Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 21(1): 54-61, Mar. 2023 Regular paper

Optimization for Large-Scale n-ary Family Tree Visualization

Kyoungju Min, Jeongyun Cho, Manho Jung, and Hyangbae Lee*

Department of Sino-Korean Literature, Chungnam National University, Daejeon 34134, Korea

Abstract

The family tree is one of the key elements of humanities classics research and is very important for accurately understanding people or families. In this paper, we introduce a method for automatically generating a family tree using information on interpersonal relationships (IIPR) from the Korean Classics Database (KCDB) and visualize interpersonal searches within a family tree using data-driven document JavaScript (d3.js). To date, researchers of humanities classics have wasted considerable time manually drawing family trees to understand people's influence relationships. An automatic family tree builder analyzes a database that visually expresses the desired family tree. Because a family tree contains a large amount of data, we analyze the performance and bottlenecks according to the amount of data for visualization and propose an optimal way to construct a family tree. To this end, we create an n-ary tree with fake data, visualize it, and analyze its performance using simulation results.

Index Terms: Family Tree, Data Visualization, n-ary Tree Search, n-ary Tree Visualization, Digital Humanities

I. INTRODUCTION

In the study of the humanities, family trees are important for understanding the influence of relationships between individuals. Complex content, such as family, friendship, marital, and teacher-disciple relationships, can be visualized and understood in the form of a graph [1-3] or network diagrams [1,2,4] with loops. As the number of children is not specified, a family tree can be constructed and visualized as an n-ary tree [5]. Various humanities information is included in the information on interpersonal relationships (IIPR) [6] from the Korean Classics Database (KCDB) [7], and among them, 67K historic persons with clear sources and over 200K relational information are provided in the form of web services [6,7].

In this study, a database was built by crawling the IIPR data contained in the KCDB. After searching for a person using this database, a family tree of that person is automatically drawn. To accomplish this, we find the top parent of the search person, construct an n-ary tree with children from the root, and visualize it using d3.js [8].

Although there are hundreds of thousands of pieces of data on a family tree of a single family, the data of the IIPR in the KCDB are centered on the characters in the collection of books. For example, in the royal family tree of the Joseon dynasty, only thousands of people are registered in the database; therefore, a family tree was drawn. In addition, it is possible to intuitively understand the relationship by visually expressing the relationship between two pieces of personal data searched within the family tree.

To effectively visualize an entire family tree spanning more than hundreds of years, we calculate the time complexity using algorithms. We measure the performance according to the amount of data by dividing it by function and identify points where the bottleneck occurs. We analyzed the visualization performance of n-ary trees by automatically generating fake data similar to the real data for large-scale tree simulations. The simulation results reveal fewer than 10K nodes should be used when visualizing family trees. This is one of the recent attempts to interpret the humanities in a

Received 13 August 2022, **Revised** 18 October 2022, **Accepted** 18 October 2022
***Corresponding Author** Hyangbae Lee (E-mail: 5386yhbbb@hanmail.net)
Department of Sino-Korean Literature, Chungnam National University, Daejeon 34134, Republic of Korea

Open Access https://doi.org/10.56977/jicce.2023.21.1.54 print ISSN: 2234-8255 online ISSN: 2234-8883

new way by using computers in the field of digital humanities to determine hidden meanings [9,10] that could not be found with existing methods.

II. Related Works

A. Metro Map Tree

Traditional trees, including family trees, are familiar to users; however, when the amount of data increases, it is difficult to express it on a screen. To solve this problem, a metro tree map [11] is used to utilize space where the tree is drawn.

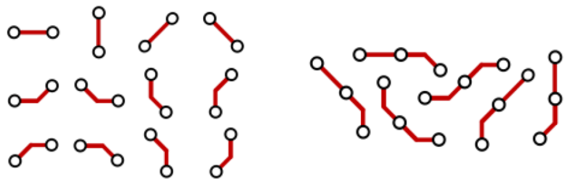

Fig. 1. (left) All 12 possible links and (right) examples of these combinations [11].

An example of all twelve possible types of links and combinations of links used in the metro tree map is illustrated in Fig. 1. This method was inspired by the London Underground (Metro) Map and makes excellent use of the empty space near the root of a traditional tree. In this study, only the experimental results of hundreds of people were presented; therefore, it is difficult to confirm the data of thousands or more, and there is a significant difference in the form of the traditional family tree.

In addition to this metro family tree, there have been studies on space utilization using stacked tree visualization [12]; however, there are difficulties in understanding traditional family trees and relationships.

B. Traditional Layered Tree

The traditional layered tree is used not only in family trees, but also in tree data structure expression. A layered tree is often used to express family trees in a Confusion culture by starting from the root at the top and drawing the children below. A representative web service of this type is the Joseon Dynasty Royal Family genealogy [13] of the Central Research Institute of Korean Studies. In this service, users can check the original images of the genealogy and the royal family tree.

One of the early branches of the family tree is illustrated in Fig. 2. Because there are tens of thousands of person data

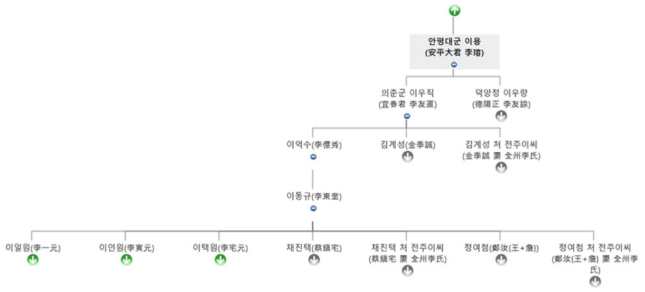

Fig. 2. Example of a Joseon Dynasty Royal Family tree.

points in the entire family tree, it cannot show all tree nodes. According to the user's mouse-click event, asynchronous JavaScript and XML (AJAX) data are received, and then tree nodes are added. This method has a disadvantage in that it is difficult to display the entire family tree and to understand the relationship between people because it requires too much space when displaying tens of thousands of data points on a screen.

III. DATA PROCESSING

A. Preparing Data

The procedure for generating a family tree in a database is conceptually represented in Fig. 3. The process of creating a family tree using a newly constructed database by crawling the IIPR in KCDB consists of four major steps. First, starting from the retrieved data *S* desired by the user, iteratively find the parent to find the root *R*. Second, it constructs an n-ary tree by recursively finding child nodes with the found *R* as the root. Third, creating a JavaScript Object Notation (JSON) file for tree visualization. Fourth, load a generated JSON file and visualize [14] family tree with d3.js.

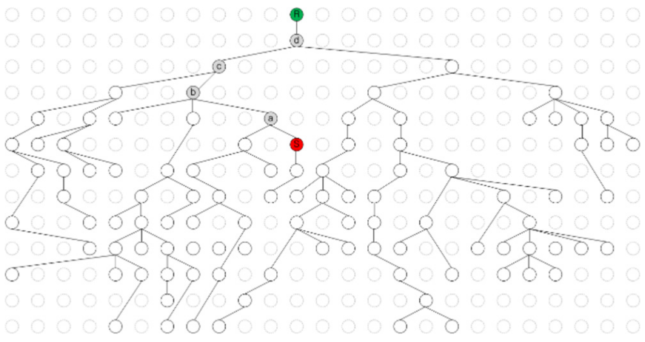

Fig. 3. Conceptual diagram of finding the top-level parent in a database and constructing the entire pedigree tree from the root.

Algorithm 1 expresses the first three steps of the four steps in preparing data with PHP in pseudo code.

Algorithm 1. Generating n-ary tree from database and generate JSON file for visualization with PHP

```
// Step 1. Find Root
$root = $searchKey;
while(true) {
   $sql = “Query for find root”;
   $data = db_fetch($sql);
   if($data[“father”])
      $root = $data[“father”];
   else
      break;
}

// Step 2. Find Child Nodes
$array = array();
if($root) {
   array_push($array, array(
      // add field here
      “children” => getChild($root)
   ));
}

// Step 3. Make JSON for Visualization
$json = json_encode($array[0],$JSON_OPTION);
save_json($json);

// Function Definition
function getChild($father){
   $sql = “Query for find children“;
   while($childData = db_fetch($sql)) {
      array_push($array, array(
         // add field here
         “children” =>
            getChild($childData[“pkey”])
      ));
   }
   return $array;
}
```

B. n-ary Tree Visualization

A partial view of the tree created with d3.js of the Song Si-yeol, a Confucian scholar in the late Joseon dynasty, family tree by loading the JSON file created in Step 3 above, is presented in Fig. 4.

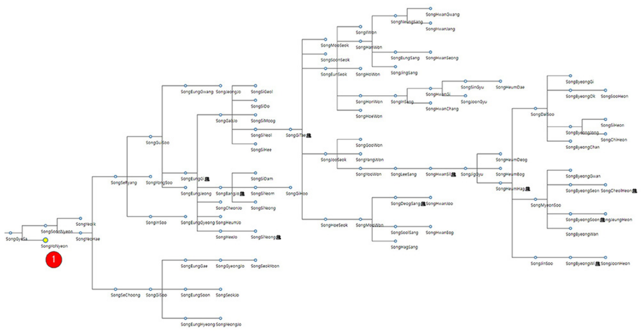

Fig. 4. Auto generated family tree of the Song Si-yeol family.

The part marked with ① in Fig. 4 indicates collapse subtree by clicking on the large, scaled tree node. Users can expand or collapse any subtrees they want.

C. Tree Shapes

Generally, a tree or pedigree is vertical, with parents at the top and children at the bottom. As the generation or depth of the tree increases, the number of nodes increases, and the interval between leaf nodes becomes narrower. In addition, the problem of overlapping texts of node data with horizontal writing in English or Korean is worse than that with vertical writing. Owing to the vertical writing of Chinese characters used in genealogy in the past, the family tree was expressed in a vertical form.

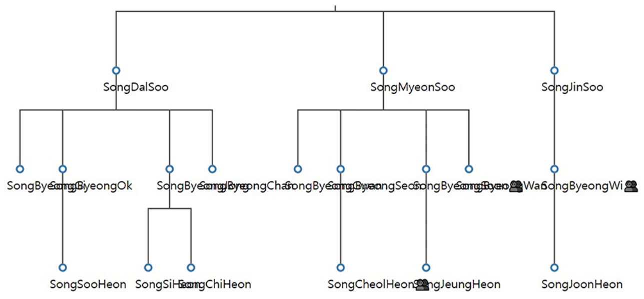

Fig. 5. Text overlapping phenomenon in vertical tree due to horizontal writing in English.

The overlapping of letters in the vertical tree is presented in Fig. 5. Compared with Fig. 4, when expressing a family tree, the horizontal type is preferable to the vertical type.

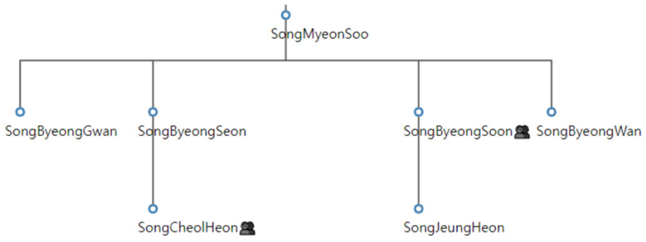

Fig. 6. Result of removing overlapping text by moving the node spacing away.

Fig. 6 presents an enlarged view of the middle subtree of Fig. 5 after removing the overlapping phenomenon by adjusting spacing between nodes. A much larger space is needed to express an entire family tree.

If we create a family tree using d3.js, a web-based JavaScript library, we can easily register and process events for each node. Users can check detailed information by hovering the mouse over the nodes, and the 👥 icon next to the name in Fig. 6 indicates that there is additional information such as wife, son-in-law, adopted son, and so on.

In traditional trees, straight lines connect nodes. In d3.js, the diagonal shapes can be expressed in various shapes, such as straight lines and curves, as illustrated in Fig. 7.

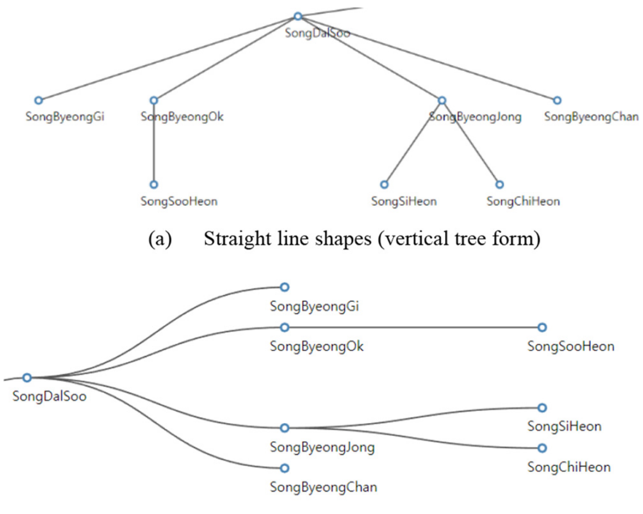

Fig. 7. Example of user-selectable tree line appearance.

D. Visualization of Interpersonal Relationship

It helps to find a person in a pedigree with a large amount of data, or to understand the relationship between two people. The number of hop counts in a Korean family tree is the simplest way to understand the influence relationship between two people. The fewer links between people in the family tree, the closer the relationship. For example, a parent-child has one hop (link), and a sibling requires two hops. The relationship between a person and their cousin has four hops: person - father - grandfather - uncle (father's brother) - cousin. A relationship connected by four hops in an n-ary family tree is called a 4-Chon (寸 , hops). Hence, the fewer hops in the family tree, the closer relationship.

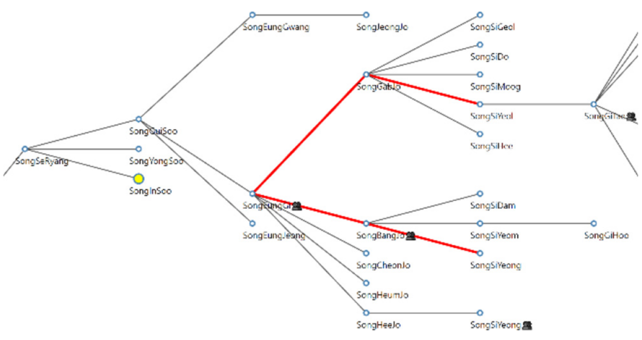

Fig. 8. Visualization result of the relationship between the two people.

An example of a visual representation of the relationship between two people is presented in Fig. 8. The larger the amount of data, the greater the visualization effect. As shown in Fig. 8, SongSiYeong is a cousin of SongSiYeol. Because there are four hops between SongSiYeong and SongSiYeol, this relationship is referred to as 4-Chon.

A three-step process to visualize relationship is as follows.

1) Step 1: Finding Node

Starting from the root node, it iteratively traverses the tree until it finds the source or destination node.

Algorithm 2. Recursive function definition for finding source or destination node from tree

```
function searchNode(curNode, findKey, who) {
   if(curNode. pkey == findKey){
      find = true;
      findNode = curNode;
      return curNode;
   }else if(curNode.children) {
      for(i=0;i<curNode.children.length;i++){
         if(curNode.children[i].pkey
               ==findKey) {
            findNode = curNode.children[i];
            return;
         }else {
            searchNode(curNode.children[i],
                           findKey, who);
         }
      }
   }
}
```

Algorithm 2 uses the "who" parameter to distinguish between source and destination, and the maximum search time for source and destination search in an n-ary tree is O(N) where the number of nodes is N, respectively.

2) Step 2: Finding Common Parent of Source and Target Nodes

In a tree implementing family tree, only the parent of the root node is null, and all other nodes have one parent information. Assume that the paths from the root to the source and destination nodes found in Step 1 are as follows.

Path to Source: Root - A - B - C - D - Src
Path to Destination: Root - A - B - E - F - G - Dst

In this case, finding the common ancestor B, Src - D - C - B - E - F - G - Dst becomes the shortest path.

Algorithm 3 describes the process for determining the best common ancestor and calculating the shortest path. The time complexity of finding the shortest path is O(D), where D is the maximum depth of the n-ary tree.

Algorithm 3. Finding common ancestor and calculate shortest path

```
// Find Best Common Ancestor
for( var s = srcArray.length -1,
         d = dstArray.length -1;
         s>=0, d>=0;
         s--, d--) {
   if(srcArray[s] == dstArray[d]) {
      commonNode = srcArray[s];
      topSrcIdx = s;
      topDstIdx = d;
   }else
      break;
}

// Shortest path to String : Source Side
for (var s = 0; s<srcArray.length; s++) {
   path.value += srcArray[s] + "-";
   if(srcArray[s] == commonNode)
      break;
}

// Shortest path to String : Dest Side
for (var d = dstArray.length -1; d >= 0; d-
-) {
   if(dstArray[d] == commonNode)
      findDst = true;
   if(findDst)
      path.value += dstArray[d] + "-";
}
```

3) Step 3: Update Links of Color and Stroke

As shown in Fig. 8, the color and stroke width of the link are changed using the iterative operation of d3.js. Algorithm 4 updates the links using an iterative operation, and the time complexity is O(ND), where N is the number of nodes and D is the maximum depth of the tree. That is, the link update time in the tree is proportional to the number of nodes.

Algorithm 4. Process of updating shortest path links

```
var linkUpdate = linkEnter.merge(link);
linkUpdate.transition()
   .duration(duration)
   .attr('d', d=> diagonal(d, d.parent))
   .attr('stroke', d => {
     if(topSrcIdx && topDstIdx) {
        for(var x = topSrcIdx-1; x>=0; x--) {
           if(srcArray[x] == d.pkey)
              return "#FF0000";
        }
        for(var x = topDstIdx-1; x>=0; x--) {
           if(dstArray[x] = d.pkey)
              return "#FF0000";
        }
     }
   }.attr('stroke-width', d => {
     // Almost same as attr('stroke') above
     // Change "#FF0000" to 5 for return val
   }
```

IV. ANALYSIS OF FAMILY TREE VISUALIZATION PERFORMANCE

A. Creating Fake Data

The IIPR in the KCDB contains information on 67K historical persons. Visualization is not an issue because a family typically contains fewer than 1K pieces of information in the IIPR. However, in order to measure performance by visualizing a family tree, the performance was measured by generating fake data similar to the real one. Using 181 surnames and 774 first names collected from the Internet, fake data of 100 to 100K were created in the same manner as in Algorithm 5, and 0–4 children were created and used for each node in the same way. In consideration of data duplication, random values of surnames were also used.

Algorithm 5. Generating fake data

```
// Function Definition
function generateFakeName()
{
   // fList (familyList), nList(nameList)
   $fList = "Smith,Johnson,Williams,Kim,…";
   $nList = "John,James,Henry,Thomas,…";

   $splitFamily = explode(",", $fList);
   $splitName = explode(",", $nList);

   $randNameNum =
       rand(0, count($splitName) -1);
   $randFamNum =
       rand(0, count($splitFamily)-1);

   $randName = $splitName[$randNameNum];
   $randFamily = $splitFamily[$randFamNum];

   return $randName . " " . $randFamily;
}
```

B. Environments

The performance was measured five times in msec unit on a localhost in the environment presented in Table 1, and the average value was obtained by measuring four decimal places.

Table 1. Environments for performance measurements

Item	Details	Remarks
OS	Windows 10	Enterprise Edition 64bit
Server	XAMPP 7.4.27	Apache 2.4.52 Maria DB 10.4.22 PHP 7.4.27
System	AMD Ryzen 9 5900X RAM 64GB	12-Core, 3.70GHz
Visualization	d3.js Version 4	
Testing	on a localhost	Average after 5 measurements

C. Performance and Bottleneck

The visualization process was divided by functions, and the performance of each function was measured to identify and improve functions that are bottlenecks when considering scalability. Functions are divided into JSON file loading, tree visualization, array processing for calculations, source node search, destination node search, common-root node search, and the tree update.

The measurement results in which the number of nodes on the x-axis is expressed in a log scale and the measurement time is linearly expressed on the y-axis are illustrated in Fig. 9. The time required to search for arbitrary nodes is almost the same as the search time for the source or destination, and the common-root search time is relatively small. In Algorithms 3 and 4, the time complexity is equal to O(D) and O(ND), respectively, and the search time has little effect on the overall performance.

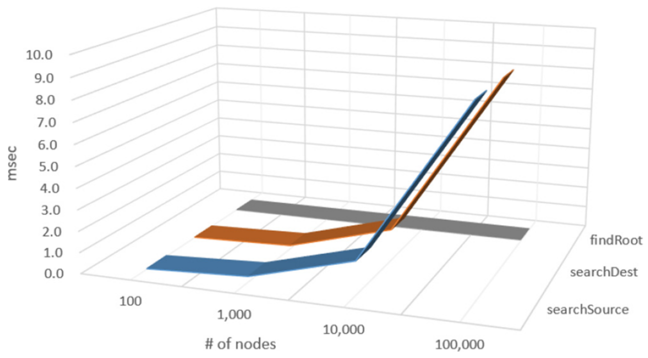

Fig. 9. Time measurement results retrieved to visualize relationships between nodes.

On the other hand, Fig. 10 presents the performance measurement results for functions that affect performance as the number of nodes increases, and both the x and y-axis are expressed in log scale.

The tree visualization feature has the most impact if a tree has more than about 1K nodes. This is because of the characteristic of performing repetitive operation of d3.js.

Computation time increases linearly in proportion to the increase in the number of nodes. JSON loading, initial tree visualization, and array processing for operation are necessary processes when starting the program. On the other hand, the tree-update operation is a necessary function for expanding or reducing a tree, moving an entire tree after searching, and visualizing between two nodes.

In the result, the JSON file loading is staggered compared with the other functions. This appears to be an error in the measurement process owing to the caching phenomenon for improving the performance of web browser.

Although the tree-update process is frequently called when

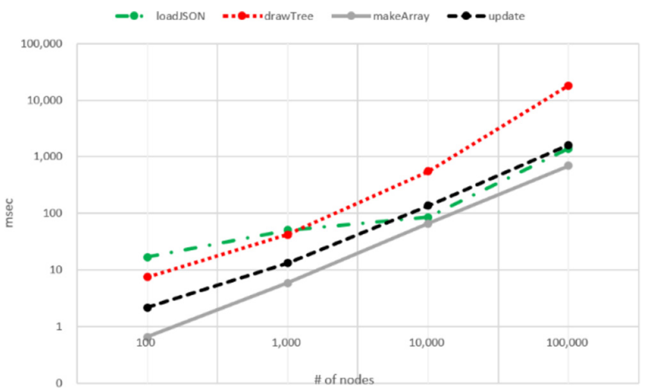

Fig. 10. Performance measurement results of functions that have a serious impact on scalability.

using the created tree, it must be processed within a tolerable level of waiting for the user to use the massive family tree. If it uses a tree with more than approximately 20K nodes, users must wait for more than 3 seconds for the family tree initialization process. Considering the relatively high performance of the system used in this experiment, it is desirable to use it with 10K nodes or fewer. This is consistent with the experimental results using graphs to visualize the relationships between people in previous studies [1,2]. This is because all the features illustrated in Fig. 10 become a bottleneck in visualization considering scalability.

D. Directions for Family Tree Visualization

The genealogy of Confucian culture, including that of Korea, has been recorded for more than several hundred years, focusing on family members. Usually, all people from 30 to 50 generations are recorded; therefore, there are data on hundreds of thousands of people. Experiments have revealed that displaying them on a single screen is impossible. There is a performance problem if users do not use a subtree with thousands to 10K nodes or fewer. This requires saving person information in a database, searching for the person that users want to display on the screen, and then visualizing the people around the person searched for. For example, it is necessary to reduce the amount of data using a partial tree, such as the 4th generation of parents and children, centering on the searched person. However, this method has a limitation in that the data to be used for new visualizations must be collected every time, focusing on the searched person. This is left for future research considering engineering and humanities implications.

E. Humanities Meaning

Previous studies that visualize the relationship network between people using large-scale graphs are very helpful for character exploration. However, the family tree also serves as an important humanities tool for determining a person's influence relationships. To date, humanities researchers have manually worked on paper to create family trees. The KCDB has been regarded as a treasure trove of the humanities, including various classical information that humanities classics researchers can use. They received a lot of help when drawing a family tree using the IIPR of the KCDB.

However, the automatic family tree system developed in the current study can dramatically shorten the time spent by humanities researchers and can be used to create accurate family trees. It also provides a visual representation of the relationships between people on the pedigree, helping to understand relationships intuitively. In addition, the visualized family tree has the advantage that humanities researchers can directly check the data integrity of the KCDB. This will be a source technology for digitizing genealogy to accurately express the pedigree information of a family.

V. CONCLUSIONS AND FUTURE WORKS

Family trees are often used to accurately understand the influence of historical persons. Until now, humanities classics researchers have drawn family trees manually and used them for their research, which requires considerable time. The IIPR in KCDB contains data on approximately 67K people and more than 200K relationships. In this study, a new database was built by crawling IIPR, and all family trees were automatically created. Since a lot of data are displayed on one screen, we tried to visually express the relationship between the two persons searched to understand the influence of interpersonal relationships more easily and use it for research.

In Confucian cultures, including Korea, the family tree is written in Chinese characters, and it is common to write it in a vertical tree form under the influence of the vertical writing of Chinese characters. However, to digitize the family tree, the horizontal tree form has been shown to be effective for space utilization because the characters can be written horizontally. In addition, to intuitively express the traditional family tree or degree of intimacy, the user can decide on various types of trees, such as straight and curved.

D3.js, which is easy to utilize events, was used for large-capacity family tree representation and intuitive understanding of users. The performance was measured by dividing it into several functions such as JSON file loading, drawing tree, array processing, visualization of search result, and update tree. The measured results confirmed that the bottleneck when visualizing a large-scaled tree by testing with 100 to 100K nodes each occurred in data loading, drawing tree, and updating tree according to user actions. The time complexity of each function was verified using the algorithm, and the measurement results were almost identical.

For future research, it seems desirable to present subtrees after searching so that they have thousands of nodes or fewer to be used in the actual family tree. For this purpose, it should be improved in consideration of humanities usability and engineering performance.

REFERENCES

[1] K. J. Min, "Plan for the improvement of interpersonal relationship network in the Korean Classics DB," *Journal of Korean Classics*, vol. 59, pp. 197-241, 2021. DOI: 10.15752/itkc.59..202111.197.

[2] K. J. Min, B. C. Jin, and M. H. Jung, "Massive graph expression and shortest path search in interpersonal relationship network," *Journal of Korea Institute of Information and Communication Engineering*, vol. 26, no. 4, pp. 624-632, Apr. 2022. DOI: 10.6109/jkiice.2022.26.4.624.

[3] Y. J. Nam and J. H. Park, "Big data and network analysis on genealogy focusing on martial relationships of Kimhae Kim's family," *Journal of Digital Convergence*, vol. 17, no. 11, pp. 39-51, Nov. 2019. DOI: 10.14400/JDC.2019.17.11.039.

[4] S. H. Pak, W. S. Yoon, and H. B. Chang, "A study on the analysis of related information through the establishment of the National Core Technology Network: Focused on display technology," *Journal of Society for e-Business Studies*, vol. 26, no. 2, pp. 123-141, May. 2021. DOI: 10.7838/jsebs.2021.26.2.123.

[5] K. Daghan, "A study on data visualization utilizing tree structure in 3D space," Ph. D. dissertation, Seoul National University, KR, 2019.

[6] Information of Interpersonal Relationship, Aug. 2022. [Internet]. Available: https://db.itkc.or.kr/people.

[7] Korean Classics DB, Aug. 2022. [Internet] Available: https://db.itkc.or.kr.

[8] E. Meeks, *D3.JS in Action: Data Visualization with JavaScript*, 2nd ed. Pennsylvania, PA: Manning Pub., 2017.

[9] S. Bekmirzaev, T. H. Kim, and B. C. Lee, "Pairwise similarity analysis and quality estimation on classical Chinese poetry of ancient Korea in 15th century," *International Journal of Applied Engineering Research*, vol. 12, no. 23, pp. 13884-13890, 2017.

[10] K. J. Min and B. C. Lee, "The analysis of Chosun Dynasty poetry using 3D data visualization," *Journal of Korea Institute of Information and Communication Engineering,* vol. 25, no. 7, pp. 861-868, Jul. 2021. DOI: 10.6109/JKIICE.2021.25.7.861.

[11] J. Korst, V. Pronk, and J. J. van Wink, "A visualization of family relations inspired by the London Metro Map," *Proceedings of the 13th International Symposium on Visual Information Communication and Interaction*, Article No 8, pp. 1-8, Dec. 2020. DOI: 10.1145/3430036.3430065.

[12] H. J. Schulz, "Treevis.net: A tree visualization reference," *IEEE Computer Graphics and Applications*, vol. 31, issue 6, pp. 11-15, Oct. 2011. DOI: 10.1109/MCG.2011.103.

[13] Joseon Royal Family Genealogy, Sep. 2022. [Internet] Available: https://visualjoseon.aks.ac.kr/jb/item?#/node?gubun=king

[14] K. J. Min and M. H. Jung, "Development of digital Daedongyeojido (Great Map of Korea) and humanities software development direction," *Journal of Korean Literature in Chinese,* vol. 62, pp. 205-240, 2022. DOI: 10.17260/jklc.2022.52.205.

Kyoungju Min
received a B.S. degree in 2000 and M.S. degree in 2002 from the Department of Computer Engineering in Chungnam National University, Korea. He is currently a Ph.D. student at the Department of Sino-Korean Literature of Chungnam National University. He has been a CEO of Eureka Solution since 2008. He is also active as an IT professional instructor. His current research interests include digital humanities, data analysis, data visualization, and network security.

Jeongyun Cho
received a B.S. degree in 1996 from the Department of Sino-Korean Literature of Chungnam National University, Korea, an M.S. degree in 1998 from the Department of Korean Language and Literature in Chungnam National University, and a Ph.D. in 2009 from the Department of Korean Language and Literature in Korea University, Korea. Since 2001, she worked as a lecturer in the Department of Sino-Korean Literature at Chungnam National University, where she now works as an assistant professor. Her research interests include Sino-Korean prose, Sino-Korean literary criticism, and digital humanities.

Manho Jung
received a B.S. degree in 1997, an M.S. degree in 1999 from the Department of Sino-Korean Literature in Chungnam National University, Korea, and a Ph.D. in 2005 from the Department of Sino-Korean Literature in Dankook University, Korea. Since 2000, he has worked as a lecturer and research professor in the Department of Sino-Korean Literature at Chungnam National University, where he has worked as an associative professor since 2019. His research interests include Sino-Korean grammar, artificial intelligence classics translation, and digital humanities.

Hyangbae Lee
received a B.S. degree in 1994 from the Department of Sino-Korean Literature of Chungnam National University, Korea, an M.S. degree in 1996 from the Department of Korean Language and Literature in Chungnam national University, and a Ph.D. in 2000 from the Department of Sino-Korean Literature in Dankook University, Korea. He has worked as professor in Department of Sino-Korean Literature at Chungnam National University since 2001. His research interests include Sino-Korean criticism and digital humanities.