Research on Big Data Analytics by Using High-Level Fuzzy Petri Nets

Cheng-chao QIU
Institute for Interdisciplinary Research, Jianghan University, Wuhan, China

Keywords: Big data, Fuzzy system, High-level fuzzy Petri nets.

Abstract. The aim of this study is to apply the theory of high-level fuzzy Petri nets (HLFPN) to big data analytics platform. The platform features the following advantages: 1) it enables to describe analytical contents through natural language approaches; 2) it can be used to verify analytical processes through modular approaches; 3) it enables to promote fuzzy theory and solving problems through nonlinear equations; 4) it can be employed to generate Map/Reduce programs automatically through the system; 5) it can be used for parallelization, thereby shortening analysis time; and 6) it enables to inquire results through an interface. Finally, we describe the experiments conducted to verify the functions of the platform.

Introduction

Background

Petri net theory was proposed by Dr. Carl Adam Petri in 1962. The Petri net is a graphical and mathematical modeling tool, which is concurrent, asynchronous, distributed, parallel, nondeterministic, and stochastic; and it can be used to model and analyze various systems[3]. However, along with the advances in the information technology, the description of a Petri net is more and more complex. Therefore, scholars conduct their researches with evolutionary Petri net theories one after another, such as colored Petri net [4], timed Petri net [5], fuzzy Petri net[6], high-level fuzzy Petri net[7], [8], and so on. This thesis adopted the HLFPN(high-level fuzzy Petri nets) to make a decision. It provides the characters of a Petri net and fuzzy theory, which can be used to express fuzzy rules and conduct fuzzy reasoning with fuzzy rules.

Motivation and Purposes

This thesis aims to propose a Big Data analysis by using high-level fuzzy Petri nets (HLFPN). We apply HLFPN to make some data analysis.

Initially, the aim of this study is to simulate Matlab fuzzy control systems [1] and develop an HLFPN module and a program generator for performing data analyses similar to that of [2]. However, there are some drawbacks. First, the more data we need to analyze, the longer it takes to perform the data analysis. Second, writing a Hadoop Map/Reduce program is very difficult and complex. Therefore, this study aims to develop the HLFPN system on the cloud platform.

An ideal big data analysis system should offer complete services to customers. It should provide with the following items: basic storage devices (structured or unstructured), data acquisition methods (streaming data or static data), analytical methods (real time analysis or learning analysis), user privacy and data security, and an user interface in which customers can control it correctly and show the results of data visualization.

Customers might come from different kinds of fields such as doctors can analyze their patients’ health care records; biologists can analyze DNAs; and business owners can analyze product sales and customer’s behavior. All of them need different data structures and analysis methods. A well customized big data platform should have the ability to handle different kinds of data structures and analytical methods.

Using the HLFPN model as a modular and analysis tool associated with the following four advantages: 1) it offers more flexible learning capability because it is able to model both IF–THEN and IF–THEN–ELSE rules; 2) it allows multiple heterogeneous outputs to be drawn if they exist; 3) it offers a more compact data structure for fuzzy production rules so as to save information storage; and 4) it is able to learn faster due to its structural reduction.
Methodology Development

Definitions

1) **Definition 1:** The HLFPN is defined as an eight-tuple

\[ \text{HLFPN} = (P, T, F, C, V, \alpha, \beta, \delta) \]

where

- \( P = \{p_1, \ldots, p_i\} \) A finite set of places.
- \( T = \{t_1, \ldots, t_l\} \) A finite set of transitions.
- \( P \cup T \neq \emptyset \) Called the flow relation and is also a finite set of arcs, each representing the fuzzy set (i.e. fuzzy term) for an antecedent or a consequent; where the positive arcs (i.e. THEN parts) are denoted by →.
- \( C = \{X, Y, Z\} \) A finite set of linguistic variables, e.g. \( X,Y, \text{and } Z \), where \( X = \{x_1, x_2, \ldots, x_n\} \), \( Y = \{y_1, y_2, \ldots, y_m\} \), \( Z = \{z_1, z_2, \ldots, z_q\} \)
- \( V = \{v_1, v_2, \ldots, v_i\} \) A finite set of fuzzy truth values known as the fuzzy relational matrix between the antecedent and the consequent of a rule.
- \( \alpha: P \rightarrow C \) Associations function, mapping from places to linguistic variables. \( \alpha(p_i) = \{c_i\}, i = 1, \ldots, I \) where \( C = \{c_i\} \) is a set of linguistic variables in the knowledge base (KB) and is the number of linguistic variables in the KB.
- \( \beta: F \rightarrow [0,1] \) Associations function, mapping from the flow relations to the fuzzy truth values between zero and one.
- \( \delta: T \rightarrow V \) An association function, mapping from transitions to fuzzy relational matrices.

2) **Definition 2:** Input and Output Functions

- \( I(t) = \{p \in P | (p, t) \in F\} \) A set of input places of transition \( t \).
- \( I(p) = \{t \in T | (t, p) \in F\} \) A set of input transitions of place \( p \).
- \( O(t) = \{p \in P | (t, p) \in F\} \) A set of output places of transition \( t \).
- \( O(p) = \{t \in T | (p, t) \in F\} \) A set of output transitions of place \( p \).

3) **Definition 3:** Negation

In an IF-THEN-ELSE rule, the ELSE part is indicated by a negation arc \( \circ \rightarrow \) and the fuzzy set in the antecedent must be complemented and is denoted by \( \square \).

4) **Definition 4:** Membership Function

The mapping function \( \text{Mem}(p) : P \rightarrow [0,1] \) assigns each place a real value, where \( \text{Mem}(p) = \text{DOM}(\alpha(p)) \), DOM represents the degree of membership in the associated proposition and data tokens are available in the set \( P \) of places.

5) **Definition 5:** Max-Min Compositional Rule

In the HLFPN, \( \forall \) transition \( t \), \( V(t) = \min \{ \text{fuzzy sets } \text{inf}(t) \} \); \( \forall \) place \( p \), \( V(p) = \max \{ \text{fuzzy sets } \text{inf}(p) \} \). The Max-Min composition operator is denoted by \( \square \).

6) **Definition 6:** Input Place, Hidden Place, and Output Place

In the HLFPN, \( \forall \) place \( p \in P \), if \( \forall t_i \in T \), \( p \in O(t_i) \), then \( p_i \) is called an input place (IP) of \( t_i \); if \( \forall t_i \in T \), \( p_i \not\in I(t_i) \), then \( p_i \) is called an output place (OP) of \( t_i \); otherwise, \( p_i \) is called a hidden place.

7) **Definition 7:** SISO, SIMO, MISO, MIMO

There are four types of relationships in the HLFPN shown as follows:

1) SISO represents the single-input -single -output, i.e.,
   \( \forall t_i \in T, \ |I(t_i)| = 1 \) and \( |O(t_i)| = 1 \).

2) SIMO represents the single-input -multiple -output, i.e.,
   \( \forall t_i \in T, \ |I(t_i)| = 1 \) and \( |O(t_i)| > 1 \).

3) MISO represents the multiple -input -single -output, i.e.,
∀t\(_j\) \in T, \quad |I(t\(_j\))| > 1 \quad \text{and} \quad |O(t\(_j\))| = 1.

4) MIMO represents the multiple-input - multiple-output, i.e.,
∀t\(_j\) \in T, \quad |I(t\(_j\))| > 1 \quad \text{and} \quad |O(t\(_j\))| > 1.

8) \textbf{Definition 8: Cyclic HLFPN}
In the HLFPN, for the subnet or the whole net, if an \(IP\) or \(OP\) is not empty and a path exists, then we call it a cyclic HLFPN.

9) \textbf{Definition 9: Heterogeneous Outputs}
In the HLFPN, if outputs possess different attributes, then we call them heterogeneous outputs.

\textbf{Fuzzy Reasoning}

Let \( R \) be a set of fuzzy production rules, where \( R = \{R_1, R_2, \ldots, R_n\} \). The general form of the \( i \)th fuzzy production rule \( R_i \) is shown as follows:

\( R_i : \text{IF} \quad d_1 (X \text{ is } A), \quad \text{THEN} \quad d_2 (Y \text{ is } B); \quad \text{ELSE}, \quad d_3 (Z \text{ is } C) \ldots (V) \).

where “ \( X \text{ is } A \)”, “ \( Y \text{ is } B \)”, and “ \( Z \text{ is } C \)” are propositions; \( X \) is called the input linguistic variable; \( Y \) and \( Z \) are called the output linguistic variables, respectively; \( A \) is called the input fuzzy set; \( B \) and \( C \) are called the output fuzzy sets, respectively; the fuzzy truth values of the propositions “ \( X \text{ is } A \)”, “ \( Y \text{ is } B \)” and “ \( Z \text{ is } C \)” are restricted to \([0, 1]\); “ \( X \text{ is } A \)” is the antecedent of the fuzzy production rule \( R_i \); “ \( Y \text{ is } B \)” and “ \( Z \text{ is } C \)” are the consequences of the fuzzy production rule \( R_i \); Let \( V \) represent the fuzzy relational matrix between the antecedent and the consequent of a fuzzy production rule.

\textbf{Fuzzy Reasoning Algorithm}

Here, we briefly review a fuzzy reasoning algorithm (FRA) to determine whether there exists a fuzzy relational matrix between the antecedent and the consequent of a fuzzy production rule.

\textbf{INPUT:} \( \text{Mem}(p\(_i\)) \quad \forall p\(_i\) \in IP \), where \( IP \) denotes a set of input places.

\textbf{OUTPUT:} \( \text{Mem}(p\(_i\)) \quad \forall p\(_i\) \in OP \), where \( OP \) denotes a set of output places.

\begin{tabular}{|l|l|}
\hline
\textbf{Step 1:} & Initially, assume that only the Degree of Memberships (DOMs) in the propositions operating on input variables are available. Consequently, the initial marking function is shown as follows:
\( M(p\(_i\)) = 0 \), if \( p\(_i\) \notin IP \); \( M(p\(_i\)) = \text{the number of data tokens} \), if \( p\(_i\) \in IP \). \\
\hline
\textbf{Step 2:} & \quad \forall t\(_j\) \in T, compute
\( V(t\(_j\)) = W_a \times W_e = (w_{a1}, w_{a2}, \ldots, w_{an}) \times (w_{e1}, w_{e2}, \ldots, w_{em}) \), where \( T \) denotes a set of transitions; \( V(t\(_j\)) \) is a fuzzy relational matrix between the antecedent and the consequent of rule \( t\(_j\) \); \( W_a = \{w_{a1}, w_{a2}, \ldots, w_{an}\} \) is a fuzzy set of weights for the antecedent; \( W_e = \{w_{e1}, w_{e2}, \ldots, w_{en}\} \) is a fuzzy set of weights for the consequent; and each element of a fuzzy set is denoted by a fuzzy weight interval.
\hline
\textbf{Step 3:} & Input a data pattern \( W_{\text{\{input\}}} \).
\hline
\textbf{Step 4:} & Fire the enabled transitions. Let \( t\(_j\) \) be any enabled transition. Then, compute \( t\(_j\) \in T / \forall p\(_k\),
\( M(p\(_k\)) = \text{the number of data tokens} \), \( W_a = W_{\text{\{input\}}} \), \( W_e = W_a \odot V(t\(_j\)) \) or \( -W_a \odot V(t\(_j\)) \) if an ELSE part is available.
\hline
\textbf{Step 5:} & For every output variable \( O \), its associated membership distribution is \( W'_o = \{w'_o\} = \bigvee w'_i, \quad i = 1, 2, \ldots, I \), where \( I \) is the in degree of output variable \( O \). Then, \( W_e \) becomes an actual output.
\hline
\textbf{Step 6:} & Go back to Step 4, while \( \exists t\(_j\) \in T / M(p\(_i\)) = 1 \quad \forall p\(_i\) \in I(t\(_j\)) \).
(That is, while the enabled transitions still exist, go to Step 4.)
\hline
\textbf{Step 7:} & The weighted average defuzzification method is applied, and the real operating value is computed.
\hline
\textbf{Step 8:} & The end.
\hline
\end{tabular}
Fuzzy Rule Parser Module

The operating processes are illustrated in Fig. 1.

![Figure 1. Operating process.](image)

This module succeeds the fuzzy rule module and receives an error dialog if users entered inaccurate or undeclared variables in the previous module. If the entered datasets are accurate, the module proceeds to six successive functions, the details and operations of which are described as follows:

[Step 1] Parse fuzzy rule: The fuzzy rules entered by users are analyzed, yielding variable values and the related fuzzy rule to acquire the types of PN modules.

[Step 2] Create new Petri net (PN) module: Continuing from the previous step, through the file-editing approach, the preset number, names, and connection modes of the .xml variables are converted into parameters.

[Step 3] Save PN module as .xml files: The existing edited or unedited PN modules are stored in the initially opened .xml files.

[Step 4] .xml file parse: The parser is applied to the .xml files, and the identifications (IDs) are recorded. The IDs are the codes in the .xml files, namely, places, transitions, and arcs; and are the keys to identifying the .xml files. No elements are permitted to have the same IDs. If two or more elements have the same IDs, the previous element may be overlapped by the next, or reading errors may result.

[Step 5] Add PN module to .xml: The edited the .xml text files from the previous step, including the rule parameters and names of variables, are copied and pasted into the text files in Step 3. During the process, the IDs of the elements are added by 1.

[Step 6] Open the .xml file again: Opening the edited text files reveals the newly added PN modules on the screen. Users can edit the PN modules according to their preferences. Moreover, the values of the places in the modules are variable in the rules and are identifiable.

Experiments and Discussion

The motivations of this experiment are to confirm 1) that the big data analytics platform operates correctly; (2) that fuzzy membership functions can be set up on the platform; (3) that HLFPN can be used and modularized; (4) that HLFPN can be converted into parallel programs; and (5) that the results can be shown on the platform and compared with the stand-alone version of HLFPN, and verify their differences.
Experimental Results

This experiment applied the analytical approach and process. However, the number of datasets in the present study is 20 times that analyzed. Furthermore, this study is divided into a normative HLFPN experiment (a control group) and a parallelized HLFPN experiment (an experimental group). The differences in time, usage, and processes between the two groups are investigated, and the advantages and disadvantages of a normative HLFPN and a parallelized HLFPN are compared with each other.

Normative HLFPN Experiment (Control Group)

The aforementioned process and architecture are used in this experiment, and the targeted datasets are stored in the internal database. The results are mainly focused on recording time. The experimental actions and consuming time are shown in Table 1.

<table>
<thead>
<tr>
<th>Action</th>
<th>Approximate Consuming Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write Program (catch RSI, WMS, PSY values)</td>
<td>20 minutes</td>
</tr>
<tr>
<td>Write Program (make MFs and fuzzy rule)</td>
<td>30 minutes</td>
</tr>
<tr>
<td>Write Program (fuzzy reasoning)</td>
<td>40 minutes</td>
</tr>
<tr>
<td>Write Program (defuzzify)</td>
<td>20 minutes</td>
</tr>
<tr>
<td>Analyze (2.5 Terabytes (TBs))</td>
<td>5340 minutes</td>
</tr>
<tr>
<td>Total Time</td>
<td>5450 minutes (90 hours)</td>
</tr>
</tbody>
</table>

As shown in Table 1, the normative HLFPN program may take more than one hour. If the program plus a big data analysis, it may be more than five hours. Next, we use the same process in our analysis platform.

Parallelized HLFPN Experiment (Experimental Group)

The parallelized HLFPN experimental actions and consuming time are shown in Table 2.

<table>
<thead>
<tr>
<th>Action</th>
<th>Approximate Consuming Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declare Variables (catch RSI, WMS, PSY values)</td>
<td>3 minutes</td>
</tr>
<tr>
<td>Declare Variables (define MFs and fuzzy rule)</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Generate Program (automatically run fuzzy reasoning)</td>
<td>1 minutes</td>
</tr>
</tbody>
</table>

As shown in Table 2, the parallelized HLFPN experiment uses the HLFPN system to decrease the time of writing programs. Users can use the HLFPN system to declare the variable, define the membership function, enter the fuzzy rule and generate the map/reduce program automatically. The results from the parallelized HLFPN compare advantageously with the normative HLFPN experiment.

Conclusion

This study applies the theory of high-level fuzzy Petri nets (HLFPN) to the big data analytics platform. The platform features the following advantages: 1) it enables to describe analytical
contents through natural language approaches; 2) it can be used to verify analytical processes through modular approaches; 3) it enables to promote fuzzy theory and to solve problems through nonlinear equations; 4) it can be employed to generate Map/Reduce programs automatically through the system; 5) it can be used for parallelization, thereby shortening analysis time; and 6) it enables to inquire results through an interface.

This platform can be improved further in the future. In addition to exploring approaches to collect big data, whether the proposed platform can be employed for structured data, unstructured data, and streaming analyses will be investigated further to provide better service interfaces, additional functions, and different analytical contents.

References