

An Adaptive Algorithm of Data Network Traverse Based on Node Analysis

Yan XU¹, Yi ZHENG^{2,*}, Yu ZHANG³, Hui LIU⁴, Hao CUI⁵ and Hai-lin LIU⁶

^{1, 2, 4, 5, 6} Shandong Provincial Key Laboratory of Ocean Environment Monitoring Technology

Shandong Academy of Sciences Institute of Oceanographic Instrumentation
National Marine Monitoring Equipment Engineering Technology Research Center
Qingdao 266001, China
Email: ZhengY@sdsa.org

³ Department of Ophthalmology, Qingdao Municipal Hospital, Qingdao 266001, China

Keywords: Data network, Ternary tree, Traverse.

Abstract. In this paper we describe a data network traverse algorithm. It traverses based on ternary tree and has Depth First Search (DFS) feature. The data network modeling employs three types of components: Line, Node and Intersection. Each type has typical data structure to connect other type. This algorithm is self-adaptable for random data network and could achieve dynamic analysis with high speed and accuracy. It could be widely used in modeling, analysis and inspection of industrial pipeline such as circuit, water path and gas path.

Introduction

The industrial pipeline modeling is one of the most important phases in engineering design field. An effective method for computer aided modeling and analysis could improve the safety and robustness of the work. In this paper we describe an algorithm which could complete the random data network analysis based on ternary tree [1, 2] on VB6.0 platform. The algorithm could inspect the attribute of each Line, Node and Intersection. This process could achieve dynamically to acquire the data in detail.

The Data Network Modeling

Three different types of components are employed in the modeling: Line, Node and Intersection. Each type has been signed in figure 1. In initial figure Node is invisible and the diamond shapes are used to enhance the understanding. The description of each type is in following:

Line is the path of data network, it describes the data transmit.

Node connects two Line in network, it describes the data transmit by pairing array to construct the traverse direction.

Intersection is the node crossing by 1 to 4 Lines (and more) in the data network, it is in counter clockwise order and describes the data transmit by two dimensional array. The Intersection could be divided into 3 types: Variable Intersection, Immutable Intersection and Input/Output Intersection.

Variable Intersection is a node which could be controlled by data network system. Immutable Intersection could not be controlled. Input/Output Intersection is a node crossing by only one Line and used for the Input/Output traverse judgment.

The data network modeling example is in figure 1:

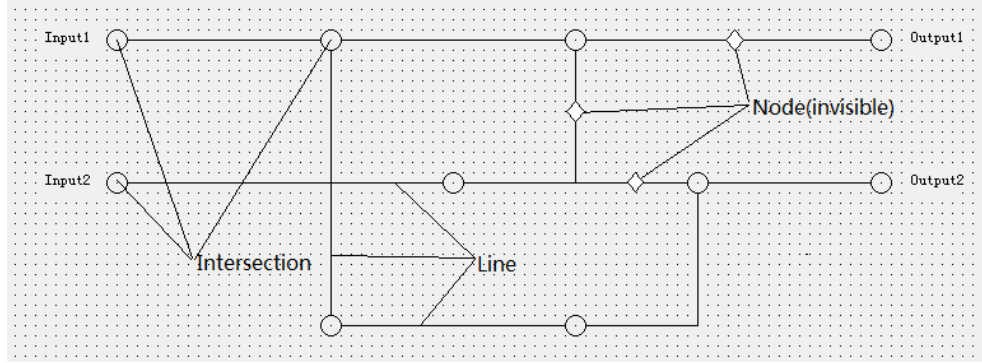


Figure 1. The data network modeling

After modeling, the connection of each component should be built by database or net list. Then the traverse and analysis could be executed based on the connections.

The data structure for Line is not mandatory. The essential information of Line could be stored in the pairing array of Node. The corresponding data structure and mathematical method could also be employed if necessary.

The data structure for Node is pairing array. This array stores the paired Lines of each Node and describes the connection and traverse direction for each Line. Its data structure is in table 1:

Table 1. The Data Structure for Node

Dimension 1: Node Number	Dimension 2: Node Line
0,1,2,...N	Line1
	Line2

The data structure for Intersection is two-dimensional array. In this algorithm the Intersection array includes 9 members. It describes the status of the corresponding Intersections and Lines. New member could be added to fulfill more requirements. The data structure for Intersection is in table 2:

Table 2. The Data Structure for Intersection

Dimension 1: Intersection Number	Dimension 2: Intersection element
0,1,2,...N	Intersection status flag
	Line1
	Line1 attribute
	Line2
	Line2 attribute
	Line3
	Line3 attribute
	Line4
	Line4 attribute

In the Intersection array, the record position for each Line is not mandatory. We only need to pay attention that the Line connection should be built in counter clockwise order.

The Function of Data Network Traverse

This section introduces three types of functions in data network traverse.

The Trigger Function of Traverse

The primary work is to determine the Input/Output Intersection in data network modeling. Theoretically the network could be built with one or more Input Intersections and any number of Output Intersections. The trigger function is necessary. It is the starting point for attribute transmits.

The traverse begins with the execution of pairing array including the Input Intersection Line. The trigger could be achieved by control button.

The code section of trigger function is in figure 2:

```
b = 1 'Input attribute
For counter1 = 0 To 11
  If gasctrl(counter1, 9) = "Shape1" Then 'Input Intersection
    For counter2 = 0 To 3
      If Len(gasctrl(counter1, counter2 * 2 + 1)) > 0 Then
        a = gasctrl(counter1, counter2 * 2 + 1)
        gasctrl(counter1, counter2 * 2 + 2) = b
      End If
    Next
  End If
Next
layer = 0
Call lineconnect(a, b)
Call nodetransfer(outlinetemp)
Call pipetest
```

Figure 2. The code section of trigger function

The Pairing Function of Node

The Node traverse is based on pairing function. This function could search all Lines in pairing array and acquire the corresponding Node. Then it could find the corresponding Output Line and complete the attribute transmit.

The code of pairing function is in figure 3:

```
Public Sub lineconnect(inlinetemp, inlinep)
  If Len(inlinetemp) = 0 Then
    outlinetemp = ""
    Exit Sub
  End If
  For Counter = 0 To 12
    If couple(Counter, 0) = inlinetemp Then
      outlinetemp = couple(Counter, 1)
    End If
    If couple(Counter, 1) = inlinetemp Then
      outlinetemp = couple(Counter, 0)
    End If
  Next
  For counter1 = 0 To 11
    For counter2 = 0 To 7
      If gasctrl(counter1, counter2) = outlinetemp Then
        gasctrl(counter1, counter2 + 1) = inlinep
      End If
    Next
  Next
End Sub
```

Figure 3. The code of pairing function

The Traverse Function of Intersection

The traverse of Intersection alternately works with pairing function in order to keep strict direction. It executes by iteration based on ternary tree. The function body employed recursive invocation to realize the code reuse. The stack array is used to store the traversed Lines. The termination condition of iteration is also conducted.

The code section of Intersection function is in figure 4:

```

For counter1 = 0 To 11
  For counter2 = 0 To 3
    If gasctrl(counter1, counter2 * 2 + 1) = inline Then
      inp = gasctrl(counter1, counter2 * 2 + 2)

      Select Case counter2 * 2 + 1
        Case 1
          If gasctrl(counter1, 0) = False And _
            (Len(gasctrl(counter1, 3)) > 0 Or Len(gasctrl(counter1, 5)) > 0 Or Len(gasctrl(counter1, 7)) > 0) Then
            gasctrl(counter1, 4) = inp
            gasctrl(counter1, 6) = inp
            gasctrl(counter1, 8) = inp
            temp(layer, 0) = gasctrl(counter1, 3)
            temp(layer, 1) = gasctrl(counter1, 5)
            temp(layer, 2) = gasctrl(counter1, 7)
            Call lineconnect(gasctrl(counter1, 3), inp)
            For counter5 = 9999 To 0 Step -1
              For counter6 = 0 To 2
                If Len(temp(counter5, counter6)) > 0 And temp(counter5, counter6) = outlinetemp Then
                  Exit Sub
                End If
              Next
            Next
            If Len(inline) = 0 Then
              Exit Sub
            End If
            Call nodetransfer(outlinetemp)
          End If
        End Select
      End If
    End For
  End For
End For

```

Figure 4. The code section of Intersection function

The Realization of Traverse Algorithm

This algorithm has the feature of Depth First Search (DFS) [3]. For example, the ternary tree has three leafs: a₁, a₂ and a₃. The parameter Layer describes the index of the stack array. ε describes the iteration operation of the first leaf. ε' describes the operation of the other leafs, the operation includes the transmit, judgment and storage. The iteration of each layer could be represented by equation (1):

$$D_n = [a_1^n \quad a_2^n \quad a_3^n] \times \begin{bmatrix} \varepsilon \\ \varepsilon' \\ \varepsilon \end{bmatrix} \quad (1)$$

It is assumed that the expectation of iteration layer is N, the iteration of one leaf could be represented by equation (2):

$$D_N = a_1 \prod_{\text{layer}=0}^N \varepsilon^{(\text{layer})} + \sum_{\text{layer}=0}^N \varepsilon'^{(\text{layer})} (a_2^{(\text{layer})} + a_3^{(\text{layer})}) \quad (2)$$

It is assumed that the total of ternary tree leafs is L. The iteration beginning from each leaf could be presented by $D_N(S)$ $S=0, 1, 2 \dots L$, so the traverse of whole data network could be represented by equation (3):

$$\xi_N^L = \sum_{S=0}^L D_N(S) \quad (3)$$

The meaning of ε and ε' could be determined as we need, it is available during the whole traverse. In the realization, several principles should be followed:

- (1) Fundamental principle: counter clockwise order
- (2) Determine the initial status of each Line, Node and Intersection.
- (3) A complete positive traverse should be executed if the Input Intersection is triggered.

- (4) With the change of Intersection status, a negative traverse and a positive traverse should be executed in order.
- (5) The negative traverse should begin with the initial status of Output Intersection.
- (6) Various mathematical methods could be employed in attribute transmit according to the requirements of inspection and modeling.

The Result of Simulation

This simulation is conducted on PC: Intel Dual-Core E5500 CPU, 2M memory. The OS is Win7 and the Language is Visual Basic 6.0.

Experiment 1: Traverse the whole data network in figure 1 and measure the cost of execution time. All the Intersections are on “OPEN” status, the first Intersection in first line is Input Intersection and its initial attribute is 1.0. Considering this paper will display in binary image, the width of Line could change into wider (BorderWidth = 8) if the Input attribute transmit completes without any error. The result is in figure 5:

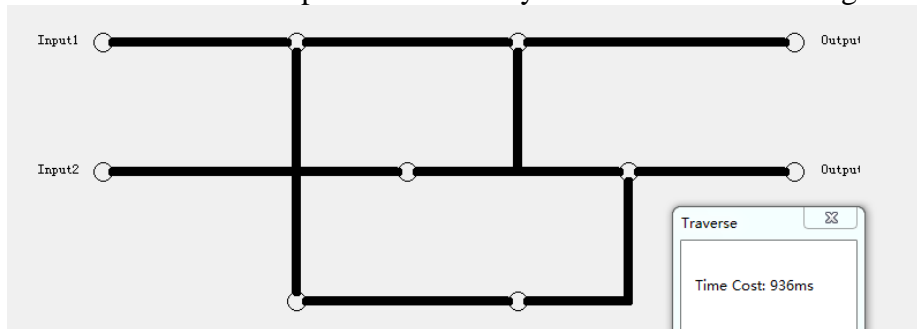


Figure 5. The simulation result of Experiment 1

In figure 5 it is proved that the data network traverse could successfully achieve based on ternary tree iteration. It costs 936 ms time. Each Line, Node and Intersection works correctly.

Experiment 2: Change the status of the third Intersection in first line in figure 1 from “OPEN” to “CLOSE”. Traverse the whole data network again and measure the cost of execution time. The Input Intersection is the same as Experiment 1. The result is in figure 6:

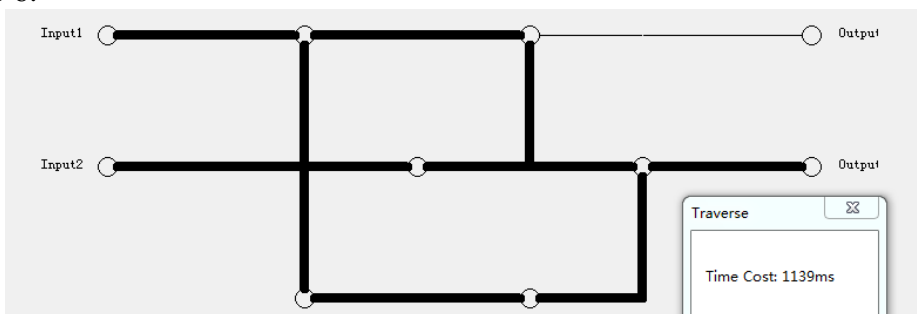


Figure 6. The simulation result of Experiment 2

In figure 6 it is proved that the traverse with changed initial status could also achieve successfully. It costs 1139 ms time. The status of each Line, Node and Intersection behind the changed Intersection will not change with traverse. It shows the correct result as we need.

Conclusion

This paper describes an adaptive algorithm for data network traverse based on node analysis. This traverse algorithm is based on ternary tree and has DFS feature. The data network modeling includes three types of components: Line, Node and Intersection. We built the data structure for each type and made verification through two experiments. The simulation proves that the algorithm is adaptive; it could achieve the intelligent traverse and dynamic analysis for random data network with high speed and accuracy.

Additionally, as a computer aided modeling method, this algorithm could optionally inject new type of component and data structure. We could design more Intersection status and data transfer mechanism to fulfill more requirements. More professional application could also be proposed by combining with existing algorithms [4, 5].

Acknowledgement

This research was financially supported by Youth Science Foundation of Shandong Academy of Sciences (2014QN031).

References

- [1] Wei-min Yan, Wei-min Wu, Data Structure, Tsinghua University Press, Beijing, 2002.
- [2] Zhuo-qun Xu, Data Structure, The Open University of China Press, Beijing, 2002.
- [3] Xue-gang Hu, Data Structure algorithm design guidance, Tsinghua University Press, Beijing, 1999, pp. 198-216.
- [4] Xiao-min Yu, Li Tang, Xiao-kun Yu, Solve the problem of circle path in graph by searching, Journal of Qiqihar University, 2004, 20(2)44-46.
- [5] Zi-li Tang, The method for uniquely determining a tree or a binary tree based on its traversal sequences, Journal of Chinese Computer Systems, 2001, 22(8)985-988.