Development of an Expert System for Preliminary Selection of Pile Foundation

C. H. JUANG AND M. L. ULSHAFER

The development of an expert system for selection of pile type for design of a pile foundation is documented. The expert system, Pile Selection version 1.0 (PS1), incorporates many unique features. Among them are approximate reasoning with fuzzy logic, the blackboard architecture, and the emulated parallel processing of fuzzy production rules. An example using PS1 for pile selection is presented, and possible enhancement of the system is discussed.

Preliminary selection of pile type is an important step in the design and construction of pile foundation. Proper initial selection can shorten the design process and reduce the project cost. For experienced engineers, this initial task of choosing one or more piles for further design consideration may not be a problem. However, it may be quite challenging for persons with less experience because there are numerous uncertainties involved in several areas, including loading requirement, subsurface soil conditions, pile material properties, methods of construction, nuisance driving effects, and space and time constraints. An expert system that can provide a consistent and reliable selection of pile foundation, taking into account these factors, would certainly be very useful. This is the motivation for developing the system Pile Selection version 1 (PS1).

The PS1 system was written in FLOPS, a Fuzzy LOgic Production System created by Siler and Tucker (1). This paper will briefly describe FLOPS, followed by detailed discussion of the development of PS1.

FLOPS FEATURES

FLOPS is an expert system shell written in C language for use in the Microsoft Disk Operating System (MS-DOS) or compatible DOS environment on microcomputers. FLOPS has several unique features that provide a great deal of power and flexibility. A brief summary of the FLOPS features follows.

Approximate Reasoning with Fuzzy Logic

FLOPS uses fuzzy logic invented by Zadeh (2,3). Fuzzy sets and logic allow for a better model of an expert's reasoning process. The subject of civil engineering applications of fuzzy sets is beyond the scope of this paper and has been documented elsewhere (4–9).

Civil Engineering Department, Clemson University, Clemson, S.C. 29634.

Deductive and Inductive Reasoning

FLOPS is a production system and as such its basic element is a "rule." The deductive logic implemented in FLOPS is no different than most expert systems. It fires the production rules sequentially. If the data permits more than one rule to be "fireable," deductive systems select one rule for firing; other fireable rules are stacked for backtracking later. FLOPS, however, also implements inductive reasoning that considers many possible outcomes at once. FLOPS' parallel rule firing scheme for implementing the inductive reasoning is rather unique. All fireable rules are fired concurrently, and thus no rule remains to be stacked for backtracking. FLOPS adopts a weakly monotonic fuzzy logic for its truth maintenance to resolve the memory conflict problem. When applicable, the inductive mode of FLOPS is much faster than the deductive mode to reach a conclusion.

Blackboard Architecture

FLOPS employs a relational structure for data stored on a blackboard, a disk on microcomputers in the context of this paper. The ability of one FLOPS program to call another and to exchange data through the blackboard could overcome the memory limitations of small microcomputers.

Expert knowledge is divided into two classes in FLOPS. One is factual knowledge, which belongs in a data base. The other is expert skills, which belong in rules. One of these expert skills knows how to use the expert factual knowledge. The programmer-written rules can generate the production rules based on the factual knowledge during the program execution.

Two methods of communicating with external programs are available in FLOPS using a *call* command. One type of call transmits a command string to the called program in the DOS environment. The other is a call by reference to a C program and thus must follow calling convention used in the C language. Details of the above features as well as others can be found in FLOPS manual (1).

FACTORS AFFECTING SELECTION OF PILE TYPE

To arrive at the optimum pile foundation solution, the engineer must have thorough information and understanding of (1) foundation loads, (2) subsurface soil and rock conditions and properties, and (3) current practices in pile design and construction. Based on site conditions and design requirements, the designer may select two or more alternatives for further consideration. Analysis will be made to check out bearing capacity and settlement requirements. Comparison of the costs of acceptable alternatives then follows. This process might repeat one or more times to reach the optimum design.

Undoubtedly, the initial pile selection is an important step in the process. However, it is often not given proper coverage in the formal college engineering education. Years of apprenticeship seem to be the only way to acquire the needed experience for mastering this task. To bridge this gap, PS1 was created to assist the designers with various backgrounds on this very task of preliminary pile selection.

The selection of appropriate pile type for any given set of circumstances depends upon many variables. In particular, the type of subsoil, the groundwater condition, the topography of the site, the design loads, the construction concerns, and the location of the site in relation to nuisance effects, availability of pile material, and transportation costs are all of importance.

There are basically two routes to acquire expert knowledge. One is to work directly with an expert or a group of experts by conducting interviews necessary to extract expert knowledge. The other is to conduct an extensive literature review to extract rules of thumb and knowledge. The latter route was taken in this study. Extensive review of the literature on pile foundation was made during the course of this study. Major references on which PS1 was based are listed below (*10–28*). It is expected that the strength of the knowledge base will continue to grow as PS1 continues to evolve. Thus, the structure of PS1 was arranged in such a way that new or better expert opinions can be easily incorporated without any significant change in PS1.

DEVELOPMENT OF PS1

General Program Structure of PS1

The expert system PS1 was written in FLOPS, and as such it is often referred to as a FLOPS program in this paper. A FLOPS program may be grouped into three sections:

1. the declaration section, similar to that of C or PASCAL language;
2. the rules section, where the actual rules appear; and
3. the input section, in which actual values are assigned to the attributes described in the declaration section.

The basic structure of PS1 closely follows that of RMC, an expert system developed for rock mass classifications by Juang and Lee (*29*). Figure 1 shows its general architecture. PS1 first reads external "factual knowledge" files, which are referred to herein as knowledge data bases or simply data bases. The knowledge data bases required by PS1 include DL.kb (design load knowledge base), PL.kb (pile length knowledge base), SC.kb (soil condition knowledge base), NE.kb (noise effect knowledge base), VE.kb (vibration effects knowledge base), AOM.kb (availability of material knowledge base), and LE.kb (local experience knowledge base). Using this knowledge and some "meta-rules," PS1 can generate all needed production rules. Depending upon the sizes of these knowledge data bases, a total of several hundred rules may be generated with a few user-written rules.

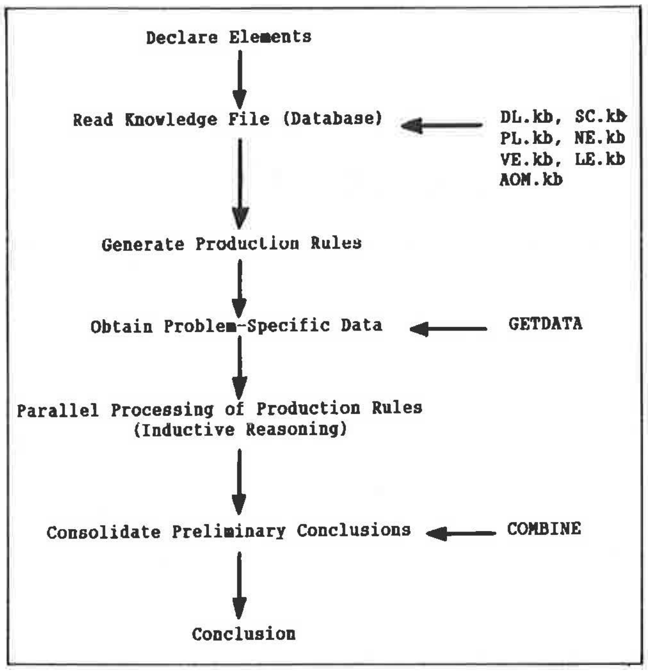
Declare Elements
Read Knowledge File (Database)
DL.kb, SC.kb
PL.kb, NE.kb
VE.kb, LE.kb
AOM.kb
Generate Production Rules
Obtain Problem-Specific Data
GETDATA
Parallel Processing of Production Rules
(Inductive Reasoning)
Consolidate Preliminary Conclusions
COMBINE
Conclusion

FIGURE 1 Overall structure of PS1.

After the rules are generated, the program will begin to ask for user input data by calling an external program GETDATA. The program GETDATA was written in C language and compiled with Microsoft C 5.1 compiler (*30*). It serves as a user interface that allows for the problem-specific data to enter PS1. Execution of GETDATA will create a data file called user.dat to store these user inputs. The user.dat file is then transferred to PS1 and matched with generated rules for determining rule fireability. (When input matches the premise of a generated rule, it makes that rule fireable.) Execution of the production rules then begins, which produces several preliminary conclusions. These conclusions are stored in a file named pile_type.dat. This new file serves as an input to COMBINE, another external program called by PS1. COMBINE, also written in the C language, consolidates the preliminary conclusions reached earlier. Result of the final conclusion reached is then reported to PS1.

It is noted that the expert system PS1 was written in parallel FLOPS. Thus, it does not involve backtracking of the rules. Instead, at any stage of rule firing, all fireable rules are fired at the same time. The problem of possible memory conflicts was resolved using weakly monotonic logic, a type of fuzzy logic in which the value of a datum may be replaced by a new value, if the confidence in the new value is greater than or equal to the old confidence (*1*). However, the preliminary conclusions reached at different stages were treated as evidences, each based on a particular knowledge source. They were not combined with the weakly monotonic logic. Instead, an external program COMBINE was used to consolidate these evidences.

To use PS1, the user needs to have an idea of the design loads to be supported, subsurface soil condition, pile length requirement, availability of material, local contractor experience, and noise or vibration constraints. The interactive, menu-driven program GETDATA will direct the user through the creation of a user-supplied file, user.dat, which is required by PS1. On-line explanation features are provided in GETDATA to assure its user-friendly style. Also, the program GETDATA can be run as part of PS1, or it can be run separately before execution of PS1. This feature allows for maximum flexibility on the part of the user when consulting PS1.

Detailed Comments on PS1

As mentioned earlier, there are three major sections in a FLOPS program. Detailed comments on each section of PS1 follow.

Declaration Section

Elements declaration is the only task in the declaration section. Figure 2 shows an example of element declaration. The command *literalize* declares a memory element, xdata, with 28 attributes of the types *atm* and *flt*. In FLOPS, the data type *atm* is for character string, and the data types *flt* and *int* are for floating point and integer, respectively. The syntax for declaration is very similar to that of a structure in C or PASCAL language. An attribute is analogous to a variable in C or PASCAL. Notice that a semicolon is only needed at the end of the entire *literalize* command, and that separation of the *literalize* command into several lines is a programming style for ease of reading and maintaining of the code.

Element xdata is needed to store the user-supplied data. Other elements are declared to store factual knowledge data. Figure 3 shows another example of the element declaration. Element pile_type has seven attributes, criterion1 through criterion7, that are declared to be of data type *fzset*, which stands for fuzzy set. The fuzzy set data type in FLOPS is unique. In common fuzzy set notation, using criterion1 as an example, it may be expressed as

```
:comment - user supplied (problem-specific) data

literalize xdata
  design_load          flt    avail_PC     atm    exp_PC     atm
  soil_type            atm    avail_PSC    atm    exp_PSC    atm
  strength             atm    avail_CIPM   atm    exp_CIPM   atm
  negative_friction    atm    avail_CIP    atm    exp_CIP    atm
  boulder              atm    avail_STL    atm    exp_STL    atm
  pile_length          flt    avail_TM     atm    exp_TM     atm
  noise                atm    avail_CPW    atm    exp_CPW    atm
  vibration            atm    avail_CPS    atm    exp_CPS    atm
                              avail_PIC    atm    exp_PIC    atm
                              avail_BP     atm    exp_BP     atm
```

FIGURE 2 Declaration of element xdata for PS1.

```
:comment - pile type selection according to each criterion

literalize pile_type
    criterion1    fzset  (PC PSC CIPM CIP STL TM CPW CPS PIC BP)
    criterion2    fzset  (PC PSC CIPM CIP STL TM CPW CPS PIC BP)
    criterion3    fzset  (PC PSC CIPM CIP STL TM CPW CPS PIC BP)
    criterion4    fzset  (PC PSC CIPM CIP STL TM CPW CPS PIC BP)
    criterion5    fzset  (PC PSC CIPM CIP STL TM CPW CPS PIC BP)
    criterion6    fzset  (PC PSC CIPM CIP STL TM CPW CPS PIC BP)
    criterion7    fzset  (PC PSC CIPM CIP STL TM CPW CPS PIC BP)
```

FIGURE 3 Declaration of element pile_type for PS1.

$$\begin{aligned}\text{criterion1} = \{&\text{m1/PC, m2/PSC, m3/CIPM,}\\ &\text{m4/CIP, m5/STL, m6/TM,}\\ &\text{m7/CPW, m8/CPS, m9/PIC, m10/BP}\}\end{aligned}$$

where m1 through m10 are the membership grades for the corresponding members PC through BP, respectively, which in turn are abbreviations for the following:

PC =	precast concrete;
PSC =	prestressed concrete;
CIPM =	cast-in-place concrete with mandrel;
CIP =	cast-in-place concrete without mandrel;
STL =	steel pile (H, I section);
TM =	timber;
CPW =	composite wood-concrete;
CPS =	composite steel-concrete;
PIC =	pressure-injected concrete; and
BP =	bored.

In FLOPS, these membership grades appear in the form of a confidence level. The confidence level is a unique data type, which is used to store the confidence toward a member of a fuzzy set. The attribute criterion1 is created to store the preliminary conclusion of pile selection based on design loads. The attributes criterion2 through criterion7 are declared in the same way, each based on a different pile selection factor incorporated in PS1. Although these seven attributes look alike, use of different attributes is necessary to preserve multiple preliminary conclusions reached at different stages in the inductive reasoning process in PS1. Otherwise, FLOPS' weakly monotonic logic could eliminate the desired membership values of these fuzzy set members before they can be combined.

Rule Section

For the PS1 system it was determined that a parallel (inductive) FLOPS program is more effective than a sequential (deductive) one. It was also decided to set up a block firing control, starting from block 0, to ensure the sequential firing of each block of rules. Within each block, however, the parallel processing ensures all rules that are fireable are fired at once.

The PS1 system starts with reading of expert knowledge files in block 0. With this knowledge, part or all of the block

0 rules become fireable and are fired at once. The actions of firing these rules generate the rules of blocks 2 through 8. It is noted that, without proper initiation of data, no rules can actually be fired.

As an example to explain how rules are generated, focus on rule r0, shown in Figure 4. The expert factual knowledge was stored in a file named DL.kb. As soon as it is transferred to the system (using the command *open*, explained below), the left hand side (LHS) of rule r0 will be satisfied. In other words, rule r0 becomes fireable; and when it is fired, the right hand side (RHS) of rule r0 will be executed. It is noted that LHS is generally referred to as premise part of a rule, while RHS is the action part of the rule.

For convenience of the further discussion, an example of the content of knowledge file DL.kb is shown in Figure 5. It basically consists of a set of *make* commands. The *make* command initiates a memory element and assigns values to its attributes. For example, when the file DL.kb is open, the first *make* command assigns the following data:

^lower1 = 0;
^upper1 = 20;
^fsmember = CIPM;
^confidence = 600.

Notice the ^ is the symbol used in FLOPS for the value of the attribute. Once these values are transferred to the system, the variables in the LHS of the rule r0 take on the following values:

⟨LB⟩ = 0;
⟨UB⟩ = 20;
⟨FSM⟩ = CIPM; and
⟨CONF⟩ = 600.

When rule r0 is fired, the action part (i.e., the RHS) of the rule yields a new rule, as shown in Figure 6. Whether the new rule is fireable depends on the actual attribute values in the elements xdata and pile_type. Notice how a membership grade of a member in a fuzzy set is represented. The term, ^criterion.CIPM, represents the confidence level (membership grade) toward the member CIPM of the fuzzy-set attribute criterion1.

Separation of factual knowledge data from the main part of PS1 is convenient for maintaining the system. When expert opinions change, we need only to change the content of the knowledge data file. We may even create a user interface to facilitate the editing of the knowledge data base. Figure 7 gives another example of knowledge data base that concerns soil conditions. Further discussion on the knowledge data bases will be presented later.

As a final note on rules in FLOPS, observe the first *rule* command of rule r0 (shown in Figure 4). A number, 1,000, appears immediately after the key word *rule*. This number is referred to as the priority of the rule or the prior confidence level of the rule. In FLOPS, the confidence level is encoded as an integer with a maximum value of 1,000, which actually means a confidence of 100 percent. When the LHS is evaluated, it also returns a confidence value. The smaller of the two confidence values is taken as the posterior confidence level. All actions involving memory updating in the RHS of that rule are assigned this posterior confidence value.

The PS1 system utilizes this feature to assign the membership grade of a member of a fuzzy set. The *modify* command in Figure 6 is an example. The generated rule shown in Figure 6 has a prior confidence level of 600. If the evaluation of LHS based on actual user input data returns a confidence of, say, 1,000, then the smaller of the two values would be 600, and this value is assigned to the fuzzy set member ^criterion.CIPM as its membership grade. The prior confidence level of the

```
:comment - rule r0 to generate block 2 rules

rule 1000 ( DL  ^lower1 = <LB>  ^upper1 = <UB>  ^fsmember = <FSM>

                ^confidence = <CONF> )

    -->

    rule <CONF> 2 (xdata  ^design_load >= <LB>  ^design_load <= <UB>)

                (pile_type  ^criterion.<FSM> = 0)
          -->
            modify 2  ^criterion1.<FSM> ;
```

FIGURE 4 The content of rule r0 of PS1.

```
:comment -- DL = design load (tons)

make DL    ^lower1  0.        ^upper1  20.  ^fsmember "CIPM"
           ^confidence  600;

make DL    ^lower1  0.        ^upper1   20.  ^fsmember "TM"
           ^confidence  1000;

   .
   .
   .
```

FIGURE 5 Partial list of contents of file DL.kb of PS1.

```
rule 600 2 (xdata  ^design_load ^>=  (0, 0, 0)

                   ^design_load ^<=  (20, 2, 0.1))

          (pile_type  ^criterion1.CIPM = 0)

   -->

   modify 2  ^criterion1.CIPM;
```

FIGURE 6 An example of a rule generated by rule r0.

```
:comments
:factor1 = soil type (cohesive, cohesionless)
:factor2 = strength of bearing soil (low, medium, high)
:factor3 = negative skin friction (likely, unlikely)
:factor4 = presence of boulders (yes, no)

make SOIL   ^factor1 "cohesive"  ^factor2 "low"  ^factor3 "unlikely"

            ^factor4 "no"  ^fsmember "PC"  ^confidence 900;

make SOIL   ^factor1 "cohesive"  ^factor2 "low"  ^factor3 "unlikely"

            ^factor4  "no"    ^fsmember  "STL"    ^confidence  500;
  .
  .
  .
```

FIGURE 7 Partial list of contents of file SC.kb of PS1.

generated rule does not have to be 1,000. In the real world there often exists some essential knowledge that is less certain than other knowledge. It may be desirable to include this less-certain knowledge in the reasoning process. PS1 incorporates this desired feature by embedding the uncertainty (confidence value) in the knowledge data base.

The confidence (or uncertainty) of a piece of knowledge reflects an expert's opinion on that piece of knowledge. For a system to be efficient, it is necessary to set up a cutoff confidence value that determines whether a particular piece of uncertain knowledge should be incorporated into the system. If no prior experience exists, a sensitivity study should be conducted. This approach was taken in the·development of PS1.

It is noted that the second *rule* command in Figure 4 has a number 2 beside 1,000. This is referred to as a block number. When that number does not appear, as in the case of the first *rule* command, the system assigns a number of 0. The block numbers are generally used to group rules for some rule firing control. It is a useful feature, especially for inductive reasoning in the FLOPS environment.

Input Section

The input section basically consists of at least a *make* command. The *make* command is used for noninteractive input or initiation of the elements and their attributes. The *run* command, although it can be issued from anywhere in FLOPS environment, is usually placed in the input section. This command causes execution of the rule section. The input section may include other FLOPS commands for specific purposes. All commands are executed sequentially in the input section.

As mentioned earlier, all production rules are grouped into blocks. By controlling the block firing sequence, the rules may be fired in some planned order. However, no particular order is set for the rules within a block. In fact, with parallel processing, all fireable rules will be fired at once, regardless their order of appearance.

The program structure shown in Figure 2 was implemented in this input section. First, the system reads in the knowledge data files with *open* command. It then sets up a control mechanism to execute each block of rules sequentially. The system begins with execution of block 0 under the command *run* The rules in block 0 generate all possible production rules. The system then executes block 1, which gathers problem-specific data by calling the external program GETDATA. Actions taken in block 1 also make blocks 2 through 8 fireable. These blocks are then fired sequentially under the next *run* command. An external data file, which contains the preliminary conclusions reached by PS1, is created after firing of these blocks. The last *run* command in the input section causes execution of block 9. This block calls an external program COMBINE to consolidate the preliminary conclusions. The final conclusion is then reported and the program stops.

Binary Logic vs. Fuzzy Logic

Notice that with the implemented structure described above, a rule will be actually fired if and only if all of the following conditions are met:

1. the block in which the rule resides is switched on;
2. the elements used in the LHS of the rule have been initiated with proper *make* commands; and
3. the LHS of the rule is evaluated to be "true."

The evaluation of the LHS of the rule begins with each individual comparison implemented in the LHS. Each comparison returns a truth value, which in binary logic takes the value of 0 (for false) or 1 (for true). The smallest of all values returned by the comparisons is taken as the confidence level of the LHS. In traditional comparison based on binary logic, this value will be either 0 or 1. In other words, the term "true" in the third condition stated above requires a confidence value of 1 (or in PS1 notion, 1,000/1,000 or simply 1,000).

The necessity of including uncertain but essential knowledge into the data base was discussed above. In a similar manner, PS1 adopts the fuzzy comparison feature whenever appropriate. The rule shown in Figure 6 provides an example of its potential advantage. The LHS consists of comparisons in two objects (elements), design_load and pile_type. The comparison in this rule is binary; it will return either 0 (false) or 1,000 (true). If a given datum of design_load is, say, 21, the comparison would return a value of 0 and the rule won't be fireable. On the other hand, a given datum of design_load of 19, although not much different from the value of 21, will return a value of 1,000 from the comparison. Such drastic change is a drawback of the binary logic, in which a proposition must be either true or false.

When fuzzy comparison is desired, the rule shown in Figure 6 may be revised into the one shown in Figure 8. A fuzzy comparison will return a value of between 1,000 (true) and 0 (false). In other words, a proposition can be partially true. Note that a fuzzy operation indicator, ~, preceding the comparison operators, such as < and ≤, was used in the new rule. This indicator tells the program to make a fuzzy comparison. Also notice that for fuzzy comparison on two scalar numbers, FLOPS requires two additional data: an absolute uncertainty and a relative uncertainty. For example, the second fuzzy comparison in the LHS of the rule shown in Figure 8 is to be carried out using an absolute uncertainty of 2 and a relative uncertainty of 0.1. The confidence returned by this fuzzy comparison is calculated in the manner described in the next paragraph.

FLOPS assumes that the attribute—in this case, the design load—is a normal variate with a mean value of 20 and a standard deviation that is determined as follows:

$$s = [A^2 + (Rm)^2]^{1/2}$$

where

s = standard deviation,

```
rule 600 2 (xdata  ^design_load >= 0.   ^design_load <= 20. )
           (pile_type  ^criterion1.CIPM = 0)
     -->
    modify 2  ^criterion1.CIPM;
```

FIGURE 8 An example of generated rule with fuzzy comparison.

A = absolute uncertainty = 2 in this example,
R = relative uncertainty = 0.1 in this example, and
m = mean value = 20 in this example.

The confidence value resulting from such fuzzy comparison then becomes a simple matter of determining a probability. For a design load of much larger than 20, the value will be very close to 0; for a design load of about 20, the value will be about 0.5 (500/1,000); for a design load of much smaller than 20, the value will be very close to 1 (1,000/1,000).

EXTERNAL PROGRAMS OF PS1

External programs used in the PS1 system are treated as commands in the DOS environment and, as such, they communicated with the FLOPS program through a *call* command, with name of the executable program as the only argument. For example, the RHS of the rule shown in Figure 9 consists of two calls to the DOS commands. One is an executable program GETDATA, treated as a command. The other is a true DOS command *pause*. Although FLOPS allows for a direct call by reference (address) to a program written in C, it is considered to be advantageous to adopt the former method for this particular expert system.

The two external programs used are GETDATA and COMBINE, both written in C language and compiled by using Microsoft C 5.1 compiler (*30*) for use in the DOS environment. It should be noted that any DOS-based C compiler may be used for compiling. Once the program is compiled, it can run without the presence of the compiler.

The program GETDATA is used for gathering problem-specific data on a particular project. GETDATA is itself a complete program and can be run separately in the DOS environment. In fact, it is often run separately to create the data file to be used in PS1. The program GETDATA essentially serves as a user interface to the PS1 system. A segment of the screen output when running GETDATA is shown in Figure 10 to give the flavor of the program.

The program COMBINE is used for consolidating the preliminary conclusions reached by the PS1 system. The data needed for running the program COMBINE are created by the system and stored in an external file called pile_type.dat, which is an ASCII stream file. The data in file pile_type.dat represent the preliminary conclusions reached by the PS1 system. These data are the degrees of confidence toward each member of the fuzzy set attributes. As defined in the PS1 system, the members of these attributes are PC, PSC, CIPM, CIP, STL, TM, CPW, CPS, PIC, and BP, as defined above. Each preliminary conclusion is reached based on each and every one of the seven factors (criteria) employed. An example of a possible conclusion is as follows:

```
:comments
:block #1 - for gathering problem-specific data from the user
:rule r27

rule 1000 1 (start)
      -->
      write '\n**********************************************\n',
      write '   Begin to gather the problem_specific data. \n',
      write '**********************************************\n',
      call GETDATA,
      transfer xdata from user.dat,
      write '\nUser-supplied data have been loaded to PS1.\n',
      call pause,
      make pile_type;
```

FIGURE 9 An example of calling DOS commands from PS1.

```
Use previously-created data file ("user.dat")? (y/n)
n

ANSWER ALL QUESTIONS ASKED ...

Is data on DESIGN LOAD PER PILE known
or can be estimated? (y/n)
y

Estimated or required design load per pile (tons) =?
25
.
.
.
Which one of the descriptions is more-or-less the most
accurate on SUBSURFACE SOIL PROFILE?
  1) very deep soft layer, 2) soft layer underlain by
  medium to stiff layer, 3) soft layer underlain by hard
  stratum, 4) why?
(Enter 1, 2, 3, or 4):
2

What is the confidence of your answer on last question?
  1) absolutely sure, 2) very sure, 3) sure
(Enter 1, 2, or 3):
1
.
.
.
```

FIGURE 10 Segment of a screen output when running GETDATA.

$$\begin{aligned} \text{criterion1} = \{&0.6/\text{PC},\ 0.6/\text{PSC},\ 0.95/\text{CIPM},\ 0.5/\text{CIP},\\ &0.1/\text{STL},\ 0.5/\text{TM},\ 0.95/\text{CPW},\ 0.9/\text{CPS},\\ &0.1/\text{PIC},\ 0.2/\text{BP}\} \end{aligned}$$

where the values are the confidences toward the individual members. (In FLOPS notation, the value 0.6 is stored as 600, 1.0 as 1,000, and so on.) In the above example, it may be interpreted that the PS1 system strongly supports the selection of CIPM, CPW, and CPS piles; it gives moderate support to selection of PC, PSC, CIP, and TM piles, and almost no support to the others.

The algorithm implemented in the current version of the program COMBINE for consolidating the preliminary conclusions is a simple weighted average method. During the development of the PS1 system, other algorithms such as FLOPS' weakly monotonic logic, weighted fuzzy union, and fuzzy weighted average (*8,9*) were considered. It was decided to implement the above algorithm on this version of COMBINE for its simplicity. It was found that PS1 is working properly with this algorithm. Future versions of COMBINE might adopt other algorithms.

KNOWLEDGE DATA BASES OF PS1

The subject of extracting knowledge from experts is beyond the scope of this paper. However, a brief overview is in order for interested readers.

As mentioned earlier, knowledge may be extracted from relevant literature or through interviews with experts. In general, the former may be used for fast prototyping of the desired system. It is generally done with intention to upgrade the knowledge data base later. This route is particularly suitable for cases where the system framework is more or less dependent on the knowledge data base. The system designer is also acting as a domain expert to determine what knowledge to incorporate, to what degree of certainty (or uncertainty) a piece of knowledge can become useful, and how to represent the knowledge (rule-based, frame-based, or others). This is the route taken in the development of PS1.

On the other hand, knowledge may be extracted from an expert or a group of experts. There are obvious advantages taking this route. For one, any system developed will perform only as well as the knowledge stored in it. This route requires very thorough planning and skillful interviews. There are some common methods in practice, but discussion of the subject is beyond the scope of this paper.

A comprehensive review of pile foundation literature was conducted during the development of PS1. It was decided that the first version of the system would deal only with the preliminary selection of pile type, and that the capacity to do design analysis and to make cost comparison will follow later. Findings of that review were documented in detail elsewhere (*31*). A summary is presented in the paragraphs that follow.

For the intended system, it was decided that 10 load-bearing piles will be covered. These pile types were listed above. The focus of the review is to determine under what circumstances a pile will be considered suitable for the project. Based on overall evaluation, seven factors (criteria) were identified as "knowledge" important to the preliminary selection. They are design load requirements, soil parameters, approximate pile length requirement, availability of pile material, local construction experience, vibration and noise effects. Obviously, on a given project these factors might weigh differently. It was decided that the weight will be handled in the system rather than in the knowledge data base. The current version of PS1 allows for the user to select the default setting or input these weights at run time. This feature makes a sensitivity study, if desired, easy to conduct.

Selection of pile type is based on the seven factors (criteria) mentioned above. Users of the developed system need only to input the required information concerning these factors through an interactive, menu-driven program. The rating scale for each criterion is stored in the knowledge base in terms of confidence level. Determination of the confidence level was based on "averaged opinions" obtained from literature review. A "preference rating" for each pile is obtained for a given set of site and design information according to each criterion. With the preference rating based on each criterion determined and weights among the seven criteria selected, the overall preference rating (in terms of a numerical index ranging from 0 to 1) can be computed for selecting each of the pile candidates considered for a given set of conditions. The user then has an option to print out the overall preference ratings of top three pile selections or all piles considered.

Among the seven factors adopted, the subsurface soil parameter is perhaps the most complicated one. Many soil characteristics might affect the selection of a pile, and indeed most of them were seriously considered for inclusion in the system. The current version of PS1, however, adopts only four general subfactors under this category: soil type, strength description, possibility of negative skin friction, and possibility of undesired conditions (such as presence of boulders). For other factors incorporated in the system, the situation is simpler. Only a rating scale, qualitative or quantitative, is needed.

Having established the system framework, it was decided that the factual knowledge will be coded in a data base like the one shown in Figure 5. The next task was to assign the confidence value for selecting each pile based on each factor under a given circumstance. It was decided any entry with a confidence value of less than 0.3 should be discarded. Also, the uncertainty associated with fuzzy comparison of the rules in PS1 was arbitrarily taken as a uniform 15 percent variation, although this datum could be directly included in the data base. Following the above general principles, a total of seven data bases were created as part of the PS1 system.

EXAMPLE

As a hypothetical example, consider a site consisting of a deep layer of loose sand overlying a deep layer of dense sand. Site investigation reveals no boulders in the ground. The design load per pile is estimated to be 25 tons. The length of pile is estimated to be 40 to 50 ft. There is no noise constraint, but a vibration constraint is present. The local contractor is knowledgeable in most types of piling construction except for auger-placed concrete piling, composite wood-concrete piling, and pressure-injected piling. No particular emphasis is placed on any pile selection criterion (factor).

With the above information input through the execution of GETDATA, the PS1 system began its internal reasoning process and reached a conclusion. The top selections recommended were a cast-in-place pile with mandrel and a timber pile. For this rather general description of the site, however, the support for other piles is also strong.

Although not shown in the paper, many examples were worked out with PS1 and the results were reasonable (*31*). For a preliminary pile selection, PS1 seems to be able to make right choice.

CONCLUDING REMARKS

The paper has documented details of the development of the expert system PS1. In particular, rule generation, inductive reasoning, combination of preliminary conclusions, and treatment of fuzzy comparison were discussed in depth. The experience gained and presented in this paper should be helpful to interested readers. Departing from the original intent because of space limit, the entire PS1 system and the screen output during its execution are not listed in this paper. However, the system can be obtained from the authors.

PS1 is working properly and is able to reach reasonable conclusions. However, further examination and calibration by experts is needed before it can be claimed as a reliable expert system. For that reason, the authors consider it at present a prototype of the intended system. Planned improvements to the system include additional quantitative soil parameters for the pile selection, as well as the design and cost analyses.

ACKNOWLEDGMENT

The writers wish to express their gratitude to the reviewers of this paper, whose comments have made it a better one.

REFERENCES

1. W. Siler and D. Tucker. *FLOPS Program and User Manual, Version 1.2c.* Kemp-Carraway Heart Institute, Birmingham, Ala., 1986.
2. L. A. Zadeh. *Fuzzy Sets. Information and Control*, Vol. 8, 1965, pp. 338–353.
3. L. A. Zadeh. Outline of A New Approach to the Analysis of Complex Systems and Decision Process. *IEEE Transactions on Systems, Man and Cybernetics SMCX-3*, 1973, pp. 28–44.
4. C. Brown. A Fuzzy Safety Measure. *Journal of Engineering and Mechanics*, ASCE, Vol. 105, No. EM5, 1979, pp. 855–872.
5. C. Brown and J. T. P. Yao. *Fuzzy Sets in Structural Engineering. Journal of Structural Engineering*, ASCE, Vol. 109, No. 5, 1983, pp. 1211–1225.
6. C. H. Juang and D. J. Elton. Fuzzy Logic for Estimation of Earthquake Intensity Based on Building Damage Records. *Civil Engineering Systems*, Vol. 3, 1986, pp. 187–191.
7. C. H. Juang, J. L. Burati, and S. N. Kalidindi. A Fuzzy System for Bid Proposal Evaluation Using Microcomputers. *Civil Engineering Systems*, Vol. 4, 1987, pp. 124–130.
8. C. H. Juang. Development of A Decision Support System Using Fuzzy Sets. *Journal of Microcomputers in Civil Engineering*, Vol. 3, 1988, pp. 157–166.
9. P. W. Mullarkey and S. J. Fenves. Fuzzy Logic in a Geotechnical Knowledge-Based System: CONE. *Proc., NSF Workshop on Civil Engineering Applications of Fuzzy Sets*, Purdue University, Lafayette, Ind., 1985, pp. 126–169.
10. J. E. Bowles. *Foundation Analysis and Design*, 4th ed., McGraw-Hill, New York, 1988.
11. J. L. Briaud and L. M. Tucker. Piles in Sand: A Method Including Residual Stresses. *Journal of Geotechnical Engineering*, ASCE, Vol. 111, No. 11, 1984, pp. 1666–1680.
12. Canadian Geotechnical Society. *Canadian Foundation Engineering Manual*, chaps. 19-21. BiTech Publishers, Vancouver, B.C., Canada, 1985.
13. R. D. Chellis. *Pile Foundation*, 2nd ed. McGraw-Hill, New York, 1961.
14. R. S. Cheney and R. G. Chassie. *Soils and Foundations Workshop Manual.* FHWA, U.S. Department of Transportation, Washington, D.C., 1982.
15. H. M. Coyle and R. Castello. New Design Correlations for Piles in Sand. *Journal of Geotechnical Engineering Division*, ASCE, Vol. 107, No. 7, 1981, pp. 965–986.
16. B. M. Das. *Principles of Foundation Engineering*, chap. 8. PWS Engineering, Boston, 1984.
17. B. H. Fellenius. *Negative Skin Friction on Long Piles Driven in Clay*. Report 18. Royal Swedish Academy of Engineering Science, Stockholm, 1971.
18. F. M. Fuller. *Engineering of Pile Installations*. McGraw-Hill, New York, 1983.
19. D. M. Greer and W. S. Gardner. *Construction of Drilled Pier Foundations*. John Wiley and Sons, New York, 1986.
20. G. G. Meyerhof. Bearing Capacity and Settlement of Pile Foundations. *Journal of Geotechnical Engineering*, ASCE, Vol. 102, No. GT3, 1976, pp. 195–228.
21. R. L. Nordlund. Bearing Capacity of Piles in Cohesionless Soils. *Journal of Soil Mechanics and Foundation*, ASCE, Vol. 89, No. 3, 1963, pp. 1–35.
22. *NAVFAC Design Manual, Foundations and Earth Structures*, Vol. 7.2. Naval Facilities Engineering Command, U.S. Department of Navy, Washington, D.C., 1982.
23. M. W. O'Neill. *Pile Group Prediction Symposium: Summary*. FHWA, U.S. Department of Transportation, Washington, D.C., 1987.
24. M. G. Spangler and R. L. Handy. *Soil Engineering*, 4th ed. Harper and Row, New York, 1982.
25. A. G. Thurman. Discussion of "Bearing Capacity of Piles in Cohesionless Soils," by R. L. Nordlund. *Journal of Soil Mechanics and Foundation*, ASCE, Vol. 90, No. 1, 1964, pp. 127–129.
26. S. N. Vanikar. *Manual on Design and Construction of Driven Pile Foundations*. FHWA, U.S. Department of Transportation, Washington, D.C., 1986.
27. A. S. Vesić. *NCHRP Synthesis of Highway Practice 42: Design of Pile Foundations*. TRB, National Research Council, Washington, D.C., 1977.
28. T. Whitaker. *The Design of Piled Foundations*. Pergamon, New York, 1976.
29. C. H. Juang and D. H. Lee. Development of an Expert System for Rock Mass Classification. Accepted for publication by *Journal of Civil Engineering Systems*, 1989.
30. *Microsoft C Compiler, Version 5.1*. Microsoft Corporation, Redmond, Wash., 1988.
31. C. H. Juang and M. L. Ulshafer. *Development of a Fuzzy Logic Expert System for Pile Selection*. Department of Civil Engineering, Clemson University, Clemson, S.C., 1989.

Publication of this paper sponsored by Committee on Foundations of Bridges and Other Structures.