Received: 16 January 2017 | Revised: 10 October 2017 | Accepted: 24 November 2017

DOI: 10.1111/exsy.12261

WILEY Expert Systems

ORIGINAL ARTICLE

Evaluation of the benefits of using a backward chaining decision support expert system for local flood forecasting and warning

Xiaoyin Zhang[1] | Gary P. Moynihan[1] | Andrew N. S. Ernest[2] | Joseph L. Gutenson[3]

[1] Department of Civil, Construction, and Environmental Engineering, The University of Alabama, Tuscaloosa, USA

[2] Department of Civil Engineering, The University of Texas – Rio Grande Valley Brownsville, Texas, USA

[3] Hydrologic Systems Branch Coastal and Hydraulics Laboratory, U.S. Army Engineer Research and Development Center Vicksburg, Mississippi, USA

Correspondence
Gary P. Moynihan, Department of Civil, Construction and Environmental Engineering, The University of Alabama, Tuscaloosa, Alabama 35497-0205, USA.
Email: gmoynihan@eng.ua.edu

Abstract

Nationwide flood forecasting and warning are available through mass media. However, running the complex numerical models requires enormous computational resources. In addition, the comparatively low accuracy of prediction for a certain region such as a small town, a community, or a single house, causes false alarms and improper responses and thus the unnecessary loss of property and/or life. One potential solution to advance forecast accuracy without occupying substantial computational resources is to develop a stand-alone local flood forecasting and warning expert system incorporating sufficient data on local hydraulic and hydrological factors and local historical experience. To date, there has been a limited amount of work developing expert systems in this area. In this paper, we discuss the development and implementation of an expert system for local flood forecasting and warning. With its extensible knowledge base combined with the information provided by users, this expert system provides reasoning routines and forecasting on the flood warning stages, possible consequences, and recommendations for community managers, landowners, or the public in general.

KEYWORDS

backward chaining, expert system, flood forecasting, python

1 | INTRODUCTION

Flood incidents can endanger human life, cause extensive property damage, and result in significant harm to the environment. To attenuate the risk and reduce the loss caused by flood accidents, flood forecasting has been studied and developed throughout human history. Although global or nationwide flood forecasts and warnings are available through mass media, the comparatively low accuracy of prediction for a certain region causes false alarms, improper responses, and therefore unnecessary loss of property and/or life. One conventional method to improve the accuracy is to increase the resolution or decrease the based cluster size. Either way, the occupancy of computational resources must be increased enormously. The higher the resolution and the smaller the cluster size, the more computing power is needed. Another alternative method is to develop stand-alone systems only for small regions. Recent examples include using ensemble numerical weather prediction systems for medium-range flood forecasting (Cloke & Pappenberger, 2009); applying data-driven approaches, such as traditional artificial neural networks, adaptive neuro-fuzzy inference systems, wavelet neural networks, and hybrid adaptive neuro-fuzzy inference systems with multiresolution analysis using wavelets to develop models for hourly run-off forecasting at Casino station on the Richmond River in Australia (Badrzadeh, Sarukkalige, & Jayawardena, 2015); coupling meteorological observations and forecasts with a distributed hydrological model to advance flood forecasting in Alpine watersheds (Jasper, Gurtz, & Lang, 2002); coupling HEC-HMS with atmospheric models for predicting watershed run-off in California (Anderson, Chen, Kavvas, & Feldmand, 2002); and combining multimodels for operational forecasting for river basins in the Western United States (Najafi & Moradkhani, 2015). Although the models or systems listed above provided overall better performance for the whole river basins or catchments examined in the cited studies, the accuracy of forecasting for a small place such as a small town, a little community, and a specific house was not mentioned or was completely ignored. The reason is the same: To obtain accurate forecasting for a comparatively small place, the resolution of the entire studied region of the river basin or catchment must be higher (Luo, Xu, Jamont, Zeng, & Shi, 2007). More detailed local situations must be collected

and considered, more memory space must be allocated, and a heavier computational burden must be loaded onto the models that already have vast amounts of meteorological, hydrologic, and hydraulic data to analyse through complicated calculations (Cloke & Pappenberger, 2009; Fang, Xu, Zhu, et al., 2014b). In fact, most incidents begin and end locally and are managed at the local level (DHS, 2013). The most useful data are locally collected, although it is correlated with the data from outside the specific region.

In the meantime, the complicated numerical models or systems employed in recent studies considerably deplete available computing power (Li, Wang, & Wang, 2014; Wang, Xu, Bi, & Xu, 2014). Broadly, most numerical models can be categorized as physics-based models or data-driven models. The physics-based models represent the intricate hydrological cycle and transform precipitation into channel flow through hydrologic and hydraulic routing. Oftentimes, these classical rainfall–run-off models with complex mathematical formulations require high computation times (Garcia-Pintado et al., 2015). In contrast, data-driven models, for example, time series models, focus on the variation of hydrological variables with time and input–output stochastic processes instead of the mechanism of the rainfall–run-off transformation (Badrzadeh et al., 2015; Fang, Xu, Pei, et al., 2014a; Xu, Liang, & Gao, 2008). Generally, the stochastic processes entail the transformation and analysis of big data and thus consume dramatic computing power (Santos & Martinho, 2017; Wang et al., 2014).

Understandably, decreasing the scales of these catchment-wide or river-wide models and systems of flood forecasting to even smaller local sizes and limiting the usage of numerical models can decrease the engagement of computational resources. Therefore, local flood forecasting systems without sophisticated numerical models are more cost-effective for smaller places, especially for those small communities with limited budgets.

2 | BACKGROUND

With the purpose of improving forecasting and forecasting-based services, the World Meteorological Organization (WMO) has initiated various programmes and projects. The "Manual on Flood Forecasting and Warning" (referred to as the WMO manual hereafter) published in 2011 is one of the crucial outcomes. This manual provides the fundamental knowledge and guidance to develop or to set up applicable and tailored systems for flood forecasting and warning. In addition, the manual offers extensive references to further sources of information in both paper and online formats (WMO, 2011).

A local flood forecasting and warning system can be programmed as an expert system with inferential logic. Expert systems are one successful form of artificial intelligence technology that emulates the decision-making ability of a human expert by utilizing knowledge represented primarily as "if-then" rules (Jackson, 1998). Typically, an expert system consists of an inference engine and a knowledge base. An inference engine, also referred to as a general-purpose shell, is created by information technology (IT) specialists to simplify and expedite the programming process. A knowledge base is deduced and compiled by knowledge engineers and domain experts in a certain domain to store pertinent facts and rules. A knowledge base is where an expert system gains power, asserted by Edward Feigenbaum (1977), the father of expert systems. An inference engine, working primarily in either a forward chaining or backward chaining mode, applies logic rules to generate new knowledge. Forward chaining, driven by known data, works top-down to assert conclusions or new facts. Backward chaining, driven by goals, works bottom-up to determine what facts must be asserted (Hayer-Roth, Waterman, & Lenat, 1983). Conventionally, a local flood forecasting and warning system can also be procedurally coded in a traditional procedural language, such as an assembly language or a high-level compiler language (C, Pascal, COBOL, FORTRAN, etc.). In this coding process, IT specialists are required from beginning to end. However, in the coding process of an expert system, IT specialists are not necessary once an inference engine is packaged. Domain experts can work independently or cooperate with knowledge engineers to develop their specific expert systems in various fields. Thus, expert systems can be more rapidly and easily developed and maintained. Expert systems have greater flexibility to run with evolving goals (da SilvaAvanzi, Foggiatto, Santos, Deschamps, & Loures, 2017; Wong & Monaco, 1995).

The most popular computer languages for programming expert systems include Visual Basic (Spyridakos, Pierakos, Metaxas, & Logotheti, 2005), Java or JESS (Java Expert System Shell; Robindro & Sarma, 2013), CLIPS (Ooshaksaraie & Basri, 2011), MATLAB or NETLAB (Mounce, Boxall, & Machell, 2010), Visual Rule Studio (Chau & Phil, 2004), ART*Enterprise (Leon, Martin, Elena, & Luque, 2000), and PyKE (Python Knowledge Engine; PyKE, 2015).

The bottleneck to encoding knowledge in structured computer programmes is knowledge acquisition. Collecting and compiling the necessary knowledge in even a small area involve the effort of countless individuals (Comas et al., 2003; Kaewboonma, Tuamsuk, & Kanarkard, 2013). Recently, a considerable amount of research utilizing artificial intelligence or expert system technology has been done on flood management (Emerton et al., 2016; Fang et al., 2015; Ghalkhani, Golian, Saghafian, Farokhnia, & Shamseldin, 2012; Luo et al., 2007; Mabrouk, Ezziyyani, Sadouq, & Essaaidi, 2015; Mahabir, Hicks, & Fayek, 2007; Pinto et al., 2015). However, in the local flood forecasting and warning research community, there is a lack of scholarship on expert systems without complex numerical models.

To take advantage of logic programming and the concept of facts and rules, we collect and assimilate local hydraulic and hydrological situations, both local and global historical flooding records, and distilled the wisdom of experts into our systems as the knowledge base or database. By matching the case facts and global facts with rules, the inferential logic determines the flooding forecasting and warnings directly. Simply speaking, the whole process is similar to using weather lore, for example, "Red sky at night, sailors' delight. Red sky at morning, sailors take warning." When we see a red sky in the morning, we get the forecast and warning of a storm or bad weather. The following sections will provide a more scientific explanation of our system.

In addition, to benefit from other existing systems, the new systems should be able to read the output data from those large systems as well as the user input data directly. Moreover, the new systems should adapt to other small places in the same region or in other regions if needed. Most importantly, the development and operation of the new systems must occupy less computational resources for a much shorter time and be economically feasible for smaller districts.

3 | MATERIALS AND METHODS

Knowledge identified as necessary for local flood forecasting and warning was obtained from an extensive literature review and assimilated into machine-readable formats. To compare the benefits of inferential logic embedded in expert system shells with the procedural logic of conventional codes and the utility of backward chaining with forward chaining, we selected an expert system shell-PyKE that was capable of both forward and backward chaining inferential logic and developed three conceptual systems: a conventional procedural pseudocode, a forward chaining expert system framework and a backward chaining expert system framework. Based on the analysis of the conceptual systems, we decided to turn the backward chaining framework into a complete backward chaining expert system for local flood forecasting and warning. In this section, we introduce the materials and methods used to develop these three Local Flood Forecasting and Warning Systems (LFFWS).

3.1 | Goals

LFFWS consist of two phases: the training phase and the determining phase. In the training phase, LFFWS collect the local hydraulic and hydrologic conditions, historical records, and heuristic expertise to realize the goal (G_{set}): training the system by setting up the variables and parameters for the next determining phase. In the determining phase, the well-trained LFFWS learn the current or proposed situations by interviewing the users to achieve the goal (G_{fore}): making the forecast and warning by matching and comparing current situations with the stored variables and parameters.

In the training phase, LFFWS gather data such as (a) the depth of past severe floods in the local area; (b) the causes of flooding in the local area; (c) the speed at which the stream flow might rise; (d) the length of time floodwater might remain in the locality; and (e) the direction of the flood flow. We categorize these data into three types of triggers: triggers related to rainfall, triggers related to stream flows, and triggers related to local conditions. Correspondingly, we name the goals to set up those variables and parameters regarding the three types of triggers as Goal Rainfall (G_{rain}), Goal Flow (G_{flow}), and Goal Local (G_{local}), respectively.

In the determining phase, after LFFWS collect sufficient data about the current situation, LFFWS determine a comma-separated values (CSV) formatted report on the warning stage (G_{stage}) and warning messages (G_{m1}, G_{m2}, and $G_{m...}$ corresponding to various triggers). Figure 1 demonstrates the relations between these goals.

3.2 | Knowledge

In LFFWS, the knowledge, covering local hydraulic and hydrologic conditions, historical records, and heuristic expertise of local flood forecasting and warning, was classified into two primary categories: facts and rules. Facts are simple statements containing data values that represent and show relationships among entities; rules are declarative knowledge linking sets of premises and conclusions (Chen, Chau, & Kabat, 1985). The expert system shell or data repository used to support procedural logic decides on the format needed to assimilate knowledge primitives.

To make LFFWS stable, flexible, and sustainable, facts are categorized as static global facts and dynamic case facts. Those general and common facts applicable to all scenarios are symbolized as global facts, whereas other specific information about each particular case is denoted as case facts. A chain of questions performed as placeholders represent those dynamically provided case facts in the knowledge base. Currently, three types of questions would be obtained: (a) current or proposed accumulation and intensity of rainfall; (b) current or proposed water depth, velocity, and rise rate of streams; and (c) historical or recorded thresholds of rainfall and stream flow at different stages. LFFWS ask the third type of questions when the system must be reset. In addition to the three essential series of questions mentioned, a unique case ID, rain gauge ID, and stream gauge ID will also be requested to specify different scenarios and locations after the introductory screen. The reasoning rules are named corresponding to the goals they prove. For example, R_{set} and R_{fore} are two main sets of rules to prove G_{reset} and G_{fore}, respectively. Specifically, R_{rain}, R_{flow}, and R_{local} are sets of rules to prove G_{rain}, G_{flow}, and G_{local}, respectively. R_{stage}, R_{m1}, R_{m2}, and $R_{m...}$ are sets of rules to prove G_{stage}, G_{m1}, G_{m2}, and $G_{m...}$. Figure 2 illustrates the relations between these rules. The global facts and rules are extracted from the literature and other authoritative information sources (WMO, 2011).

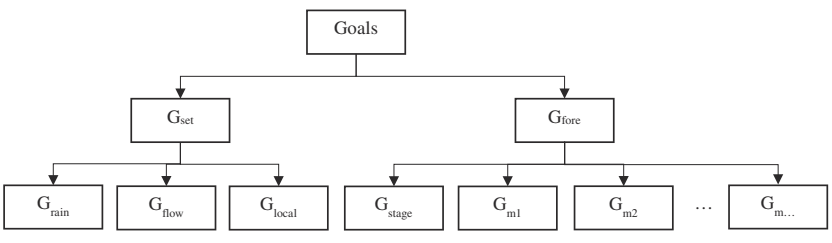

FIGURE 1 Goals of Local Flood Forecasting and Warning Systems

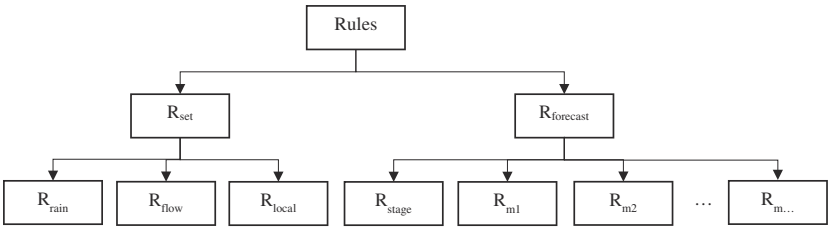

FIGURE 2 Rules of Local Flood Forecasting and Warning Systems

3.3 | System architectures

In the system architecture of the procedural code shown in Figure 3, questions, case facts, and rules are incorporated into the operation structure (e.g., from querying information to asserting case facts to proving goals in a fixed sequence: $G_{rain} \rightarrow G_{flow} \rightarrow G_{local} \rightarrow G_{stage} \rightarrow G_{m1} \rightarrow G_{m2} \rightarrow \ldots \rightarrow G_{m...}$ but are detached from global facts (referred to as the database). The system architectures of forward chaining and backward chaining expert systems are, respectively, shown in Figures 4 and 5. Unlike the procedural code, inference engines assemble operation structures separated from the knowledge base containing questions, case facts, global facts, and rules. Goals in expert systems are proved in parallel. The forward chaining inference engine begins by gathering all available information; however, backward chaining starts from the goal selection and collects necessary information for the certain goals. The backward chaining inference engine searches for the needed data from the existing case facts or previous analyses first, then interviews users for the remainder (if there is any) according to the query rules. After new case facts are asserted, the engine proves particular goals with all related case facts, global facts, and reasoning rules.

3.4 | Language

The backward chaining expert system for local flood forecasting and warning is developed with a combination of Python and its Knowledge Engine PyKE because (a) unlike a compiled language, Python, as an interpreted language, allows quick "ad hoc" development once the code is published and deployed; and (b) PyKE provides a way to directly "programme in the large". The first two advantages speed up programming an expert system with a vast knowledge base such as LFFWS by reusing code and reducing loop. (c) In addition to forward-chaining logic, PyKE includes backward-chaining logic, which enables interactive data acquisition. This capability is the key to the comparison of the expert systems with forward and backward chaining inference engines, respectively. (d) Python is open source and free. Any software based on Python is friendly to almost all platforms including Windows, Mac, Linux, iOS, and Android. This flexibility helps create a large market for LFFWS (PyKE, 2015; Python, 2015). (e) Extensive libraries are available. For instance, the Graphic User Interface of our systems can be built on its library Kivy to allow quick and easy interaction design and rapid prototyping (Kivy, 2015). Better performance for user practice can take advantage of this benefit.

4 | USE OF THE SYSTEM

The following example of case study demonstrates the functionality and context for use of the LFFW system. The data and associated scenario are suggested by Lueckemeyer and Lueckemeyer (2017). Assume that community emergency management personnel are monitoring the approach of an oncoming hurricane. The specific concern is directed toward a small neighbourhood of older wooden structures, located on an adjacent floodplain. The community emergency management personnel initiate the LFFW system to obtain guidance on if the specific area should be evacuated.

Figure 6 depicts the banner screen displayed upon entry to the system. Inputting "Y" loads the system reset screens, including resetting triggers of rainfall, water level, and local conditions shown in Figure 7. Once the user selects any one, two, or all three types of triggers to reset the characteristics and parameters, a corresponding parameter-resetting screen appears. For example, if the user selects stream water level triggers to reset, the system then conducts an interview about the stream gauge and the threshold of flood warning stages corresponding to the stream gauge. Figure 8 illustrates the conversation about resetting water level triggers between the system and system trainer, a local hydrologist, or an experienced flood response official. According to the historical records of the specific floodplain area, the average gauge height read from the nearest stream gauge was 0.5 m. When the gauge height approached 5 m (flood watch stage), there was the possibility of some flooding. If the gauge height reached 7 m (flood warning stage), flooding was expected and would cause disruption. If the water level kept rising and the gauge height was about 8 m (severe flood warning stage), serious flooding was expected, and there was an imminent danger to life and property. After the flood, the water level would recede. When the gauge height became lower than 4 m (all clear stage), no flood watches or warnings were in force any longer (USGS, 2017; WMO, 2011).

The user can either confirm all the triggers that were reset (shown in Figure 9) or decide to use existing default values by selecting no triggers to reset. The system then moves to the second phase to collect necessary and available situational data. Assume that the user monitors the news regarding water levels, which are represented by gauge heights. Gauge heights are updated every 15 min. After latest gauge height of 5.34 m is input, LFFW issues a "Flood Watch" with a simple explanation to this stage. Several recommendations are given in a CSV report shown in Table 1, as well. Concern regarding other warning stages and crisis awareness motivates the user to further investigate LFFW by projecting the

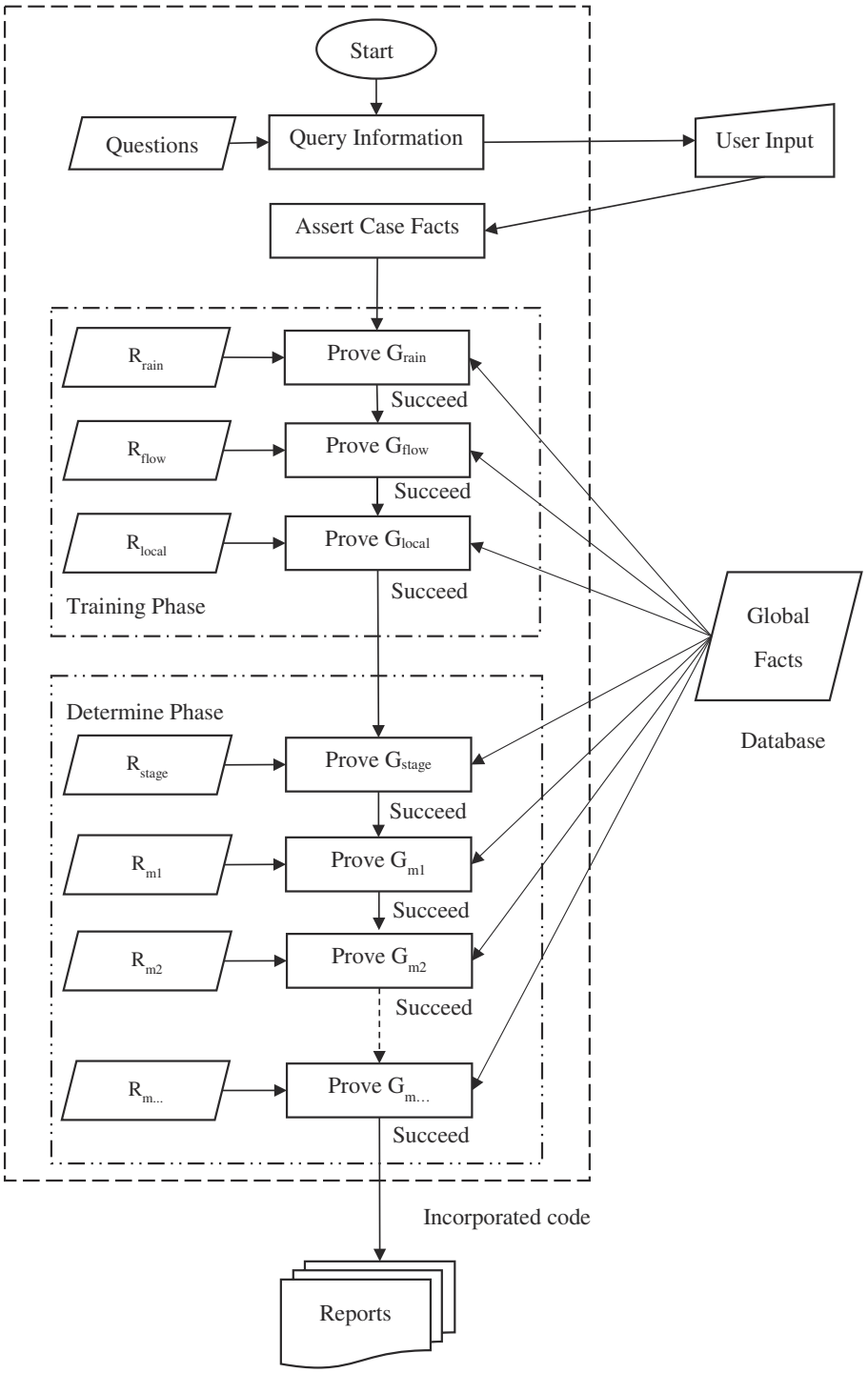

FIGURE 3 Architecture of the conventional procedural code of Local Flood Forecasting and Warning Systems

effects of increasing the water levels from 7.52 to 8.5 m. Correspondingly, LFFW generates CSV reports (shown in Tables 2 and 3) to provide a brief introduction to the stages of flood warning and severe flood and suggested critical actions. Subsequently, the river is observed to crest, and the water level starts to fall. Around midnight, the gauge height is anticipated to be 3.87 m. With the guidance of the LFFW system output (e.g., report shown in Table 4), the community emergency manager recognizes that the "All Clear" stage is now in effect and the neighbourhood homes are safe.

This simplified case demonstrates that LFFW is an applicable local flood forecasting and warning system. It can be easily tailored to work for a specific sensitive area (e.g., neighbourhood in a portion of floodplain) and fragile infrastructure (e.g., dated wooden houses) by initializing the system with historical local data. It is worth mentioning that triggers, such as water level, are not limited to determining the warning stages. Adding the thresholds of significant happenings can provide users with more thorough guidance. For example, the gauge height at which water flows out of channel onto the floodplain and submerges specific low-lying roads. People should avoid driving or walking on the possible flooded roads, keep

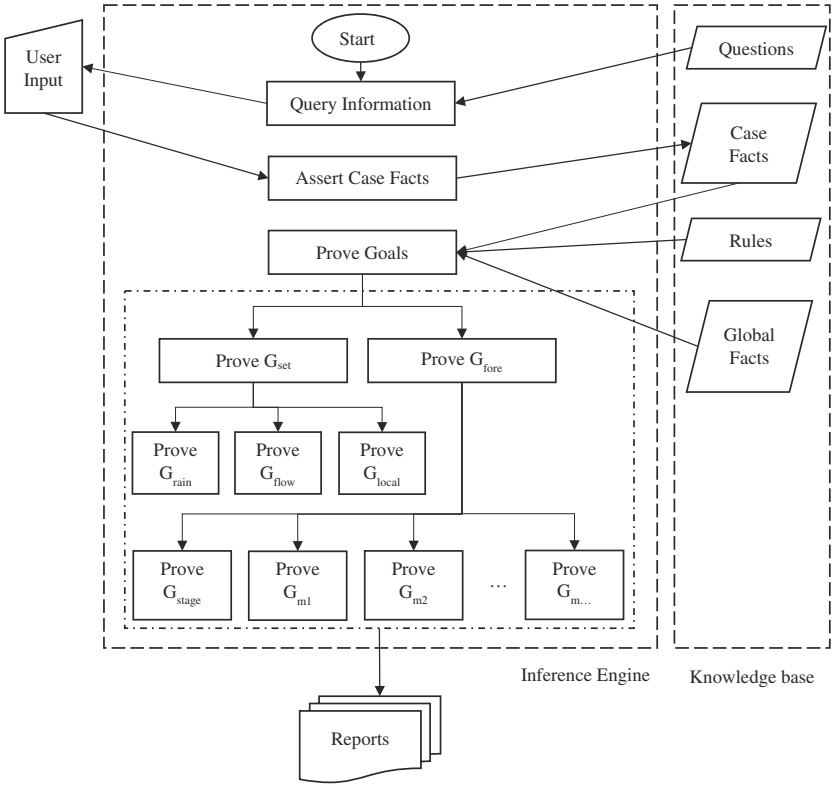

FIGURE 4 Architecture of the forward chaining expert system of Local Flood Forecasting and Warning Systems

away from the likely affected area, and adjust their travel plans for an escape. Because LFFW relies on "rules of thumb" instead of complicated numerical models, common users can eventually have a well-trained system installed and operated on their own mobile devices (like cellphones) to get rapid flood forecasting and warning. In addition, the explanatory capability of LFFW helps communicate the reasoning leading to the output guidance. Applying hypothetical data to the system allows LFFW to provide a training tool for flood responders and better acquaint them with the specifics of various warning stages, vital events (e.g., roads are flooded), preparation suggestions, and recommended response actions. Skilled users can fine-tune their own systems by resetting the variables and parameters or making endeavours according to their demand.

5 | VERIFICATION AND VALIDATION

Verification and validation (V&V) processes are critical components to guarantee the quality of developed expert systems. V&V processes include the analysis, evaluation, review, inspection, assessment, and testing of products. The technical aim of expert systems' V&V is determining whether the expert systems conform to the requirements and satisfy customers (IEEE, 2012; O'Keefe & O'Leary, 1993).

To assure an expert system is built correctly, developers typically verify their software by using a set of test cases either collected from real life situations or designed by domain experts to represent the possible problems in implementation (Adrion, Branstad, & Cherniavsky, 1982; O'Leary, Goul, Moffitt, & Radwan, 1990). With the assistance of debugging tools built in Python, we periodically verified our system throughout the development stage by conducting a complete set of predefined tests. Based on the results of the tests of V&V, we redesigned and reprogrammed the necessary heuristic knowledge and inferential logic.

To assure that we built the correct expert system, a paradigm for prototype validation combines face validation (the process by which the experts assess the prototype "at face value") with component testing and system validation through a case or Turing tests (O'Leary et al., 1990). According to this method, experts from a water resource management area reviewed the system's operation, output, and documentation. In addition, the experts tested our system using selected cases from their experience. The accuracy of the system is evaluated by comparing the forecasting provided by the system with the documented test cases.

6 | RESULTS AND DISCUSSION

In the following results and discussion section, the advantages of development and maintenance of expert systems over procedural codes are illuminated and analysed; then, the practicalities of forward and backward chaining inferential logic are studied and explained.

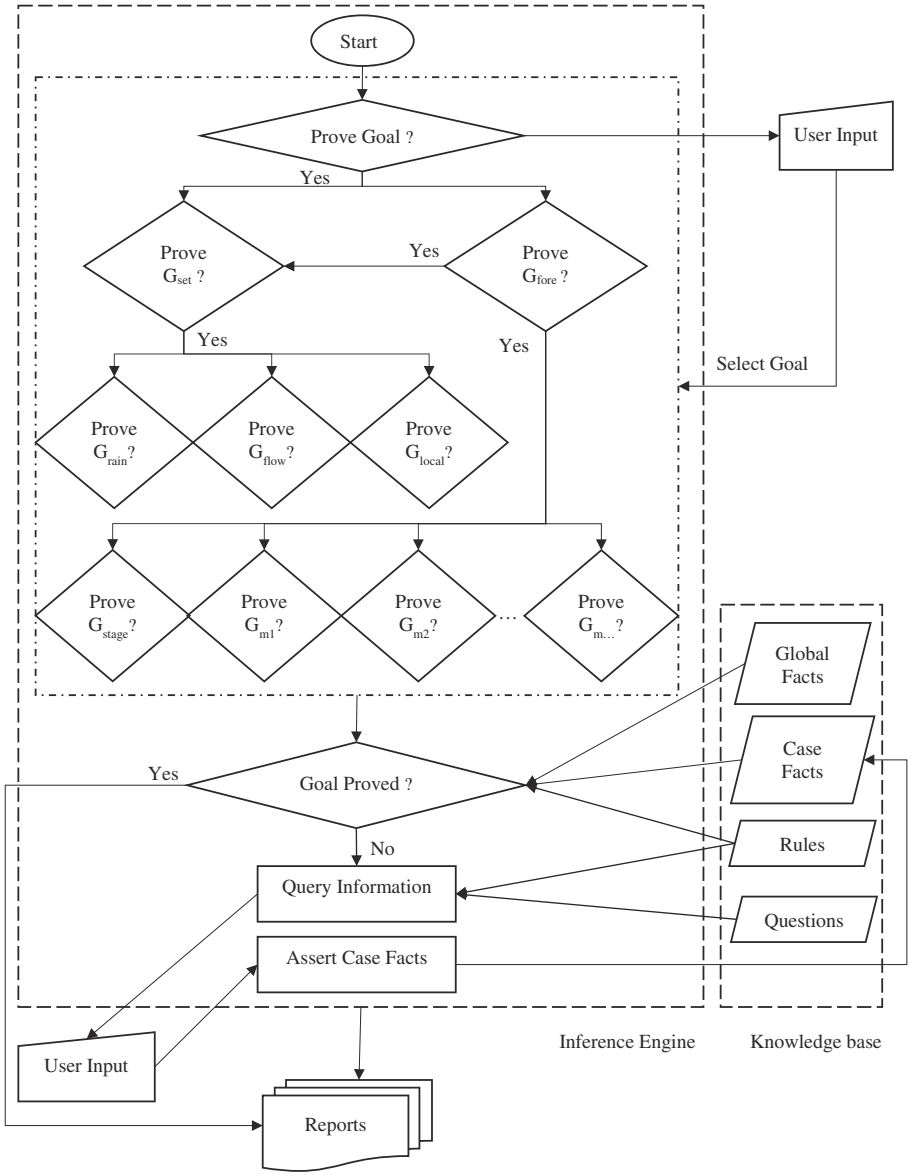

FIGURE 5 Architecture of the backward chaining expert system of Local Flood Forecasting and Warning Systems

```
Flood warning Wizard provides warnings and recommendations for local
community in the event of inundations.

Through the following interview, you will be asked a series of questions
regarding your river system and the flood event, from which you'll be
provided recommendations for flood rescue.

Do you wish to proceed? (y/n) y
```

FIGURE 6 Banner screen

```
------------------------------------------------------------------------
Reset water level triggers based on local conditions can provide more
precise warnings.

Do you wish to reset water level triggers? (y/n) y
```

FIGURE 7 System resetting screen

6.1 | Expert system versus procedural code

An expert system for local flood forecasting and warning has three primary advantages over procedural code: (a) an updatable knowledge base. The knowledge base of an expert system is explicit and separated from the inference engine. With no impact on other system components, errors and obsolete data can simply be corrected and replaced, whereas new knowledge corresponding to existing goals and/or new goals can be easily added.

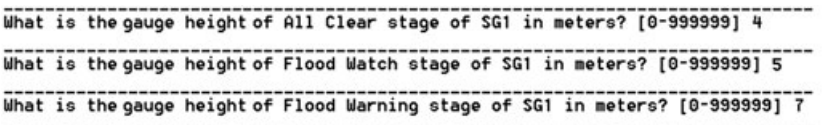

FIGURE 8 Water level triggers resetting screen. SG1 = stream gauge

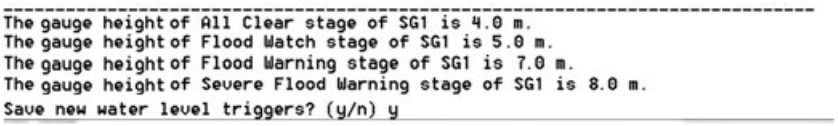

FIGURE 9 Water level triggers confirming screen. SG1 = stream gauge

TABLE 1 Report for Case 1 on flood forecasting and warning presented to users as a CSV file

CaseID	StreamgaugeID	Height (m)	Warning_Stage	Note
Case 1	SG1	5.34	Flood watch	This is the first stage of the warning. If your area is issued a flood watch, it means there is the possibility of some flooding. You are advised to keep a close eye on local radio or television reports, alert your neighbours, watch water levels, check on your pets, reconsider any travel plans, make sure you can put your flood plan into action, and ring the flood information telephone line for further information and advice.

Note. CSV = comma-separated values; SG1 = stream gauge.

TABLE 2 Report for Case 2 on flood forecasting and warning presented to users as a CSV file

CaseID	StreamgaugeID	Height (m)	Warning_Stage	Note
Case 2	SG1	7.52	Flood warning	If a flood warning is issued in your area, it means flooding is expected and will cause disruption. You are advised at this stage to move pets, vehicles, food, valuables, and other items to safety, be prepared to turn off the gas and electricity, be ready to evacuate your home, and put sandbags or flood boards in place to protect your home.

Note. CSV = comma-separated values; SG1 = stream gauge.

TABLE 3 Report for Case 3 on flood forecasting and warning presented to users as a CSV file

CaseID	StreamgaugeID	Height (m)	Warning_Stage	Note
Case 3	SG1	8.5	Severe flood warning	This is the warning issued when serious flooding is expected, and there is an imminent danger to life and property. If your warning is upgraded to this, you should be prepared for your gas, electricity, water, and telephone supplies being lost. You are advised to keep calm and reassure others and cooperate with the emergency services.

Note. CSV = comma-separated values; SG1 = stream gauge.

TABLE 4 Report for Case 4 on flood forecasting and warning presented to users as a CSV file

CaseID	StreamgaugeID	Height (m)	Warning_Stage	Note
Case 4	SG1	3.87	All clear	This is issued when the flood water levels are going down, and no flood watches or warnings are in force any longer. At this stage, you can check if it is safe to return home.

Note. CSV = comma-separated values; SG1 = stream gauge.

IDENTIFIER($ARGUMENT1, $ARGUMENT2, …)

Where,

"IDENTIFIER" represents a certain category of facts;

"$ARGUMENT1" and "$ARGUMENT2" represent different facts corresponding to the identifier, respectively.

FIGURE 10 General Python Knowledge Engine syntax of adding facts

(b) Flexible workflow: End users' goals can be proved in parallel by either a forward or backward chaining inference engine of an expert system. Without changing the original scripts of an expert system, all goals can be proved in various sequences. With minor modifications to the scripts, existing goals can be modified or deleted, and fresh goals related to different local flood forecasting and warning issues can enter the workflow. (c) Explanatory capability. Developers and end users can track the knowledge primitives' applied validly to prove a goal and understand the reason routine. These advantages enable rapid development, simple maintenance, and quick diagnoses.

6.1.1 | Benefits of an extensible knowledge base

The system architecture (shown in Figure 3) of a procedural code predetermines the difficulty of upgrading its database. The rubrics of applying particular knowledge to prove goals are merged into implicit operation structure. As a result, every modification of the database, such as inserting additional knowledge entities about a new stream gauge, requires adjusting the implicit operation code. In contrast, an enlarging knowledge base of an expert system has no requirement to change the scripts of inference engines. To demonstrate the details, we assume the following scenario: A new stream gauge, called Gauge 1, is installed or considered. Adding knowledge primitives such as gauge ID and warning stage thresholds about this gauge is then required. To solve this problem, sample codes of expert systems in PyKE syntax are given below.

In the manner of the general syntax for adding facts shown in Figure 10, one category of facts, called "warning_stage_triggers" with five essential information entities can be coded in the following way, as shown in Figure 11.

The last step is to replace arguments shown above with specific thresholds of stream flows in meters of Gauge 1 in the same sequence as follows (see Figure 12):

In this sample code, the text in the first pair of quotation marks is the first argument, "$gaugeID," which records the name or ID of the gauge; the number "1.5" in the second pair of quotation marks is the second argument. Then, "$clear" records the threshold of flood warning stage "All Clear" in meters, followed by "$watch," "$warning," and "$severe_warning." Unlike procedural code, the inference engine of an expert system can automatically search all facts under the same category (referred to as identifiers) in every loop.

6.1.2 | Benefits of a flexible workflow

In procedural code, the workflow is packaged in a fixed order and combined with questions and rules that correspondingly prove the goals. Therefore, the original scripts of the operation structure require amendments after any change in the workflow. For instance, inserting one novel goal into the workflow requires one to edit the original scripts of the operation structure, and all existing goals, rules, and questions are potentially impacted. If the new goal fails for some reason, all goals after it will stop proving. In contrast, all the goals of an expert system can be proved in parallel. After writing a short line of code to simply insert a new goal into the workflow, the inference engines automatically modify the compiled code. Even if the new goal fails, any other existing goals remain functional. To demonstrate the details, we assume that users want to know the warning stage; then, G_{stage} is required to be added into the workflows. To solve the problem, sample codes of the backward chaining expert system in PyKE syntax are given in Figures 13 and 14. In the same manner of the general syntax for adding a goal shown in Figure 13, G_{stage} is added by replacing those capitalized parameters with the specific information entities corresponding to G_{stage}, shown in Figure 14.

```
warning_stage_triggers($gaugeID, $clear, $watch, $warning, $severe_warning)

where,
 "$gaugeID" represents the ID of the stream gauge;
"$clear" represents the threshold of the warning stage "All Clear";
"$watch" represents the threshold of the warning stage "Flood Watch";
"$warning" represents the threshold of the warning stage "Flood Warning";
"$severe_warning" represents the threshold of the warning stage "Flood Warning".
```

FIGURE 11 General format of adding warning stage thresholds

```
warning_stage_triggers ("Gauge 1","1.5","2","2.5","3")
```

FIGURE 12 Specific facts of Gauge 1 in the warning stage category

```
with engine.prove_goal('rulebase.RULE_IDENTIFIER($ARGUMENT1, …)') as gen:
    for vars, plan in gen:
        …

where,
 "RULE_IDENTIFIER" represents a certain rule;
"$ARGUMENT1" represents a certain knowledge primitive corresponding to the rule.
```

FIGURE 13 General Python Knowledge Engine syntax of adding a goal

```
with engine.prove_goal('rulebase.warning_stage($gaugeID,$depth,$trigger)') as gen:
    for vars, plan in gen:
        …
```

FIGURE 14 Syntax of adding G_{stage}

Based on our study, we identify that there are essential information entities corresponding to the R_{stage}, such as gauge ID ($gaugeID), the actual water level in meters ($depth), and the threshold of flood warning stage ($trigger), so three arguments are included in the inner parentheses. In the same fashion, to insert an original goal into the workflow, we simply need two steps: (a) copy and paste one old goal; and (b) "plug and chug" the rules and facts to prove the original goal. Within the architectures of expert systems demonstrated in Figures 4 and 5, goals execute in various parallel sequences according to the availability of information at hand. The newly joined knowledge and goals will not affect the existing goals. In addition, backward chaining inferential logic also enables the workflow to adapt to the demands of users. However, the procedure code can only work in a fixed workflow, for example, G_{rain} ➔ G_{flow} ➔ G_{local} ➔ G_{stage} ➔ G_{m1} ➔ G_{m2} ➔ … ➔$G_{m...}$. If G_{rain} fails, then all goals after G_{rain} do not process. One common cause of the failure of G_{rain} is the lack of rainfall data, which often occurs for the following reasons: (a) rain gauge malfunction because of poor maintenance or other technical problems; (b) no local precipitation, for example, it rains heavily upstream but outside the local boundary or at least beyond the rain gauge; and (c) no present precipitation, such as in the case of a snowmelt flood. Therefore, the procedure code cannot cope with the scenarios stated above. In contrast to the incapability of procedural code in such situations, without making any changes to the original scripts, expert systems can skip G_{rain} automatically and prove the rest of the goals that are unrelated to G_{rain}. Although G_{rain} fails, the known rising rate, depth, and/or velocity of stream flow can still fulfil the other goals such as the flood warning stage and other warning messages related to stream flows. Simply speaking, our expert systems can prove the goals in any sequence of workflows.

Therefore, we can develop our system by each goal and later pool the tested goals together. In this study, we only address partial issues listed in the 2011 WMO manual. However, as research continues, new knowledge, including facts and rules corresponding to other triggers and novel goals, will definitely be needed in LFFWS. When more issues of local flood forecasting and warning have been considered as fresh goals, one or two of these goals may fail due to the absence of information at hand or may be skipped because of lack of user interest. The LFFWS should enable the users to select some goals to be skipped or prove other goals first. At the same time, the system should automatically skip failed goals and move ahead to other goals. From this perspective, expert systems have striking benefits over procedural codes. The benefits from the extensible knowledge base and the flexible workflow of an expert system will be more and more attractive as more issues are considered.

6.1.3 | Benefits of explanatory capability

On the one hand, because facts are isolated from rules in conventional procedural codes, conventional procedural systems habitually lack the capability to explain why a fact is deduced or inferred in a particular way. In other words, procedural codes cannot tell users which facts and rules lead to creating the reasonable conclusions.

On the other hand, facts and rules are stored together in the knowledge base of expert systems. Developers can detect all reasoning routines for logic or syntax errors by tracing the list of valid facts and rules applied to solve a certain problem. As a result, problematic scripts can be locked down quickly. This explanatory capability is especially helpful when the developers are coding complex courses, such as the procedure of local flood forecasting and warning. The explanatory capability simplifies and accelerates the development of the computer systems. In addition, the explanatory capability can train those local flood managers with routines for reasoning in particular scenarios. For example, our expert systems can create a new fact in the following way, shown in Figure 15:

Although developers or users read the analysis and conclusion, they can understand the cause-effective routines from the list of knowledge primitives validly applied to the goal. For example, assume that knowledge Primitive 1 is accidentally coded incorrectly as "The threshold of Severe Flood Warning is 3000 meters." Then, the developers will not obtain the expected Severe Flood Warning. Instead of searching for all facts, the developers search the knowledge primitives on the reasoning list only. Obviously, diagnosing and correction processes are expedited in this way. Therefore, the larger the knowledge base and the more complicated the reasoning routines are, the more appealing the explanatory capability of an expert system is. Thus, developing an expert system is more proficient than procedural code for LFFWS.

6.2 | Backward chaining versus forward chaining

To illustrate how the backward chaining mechanism is applied to and enhances our expert system, simplified forward and backward chaining logic is shown in Figures 16 and 17, respectively. Different dash types and arrow types indicate diverse information flows. Generally, thresholds of rainfall,

User input: Water depth is 3.2 meters.
Knowledge primitive 1: The threshold of Severe Flood Warning is 3 meters.
Knowledge primitive 2: Severe Flood Warning: "This is the warning issued when serious flooding is expected…"
➔ Infer new knowledge primitive: Issue Severe Flood Warning: "This is the warning issued when serious…"

FIGURE 15 Example of fact assertion

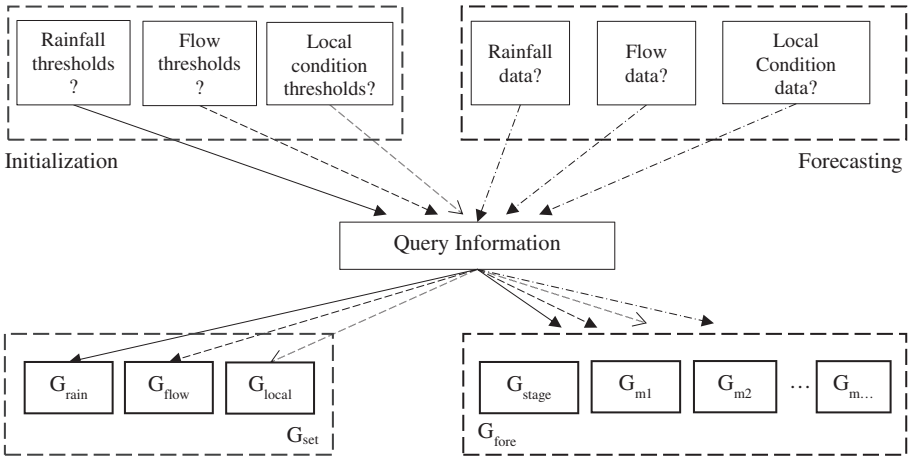

FIGURE 16 Simplified forward chaining logic to prove goals

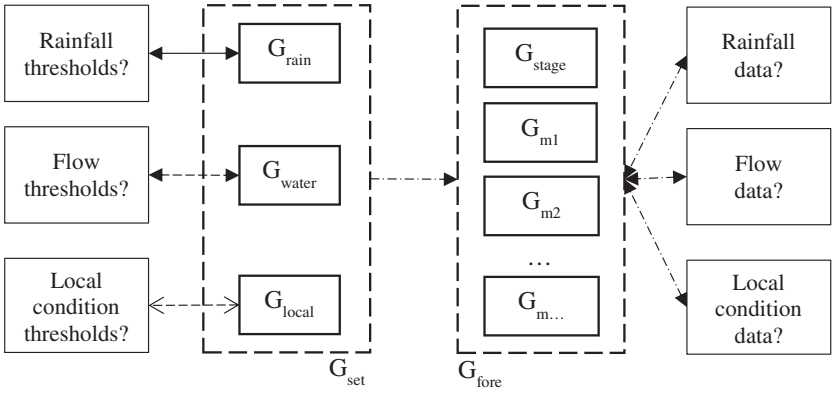

FIGURE 17 Simplified backward chaining logic to prove goals

flow, and local conditions are only required during system initialization to prove G_{rain}, G_{flow}, and G_{local}, respectively. Once the system is well-trained, these parameters and variables are saved for repeated use until the system resets. Other current or proposed data on rainfall, stream flows, and local conditions are only needed during the forecasting phase. Take the proof of G_{flow} as an example. In this case, only the thresholds of stream flow, such as depth, velocity, and rising rate, are essential, so the backward chaining inference engine only collects the facts on these triggers. On the other hand, the forward chaining inference engine also blindly collects other information such as thresholds of rainfall, thresholds of local conditions, and current or proposed data on rainfall, stream flows, and local conditions. With the forward chaining expert system, users must provide the complete query for all incorporated goals at the beginning in order for the expert system to complete processing, even when some of this information is not available or is not of interest to the users. To allow for response questions only as inferential logic and thus simplify the user experience, the knowledge base in this study is entirely written for backward chaining inferencing.

The advantages of backward chaining do not present significant benefits in the determinant phase. The combinations of current data of rainfall, stream flows, and local conditions are required multiple times to determine distinct forecasting and warning messages. In addition, common users do not usually have any clear desire for specific goals. To enter all known information and expect every potential forecasting and warning recommendation is less challenging than selecting "useful" goals. Therefore, setting up an ultimate goal (G_{fore}) to discover complete information on the current or proposed situation and prove the entire goal set of the determinant phase, which is to some extent equivalent to forward chaining logic, is desired to shorten the practice of common users. In the meantime, the convenience of goal selection is there for skilled users and system developers.

7 | CONCLUSIONS

To break through the bottleneck of knowledge acquisition in local flood forecasting and warning, we identified the knowledge necessary to LFFWS and assimilated these dynamic and static knowledge primitives into the knowledge base. The case study illustrates that the collected knowledge works successfully to initialize the system and provide flood forecasting and warning messages on current or proposed data of rainfall, stream flows, and local conditions.

In addition, this study shows that, to provide local flood forecasting and warning, developing an expert system is more effective than procedural code. The advantages of rapid development and easy maintenance stem from the system architecture of expert systems. The explicit knowledge base and packaged expert system shell ensure that (a) new facts, rules, questions, and goals can be easily added to the extensible knowledge base by domain experts, such as environmental engineers or even skilled users to adapt to more scenarios; (b) all goals can be skipped or proved in various sequences automatically (forward chaining) or according to users' demands (backward chaining); (c) partially developed expert systems can be functional; (d) logic or syntax errors and outdated data can be rapidly identified and corrected; and (e) the users can understand and learn the reasoning simultaneously when they obtain the reports.

Furthermore, the backward chaining method is shown to work more effectively than forward chaining to satisfy local flood managers' evolving demands (referred to as goals) and growing new information on LFFWS. Backward chaining enables the inference engine of expert systems to work with incomplete information at the beginning and to keep running as more and more data become available. With a backward chaining inference engine, our expert system can quickly figure out and optimally collect the necessary data from previous analysis results or user interviews based on the evolving goals and efficiently develop reports and recommendations with the growing information at hand. In the meantime, without sacrificing convenience for skilled users, to ease the difficulty of goal selection and shorten the practice for common users, an ultimate goal (G_{fore}) is set up to acquire a complete picture of the current or proposed situation and issue all likely forecasting and warning messages.

Although the contemporary LFFWS currently work with limited goals for local flood forecasting and warning, additional goals and their corresponding knowledge related to other key issues can be easily updated in LFFWS. This prototype of a backward chaining expert system of local flood forecasting and warning, giving reasonably accurate predictions and recommendations, can decrease the engagement of computational resources by minimizing the boundary of the interested area and decreasing the usage of numerical models. This makes flood forecasting more cost-effective and therefore feasible for small communities, especially for those with tight flood management budgets. This research represents an advance in the applicability of expert systems to solve flood prediction and management problems. Further, the framework of our expert systems can easily be duplicated to create other expert systems. Domain experts in other areas can make use of our framework to record their valuable expertise and undocumented “rules of thumb” in computer-readable language and create more expert systems to perform repeated work efficiently.

CONFLICT OF INTEREST

None.

ORCID

Xiaoyin Zhang http://orcid.org/0000-0001-8345-4785

REFERENCES

Adrion, W. R., Branstad, M. A., & Cherniavsky, J. C. (1982). Validation, verification, and testing of computer software. *ACM Computing Surveys*, *14*(2), 159–192.

Anderson, M. L., Chen, Z.-Q., Kavvas, M. L., & Feldmand, A. (2002). Coupling HEC-HMS with atmospheric models for prediction of watershed runoff. *Journal of Hydrologic Engineering*, *7*(4), 312–318.

Badrzadeh, H., Sarukkalige, R., & Jayawardena, A. W. (2015). Hourly runoff forecasting for flood risk management: Application of various computational intelligence models. *Journal of Hydrology*, *529*, 1633–1643.

Chau, K. W., & Phil, M. (2004). Knowledge-based system on water resource management in coastal waters. *Water and Environmental Journal*, *18*(1), 25–28.

Chen, M.-K. S., Chau, C.-F. C., & Kabat, W. C. (1985). Decision support systems: A rule-based approach. In *ACM '85 proceedings of the 1985 ACM annual conference on the range of computing : Mid-80's perspective: mid-80's perspective* (pp. 511–515). New York: Association for Computing Machinery.

Cloke, H., & Pappenberger, F. (2009). Ensemble flood forecasting: A review. *Journal of Hydrology*, *375*, 613–626.

Comas, J., Llorens, E., Marit, E., Puig, M., Riera, J., Sabater, F., & Poch, M. (2003). Knowledge acquisition in the STREAMES project: The key process in the environmental decision support system development. *AI Communications*, *16*, 253–265.

da SilvaAvanzi, D., Foggiatto, A., Santos, V. A., Deschamps, F., & Loures, E. F. (2017). A framework for interoperability assessment in crisis management. *Journal of Industrial Information Integration*, *5*, 26–38. https://doi.org/10.1016/j.jii.2017.02.004.

DHS. (2013). National response framework.

Emerton, R. E., Stephens, E. M., Pappenberger, F., Pagano, T. C., Weerts, A. H., Wood, A. W., … Cloke, H. L. (2016). Continental and global scale flood forecasting systems. *Wiley Interdisciplinary Reviews: Water*, *3*(3), 391–481.

Fang, S., Xu, L., Pei, H., Liu, Y., Liu, Z., & Zhu, Y. (2014a). An integrated approach to snowmelt flood forecasting in water resource management. *IEEE Transactions on Industrial Informatics*, *10*, 548–558.

Fang, S., Xu, L., Zhu, Y., Liu, Y., Liu, Z., Pei, H., … Zhang, H. (2015). An integrated information system for snowmelt flood early-warning based on internet of things. *Information Systems Frontiers*, *17*(2), 321–335. https://doi.org/10.1007/s10796-013-9466-1.

Fang, S., Xu, L. D., Zhu, Y., Ahati, J., Pei, H., Yan, J., & Liu, Z. (2014b). An integrated system for regional environmental monitoring and management based on internet of things. *IEEE Transactions on Industrial Informatics*, *10*(2), 1596–1605.

Feigenbaum, E. (1977). *The art of artificial intelligence: Themes and case studies of knowledge engineering*. Stanford: School of Humanities and Sciences, Stanford University, Computer Science Department.

Garcia-Pintado, J., Mason, D. C., Dance, S. L., Cloke, H. L., Neal, J. C., Freer, J., & Bates, P. D. (2015). Satellite-supported flood forecasting in river networks: A real case study. *Journal of Hydrology*, *523*, 706–724.

Ghalkhani, H., Golian, S., Saghafian, B., Farokhnia, A., & Shamseldin, A. (2012). Application of surrogate artificial intelligent models for real-time flood routing. *Water and Environment Journal*, *27*(4), 535–548.

Hayer-Roth, F., Waterman, D., & Lenat, D. (1983). *Building expert systems*. Boston: Addison-Wesley.

IEEE. (2012). IEEE standard for system and software verification and validation. Retrieved from IEEE Standards Association: https://standards.ieee.org/findstds/standard/1012-2012.html (accessed 11 September 2017)

Jackson, P. (1998). Introduction to expert systems. In P. Jackson (Ed.), *Introduction to expert systems* (3rd ed.) (pp. 2). New York: Addison Wesley.

Jasper, K., Gurtz, J., & Lang, H. (2002). Advanced flood forecasting in Alpine watersheds by coupling meteorological observations and forecasts with a distributed hydrological model. *Journal of Hydrology*, *267*, 40–52.

Kaewboonma, N., Tuamsuk, K., & Kanarkard, W. (2013). Knowledge acquisition for the design of flood management information system: Chi river basin, Thailand. In *Social and behavioral sciences* (Vol. 73) (pp. 109–114). Budapest: Elsevier Ltd.

Kivy. (2015). Retrieved from http://kivy.org/#home (accessed 14 September 2017)

Leon, C., Martin, S., Elena, J. M., & Luque, J. (2000). Explore: Hybrid expert system for water networks management. *Journal of Water Resources Planning and Management*, *126*(2), 65–74.

Li, L., Wang, B., & Wang, A. (2014). An emergency resource allocation model for maritime chemical spill accidents. *Journal of Management Analytics*, *1*(2), 146–155. https://doi.org/10.1080/23270012.2014.943137.

Lueckemeyer, O., & Lueckemeyer, O. (2017, August 25). Residents of Onion Creek, Southeast Travis County prepare for potential flooding. Retrieved from Community Impact Newspaper: https://communityimpact.com/austin/southwest-austin/news/2017/08/25/residents-onion-creek-southeast-travis-county-prepare-potential-flooding/ (accessed 25 August 2017)

Luo, J., Xu, L., Jamont, J.-P., Zeng, L., & Shi, Z. (2007). Flood decision support system on agent grid: Method and implementation. *Enterprise Information Systems*, *1*(1), 49–68. https://doi.org/10.1080/17517570601092184.

Mabrouk, M. E., Ezziyyani, M., Sadouq, Z. A., & Essaaidi, M. (2015). New expert system for short, medium and long term flood forecasting and warning. *Journal of Theoretical and Applied Information Technology*, *286-302*.

Mahabir, C., Hicks, F., & Fayek, A. R. (2007). Transferability of a neuro-fuzzy River ice jam flood forecasting model. *Cold Regions Science and Technology*, *48*(3), 188–201.

Mounce, S., Boxall, J., & Machell, J. (2010). Development and verification of an online artificial intelligence system for detection of bursts and other abnormal flows. *Journal of Water Resources Planning and Management*, *136*(3), 309–318.

Najafi, M. R., & Moradkhani, H. (2015). Ensemble combination of seasonal streamflow forecasts. *Journal of Hydrologic Engineering*, 2438–2453.

O'Keefe, R. M., & O'Leary, D. E. (1993). Expert system verification and validation: A survey and tutorial. *Artificial Intelligence Review*, *7*, 3–42.

O'Leary, T. J., Goul, M., Moffitt, K. E., & Radwan, A. E. (1990). Validating expert systems. *IEEE Intelligent Systems*, *5*(3), 51–58.

Ooshaksaraie, L., & Basri, N. E. (2011). An expert system applied in construction water quality monitoring. *American Journal of Environmental Sciences*, *7*(11), 75–81.

Pinto, D., Castro, L., Cruzat, M., Barros, S., Gironas, J., Oberli, C., … Cipriano, A. (2015). Decision support system for a pilot flash flood early warning system in Central Chile. *International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, *9*(3), 990–997.

PyKE. (2015). Retrieved from welcome to Pyke: http://pyke.sourceforge.net/ (accessed 11 September 2017)

Python. (2015). Retrieved from Python: https://www.python.org/ (accessed 11 September 2017)

Robindro, K., & Sarma, S. K. (2013). JESS based expert system architecture for diagnosis of rice plant diseases: Design and prototype development. *2013 4th International Conference on Intelligent Systems, Modelling and Simulation*, (pp. 674–676).

Santos, M. Y., & Martinho, B. (2017). Modelling and implementing big data warehouses for decision support. *Journal of Management Analytics*, *4*(2), 111–129.

Spyridakos, T., Pierakos, G., Metaxas, V., & Logotheti, S. (2005). Supporting the management of measurement network with an expert system: The NeMO system. *Operational Research An International Journal*, *5*(2), 273–288.

USGS. (2017, September). USGS 08158700 Onion Ck nr Driftwood, TX. Retrieved from USGS National Water Information System: Web Interface: https://nwis.waterdata.usgs.gov/usa/nwis/uv/?cb_00045=on&cb_00060=on&cb_00065=on&format=gif_mult_parms&site_no=08158700&period=&begin_date=2013-03-16&end_date=2017-09-23 (accessed 14 September 2017)

Wang, L., Xu, L. D., Bi, Z., & Xu, Y. (2014, February). Data cleaning for RFID and WSN integration. *IEEE Transactions on Industrial Informatics*, *10*(1), 408–418. https://doi.org/10.1109/TII.2013.2250510.

WMO (2011). *Manual on flood forecasting and warning*. Geneva: WMO Retrieved from http://www.wmo.int/pages/prog/hwrp/publications/flood_forecasting_warning/WMO%201072_en.pdf.

Wong, B. K., & Monaco, J. A. (1995). Expert system applications in business: A review and analysis of the literature (1977–1993). *Information Management*, *29*(3), 141–152.

Xu, L., Liang, N., & Gao, Q. (2008). An integrated approach for agricultural ecosystem management. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *38*(4), 590–599. https://doi.org/10.1109/TSMCC.2007.913894.

Xiaoyin Zhang is currently a PhD candidate in the Department of Civil, Construction and Environmental Engineering at the University of Alabama. She received her BS degree in Environmental Engineering from Southwest Jiaotong University and obtained MS degree in Donghua University, China. Her research interests include expert systems and decision support systems in water resource management.

Gary P. Moynihan is a Professor and Lead Engineer for the Architectural and Construction Engineering Programs in the Department of Civil, Construction and Environmental Engineering at the University of Alabama. He received his BS and MBA Degrees from the Rensselaer Polytechnic Institute and a PhD from the University of Central Florida. His primary area of specialization is the development of expert systems and decision support systems. Prior to joining the Faculty of the University of Alabama, he held positions in the aerospace, computer, and chemical processing industries.

Andrew N.S. Ernest was a Professor in the Department of Civil, Construction and Environmental Engineering at the University of Alabama. Currently, he is a Professor at the University of Texas—Rio Grande Valleya. He earned a BS and MS in Civil Engineering from the University of Southwestern Louisiana and a PhD in Civil Engineering from Texas A&M University. His specialties include water resource management; regional environmental sustainability; environmental informatics; environmental modelling; decision support systems; and environmental management systems.

Joseph L. Gutenson is currently a Research Civil Engineer at the U.S. Army Engineer Research and Development Center (ERDC) in Vicksburg, Mississippi. He has a Bachelor of Science in Geography from Western Kentucky University and a Masters and PhD in Civil Engineering from The University of Alabama. Joseph has worked on a variety of water-related projects, most notably in the development of TMDLs, a rule-based decision support system for water resource management, and the National Flood Interoperability Experiment (NFIE).

How to cite this article: Zhang X, Moynihan GP, Ernest ANS, Gutenson JL. Evaluation of the benefits of using a backward chaining decision support expert system for local flood forecasting and warning. *Expert Systems*. 2018;e12261. https://doi.org/10.1111/exsy.12261