effect of optimization. In order to study restore efficiency of multiple version through the definition of incremental version of double directions, we compute two intermediate base versions using directed acyclic graph structure and undirected graph structure respectively. Table 1 shows the comparison of optimized results under two structures.

Table 1 Comparison of optimized results under two structures in same multiple node system

multiple version structure	Directed acyclic graph	Undirected graph
Versions of intermediate base	*v7, v6*	*v5, v8*
Restore steps of multiple version	60	45

It shows that the version recovery cost is saved (15 steps in this example, nearly 1/4) and the version reduction efficiency is improved using incremental version definition with double directions restored mode in same multiple node system

At the same time, there are several version restoration paths to select in undirected graph, so security and reliability of the version data are improved distinctly.

Restoration of v2 conduct only based on v1 if the evolution of the version is directed. On the other hand, v2 can conduct also based on v5 and v1; meanwhile it can save 5 restoration steps if it is undirected in the multiple nodes example (Figure 3).

2) Calculation Analysis. The work is mainly in the calculation of the shortest path through distribution pattern optimization algorithm. The calculation of add operation is m^3 if there are m^2 nodes in the distance matrix for normal Floyd algorithm, because when calculating the distance between v_i and v_j, the addition operation $d_{ik}+d_{kj}$, is performed and then compared the sums with *dij* for each node.

Using the improved algorithm, computation of add operation is reduced greatly because we don't need to compare $d_{ik}+d_{kj}$ and d_{ij} when $d_{ik}>d_{ij}$ or $d_{kj}>d_{ij}$, as the fact of k<>*i, j* and the symmetry of distance matrix.

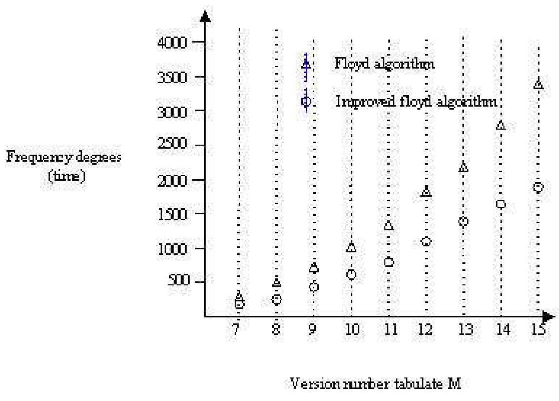

Fig. 4. Comparison in computation of improved Folyd Algorithm

Fig.4 shows that with nodes increased, the adding operations are increased. We can greatly reduce calculation using improved the Floyd algorithm, especially in numerous nodes system.

Conclusions

A new algorithm of weighted undirected graph structure for stored distribution optimization in multiple-node information system has been developed. Data security and reliability and version query efficiency of versions in multiple-node information system have been increased by middle complete base selected. This policy has been preliminary used in the project –Key technologies research and application in wireless sensor network in forest fire prevention and fire monitoring. The research is significant to the version management in coordinated graphic design of network.

Reference

[1] Zhu ming, Dou Wanfeng. The technology for improved multiple versions in coordinated graphics edit system [J], COMPUTER SYSTEMS, 2007.28(7):1318-1321.

[2] Shao Weifeng, Yang yang. The storage and query of multiple versions documents of XML in coordinated edit system [J], Computer Engineering, 2006.32(24)：75-77.

[3] Zhu Canshi. A kind of optimization method research for Warshall and Floyd algorithm [J], Computer and Modernization, 2010.(4)：43-45.