Journal of International Information Management

Volume 14 | Issue 2 Article 2

2005

Inference-Guiding for Intelligent Agents

Jinchang Wang
The Richard Stockton College

Inference-Guiding for Intelligent Agents

Jinchang Wang
The Richard Stockton College of New Jersey

ABSTRACT

In many applications of intelligent agents, initially given facts are not sufficient to reach a decision, and more data are needed. In that case, Inference-guiding is needed to identify the missing information and lead inference to a conclusion. This paper presents a new inference-guiding strategy that selects the key pieces of missing information in such a way that the total cost of acquiring additional information for reaching a conclusion is the lowest. The computational experiments show that the new strategy is more effective and economical than the inference-guiding strategies currently available for the intelligent systems.

INTRODUCTION

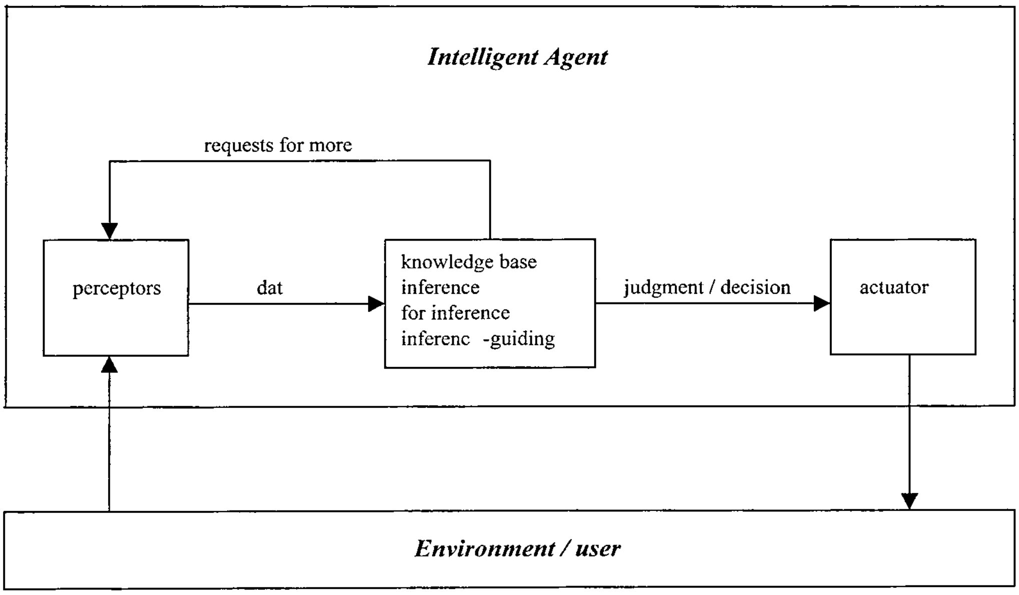
The '*intelligent agent*', a term in artificial intelligence, refers to a device or a system that can, to some extent, 'think' as human beings and 'act' rationally. A robot is a typical example of an intelligent agent. An intelligent agent as a node in a computer network handles the information transmitted through the node. Other examples include an automatic real-time control mechanism, a computerized system for disease diagnosing, debugging, or professional training. Comparing to an expert system and a knowledge-based system, which are computer systems that store human's knowledge and mimic the human's logic to solve problems in certain domains, an intelligent agent is more self-contained, more autonomous, and more action-oriented.

An intelligent agent has three major functions: perceiving the environment, making judgments / decisions, and acting. Percepts of environment are obtained through *perceptors* such as various sensors, data input channels from a database, and devices for inputs from users. Judgments and decisions are made in the '*brain*' of the intelligent agent, which contains a knowledge base and an inference engine. The knowledge base stores knowledge. The *inference engine* is software for doing logical inference. The third function, acting, is to execute the decision or announce the judgment through *actuators*. An actuator is a device such as an arm of a robot, a control mechanism, and a natural language output device.

In many applications of intelligent agents, such as diagnosing, training, and real-time control, initial data about the environment are often incomplete, because of huge amount of the 'complete' data and the cost of obtaining them. The cost of obtaining data can be monetary cost or cost of time. To get information from electrocardiograms and CT, for example, would cost money, and to ask a patient questions would cost time. If no solid conclusions can be made due to insufficient data, an intelligent agent should be able to pinpoint the missing data, make hypothesis, collect more information, and prove the hypothesis. In this sense, an intelligent agent should be not only a 'thinker' but also an 'investigator'. Just as a human inspector has to do investigation when initial clues for a case are not sufficient. An intelligent agent for disease diagnosing, which makes judgment on the ailment that a patient may have, must also do investigations by, for example, asking the patient to provide more information on symptoms and selecting some examinations for the patient to do, when the information on hand is not sufficient to get any solid conclusion. The process of 'investigation' is called *inference-guiding (IG)*, or *question-asking* [Wang et al. (1990)], which pursues more information about the environment for further inference. A good IG process would be able to identify a few relevant and key missing data in an efficient way, and lead inference quickly to a conclusion. A bad IG process, on the other hand, would ask irrelevant, costly, and silly questions, and retard the inference process.

The function of inference-guiding resides in the inference engine of an intelligent agent. The inference engine should be able to, like human being, figure out what information should be pursued if the current known data are not sufficient. A human inspector ought to be good in not only analysis but also investigation. That is, he should be good in exploring the meaning of the clues he has by logically linking them, and reaches a conclusion if the clues are sufficient; and he should also be good in investigations, if the clues are not sufficient, by figuring out the key missing data and collecting the data to prove his hypothesis. An inference engine of an intelligent agent thus has two tasks. One is *logical inference* (or simply *inference*), which is to make deductions to reveal logical implications of the known information and to make the judgment or decision. Another task is *inference-guiding*, which is to identify the missing data if given facts are not sufficient to reach any solid conclusion. Figure 1 shows an intelligent agent with inference-guiding function and its interaction with the environment.

Figure 1. The intelligent agent with inference guiding function.



Logical inference is a subject that has been studied extensively. The Davis-Putnam algorithm [Davis et al. (1960)] and DPLL backtracking algorithm [Davis et al. (1962)] were among the earliest effective algorithms for propositional knowledge bases. Thereafter Robinson developed the full resolution rule [Robinson (1965)]. Due to the close relation between propositional inference and the satisfiability problem (SAT), all the algorithms developed for SAT are actually working for propositional inference either [Gu et al. (1997)]. Modus ponens [Bonissone (1993)] [Awad (1996)] is a deductive rule among implications. Forward chaining (or data-driven) and backward chaining (or goal-driven) [Turban et al. (2001)] are two alternative methods for controlling inference in rule-based intelligent systems, based on which there are many variations and extended technologies, such as the production system [Palopoli et al. (1997)] and deductive database [Ramakrisshnan et al (1995)] [Ullman (1989)]. The Jeroslow-Wang algorithm [Jeroslow et al. (1990)] utilized the integer programming techniques into logical inference. That algorithm was further improved in [Wang (1997)] and [Wang (1998-B)].

Comparing to logical inference, inference-guiding has been less explored. But still some research results have been achieved. EXPERT used pre-listed orderings of rules and questions [Hayes-Roth et al. (1983)]. KAS, a shell over PROSPECT, used both forward and backward chaining, together with a scoring function, for picking more relevant missing data [Duda et al. (1979)]. Mellish's procedure [Mellish (1985)], using a so-called 'Alpha-beta pruning technique', eliminated irrelevant questions for acyclic inference nets. Wang and Vande Vate proved that the problem of identifying the fewest key questions was computationally hard even in a Horn system, and they developed a heuristic algorithm for Horn systems [Wang et al. (1990)]. A cost-effective IG strategy for Horn systems was developed by

Wang and Triantaphyllou [Wang et al. (1996)]. For the propositional systems, Wang's IG algorithm in [Wang (1998-A)] aimed at selecting fewer questions. The algorithm in [Wang (2005)] took the cost factor into account.

We present a new IG approach in this paper, which is an improvement of the algorithm in [Wang (2005)]. Section II introduces fundamental concepts and terms. Section III discusses the general process of so-called 'top-level-conclusion oriented inference' in an intelligent agent. In Section IV, we review three currently existing IG algorithms. The new IG strategy is presented in Section V, and the results of computational experiments shown in Section VI. Section VII discusses the managerial implications and applications of the new IG strategy.

FUNDAMENTALS

A *predicate* names a relationship between objects. If each object is an assertion (or a statement), whose value is either true or false, then the predicate is called a *propositional formula*, or simply a *proposition*. The relationship between assertions can be "OR" ($\vee$), "AND" ($\wedge$), "IMPLY" ($\rightarrow$), "EQUIVALENT" ($\leftrightarrow$). The negation is denoted as $\neg$. A *literal* is a propositional assertion or its negation. A *truth valuation* of a proposition is a set of value assignments to all assertions.

A propositional formula B is in a *conjunctive normal form (CNF)* if it is a conjunction of formulas B_i ($1 \leq i \leq t$), where each B_i, called a *clause,* is a disjunction of literals. A *unit clause* is a clause consisting of one literal. Any propositional formula can be transformed into a CNF. A disjunctive clause can be put into an *implication*, i.e., a form of "if...then..." rule, and vice versa, by applying the relation $(B_1 \vee B_2) = (\neg B_1 \rightarrow B_2)$. In an implication $B_1 \rightarrow B_2$, B_1 is called the *premise* and B_2 is called the *conclusion.*

In applications of intelligent agents, inference is carried out to prove some goals that are called *top-level-conclusions (TLC).* For example, possible faults in an engine are TLCs in a diagnosing system for maintenance of an aircraft; and required operating adjustments are TLCs of a real-time piloting control system. An assertion is *observable* if its value can be obtained from the environment or user directly with no inference is needed. For example, the assertions about the result of an X-ray examination and about the answer of a patient to a question from the doctor are observable. An observable assertion is called an *unconfirmed observable assertion (UOA)* if its value is not yet obtained. Each UOA has a *cost* that indicates the cost to obtaining its value, which is called *questioning cost* or *confirmation cost.*

A literal L is *derived* (*reached* or *proved*) if and only if $\{K \wedge F \rightarrow L\}$ is a tautology (i.e., it is true for any truth valuation), where K is the knowledge base; F is the set of known facts. For example, $K = \{\neg A_3 \rightarrow A_2, A_4 \wedge A_2 \rightarrow A_1\}$ is a knowledge base, $F = \{A_3 = \text{false}, A_4 = \text{true}\}$ is the given fact set. We can see that every truth valuation of A_1, A_2, A_3, and A_4 would make $\{K \wedge F \rightarrow L\} = \{((\neg A_3 \rightarrow A_2) \wedge (A_4 \wedge A_2 \rightarrow A_1) \wedge (\neg A_3) \wedge (A_4)) \rightarrow (A_1)\}$ true. In this case we say A_1 is derived. But if $F = \{A_3 = \text{false}\}$, then $\{K \wedge F \rightarrow L\} = \{((\neg A_3 \rightarrow A_2) \wedge (A_4 \wedge A_2 \rightarrow A_1) \wedge (\neg A_3)) \rightarrow (A_1)\}$ is no longer a tautology, since the truth valuation $\{A_1 = \text{false}, A_2 = \text{true}, A_3 = \text{false}, A_4 = \text{false}\}$ would make it false. In this case, we say A_1 is not derived (not implied by K and F).

There is an alternative way of defining derivation of a literal. A literal is *derived* (*reached* or *proved*) if it must be logically true given the knowledge base and the known facts. In the example of the last paragraph, $K = \{\neg A_3 \rightarrow A_2, A_4 \wedge A_2 \rightarrow A_1\}$ and $F = \{A_3 = \text{false}, A_4 = \text{true}\}$ would derive A_1 because A_1 must be logically true per to K and F. But if $F = \{A_3 = \text{false}\}$, then A_1's value can be either 'true' or 'false', therefore A_1 is not derived. A derived literal is called a *logical consequence* of the knowledge base and the given facts.

Suppose K is a knowledge base in which knowledge is represented by implications (i.e., if...then... rules). There are n assertions, $A_1, A_2, ..., A_n$, and m if...then... rules (or clauses) in K. We use the symbol R(i,j) to denote the j-th rule whose conclusion is L_i. For an observable assertion A_i, we reserve R(i,0) to denote a clause of fact "A_i is true" or "A_i is false". Thus each assertion, including each UOA, appears as a conclusion in some rule. A *proof* δ is a 2n-dimensional vector such that for each literal L_i, $R(i,\delta(i))$ is a rule concluding L_i. A proof δ gives a possible way of proving each assertion. The *deriving cost* of a literal associated with a proof is the sum of the questioning costs of the UOAs in the proof.

If there is at most one positive literal in a disjunctive clause, then the clause is called a *Horn clause.* A Horn clause in form of implication is that a set of positive assertions implies one positive assertion. A conjunction of Horn clauses is called a *Horn system.*

We define an *unconfirmed observable assertion set (UOA-set)* of A_i in a Horn system as a set of UOAs such that if all the UOAs were confirmed true, then A_i would be proved, but if any one were false, then A_i could not be concluded from the others in this set. A UOA-set thus is a minimal set of UOAs that could derive a TLC. An assertion A_i may have many UOA-sets associated with it. Each UOA-set of A_i corresponds to a proof δ in such a way the UOA-set contains the UOAs in the proof δ, and the proof δ indicates how A_i can be proved by the UOA-set.

TLC-ORIENTED INFERENCE

TLC-oriented inference refers to the intellectual process that starts with some data and aims at deriving a TLC (top-level-conclusion). Note the difference between ordinary logical inference and TLC-oriented inference. Ordinary logical inference does not have a particular 'goal' during inference. It is just to derive, based on the given facts, as many logical consequences as possible. It stops when it cannot derive any more logical consequences. TLC-oriented inference, on the other hand, has TLCs as the 'goal'. It will continue as far as no TLC is derived. In the circumstance that ordinary logical inference has to stop due to failing to generate new logical consequences, TLC-oriented inference would seek more information from the environment to make logical inference continue. TLC-oriented inference has a lot of applications, particularly in the intelligent systems for problem diagnosing, troubleshooting, consulting, and real-time controlling.

TLC-oriented inference has a counterpart in human's problem-solving process whose goal is '*solving a problem*'. '*Data analysis*' is the first phase in the human's problem-solving process, which makes a thought on whether the problem can be solved with data currently on hand. If not, then the second phase, '*investigation*', must be carried out to find more information. Data analysis and investigation are carried out alternately until the problem is solved. The corresponding two phases in TLC-oriented inference are ordinary logical inference and inference-guiding. Let K denote the knowledge base, F the set of known facts. The process of TLC-oriented inference is as follows:

TLC-oriented Inference:

Alternately run the two phases until a TLC is derived:

Phase 1. *Logical inference (inference).*

To see whether any TLC can be derived as a logical consequence of F and K. That is, whether $K \wedge F \rightarrow T$ *where T is a TLC. If so, Stop, we are done, otherwise go to Phase 2.*

Phase 2. *Inference-Guiding.*

Pick up a UOA and obtain its value from some information source (such as the user, a sensor, or a meter). After getting its value, put the new fact into F and go to Phase 1.

The above TLC-oriented inference tells what to do. It does not tell how to do. Many algorithms have been developed for 'how' in either phase, as we reviewed in Section I. In this paper, we focus on 'how' to do Phase 2, inference-guiding.

The IG problem is an optimization problem by its nature since it seeks the missing data that would contribute to reaching a TLC fast and economically. On the other hand, the IG problem is computationally hard [Wang et al. (1990)], which means that there is no known efficient algorithm to identify the 'optimal' questions to ask. Given the hardness of the IG problem, we develop heuristic algorithms for IG, which pursue "good", instead of the "best", results in an efficient way.

REVIEW OF THREE CURRENT IG STRATEGIES

Inference-guiding is Phase 2 of the TLC-oriented inference process. Our goal is to develop more effective and efficient algorithms for inference-guiding. In this section, we review three currently existing IG strategies.

Wild Randomness IG Strategy (WRS)

The *wild randomness IG strategy (WRS)* selects next question from the pool of current UOAs randomly. Let UPL denote the pool of UOAs. UPL initially contains all UOAs. The wild randomness IG method is as follows.

```
Step 1. Pick up a UOA, $P_k$ for example, from UPL randomly;
Step 2. Obtain the value v of $P_k$ from an information source, and put its value into the fact set F.
        Let UPL=UPL/{$P_k$}.
```

The main advantage of WRS is its simplicity. It just picks a UOA randomly. Its weakness is obvious, - it may select many irrelevant questions.

Backward Chaining IG Strategy (BCS)

The *backward chaining IG strategy (BCS)* is widely used in the knowledge-based systems that apply backward chaining [Luger et al. (1989)] [Horowitz (1978)] as the logical inference algorithm. It starts with a TLC and takes it as a goal to pursue. It then looks at the rules that can be used to prove the goal. The premises of such a rule are then taken as subgoals to pursue. This process goes backward from a TLC, until a UOA is encountered, and that UOA is the question to be asked. This method is stated formally as following.

```
Step 0: Place a TLC in a stack S.
Step 1: Pop out a literal from the top of the stack S.
        If the literal is observable and its value is known, go to beginning of Step 1;
        if the literal is a UOA, go to Step 2;
        Otherwise, select a rule that can prove it and put the premise assertions of the rule into the stack S, go
                to beginning of Step 1.
Step 2: Ask about the selected UOA, obtain its value, and put its value into the fact set F.
```

This strategy is better than the wild randomness IG strategy because it aims at a TLC all the time, which guarantees that all the questions are relevant to a possible TLC. However, it still involves randomness. A TLC is randomly selected in Step 0. And when a goal or subgoal can be proved by many rules, a rule is selected randomly. Moreover it does not consider total cost or total number of questions when selecting questions.

Cost-Reducing IG Strategy (CRS)

Let K be a propositional knowledge base which is composed of "if...then..." rules. Let K_{Horn} be the knowledge base, which is the same as K except that all the negation signs "$\neg$" are removed. K_{Horn} is a Horn clause system that is referred to as *the associated Horn system* of propositional knowledge base K. A set of UOAs is called a *PH-UOA-set (pseudo-Horn-UOA-set)* of K if it is a UOA-set in K_{Horn}.

Identifying a good PH-UOA-set in Horn system K_{Horn} is easier than identifying a good UOA-set in propositional system K. To find a good PH-UOA-set of low cost, a heuristic labeling algorithm is used on K_{Horn}, which calculates the cost (more exactly, the upper bound of the cost) of reaching each goal and subgoal. After a low-cost PH-UOA-set that could reach a TLC is identified, questions are selected from the PH-UOA-set to ask, until: (a) an answer is obtained and it is not in favor of proving the TLC through the proof associated with the selected PH-UOA-set, and another PH-UOA-set has to be identified; or (b) all UOAs in that PH-UOA-set are asked, and answers are all in favor of proving the TLC through the proof associated with the PH-UOA-set, then the TLC is proved.

The Cost-Reducing IG Strategy with PH-UOA-Set (CRS) is outlined as follows.

***Step 1**. Convert the current propositional system K into K_{Horn} by removing all the negation signs of assertions.*
***Step 2**. Identifying a PH-UOA-set U in K_{Horn} by using the heuristic developed in [Wang (2005)].*
***Step 3**. Select a UOA A_k from the PH-UOA-set U; Get the value of A_k from an information resource about the value of A_k; Put its value into the fact set F.*

The heuristic of identifying a PH-UOA-set in Step 2 is a labeling algorithm to find a good UOA-set in the Horn system. It labels each assertion with the smallest upper-bound-cost of obtaining its value in a forward process; then picks a TLC with the smallest label and identifies the corresponding UOA-set in a backward process. This IG strategy was put forward in [Wang (2005)]. The experiments showed that this IG strategy is substantially better than BCS and WRS.

THE NEW IG STRATEGY (NewS)

Even though the strategy CRS performed significantly better than WRS and BCS, CRS contains two approximations in selecting a UOA-set. One is using the upper bound of the minimum cost of proving an assertion, which is represented by the label of an assertion, to approximate the minimum cost of proving the assertion. Another is using the associated Horn system, instead of the original propositional system. The two approximations make CRS pick up a UOA fast, though it may reduce the quality of the UOA-set.

The second approximation, using the associated Horn system, has a major weakness. The PH-UOA-set selected by CRS is not necessarily leading to a TLC. For example, we have a small propositional system with three clauses $K=\{A_2\wedge\neg A_3\rightarrow A_1, A_3\wedge A_4\rightarrow A_2, A_4\wedge\neg A_5\rightarrow\neg A_3\}$ in which A_1 is TLC, A_4 and A_5 are UOAs. The value of TLC A_1 is not forced to be true or false, so it is not derived. The Horn system associated with the propositional system is $K_{Horn}=\{A_2\wedge A_3\rightarrow A_1, A_3\wedge A_4\rightarrow A_2, A_4\wedge A_5\rightarrow A_3\}$, by removing all negation signs. One can see that $\{A_4,A_5\}$ is a PH-UOA-set in K_{Horn}. However, no matter what values A_4 and A_5 take, the TLC A_1 is not forced to be either true or false in K. This weakness may mislead inference and cost more by asking irrelevant questions that lead to no TLCs. How often such misleading may happen depends on the specific knowledge bases.

The new IG strategy improves CRS by making sure that each UOA-set identified would be associated with a TLC. We define a *propagation-UOA-set* in a propositional system as a set of UOAs that can be identified by using backward chaining so that if these UOAs took certain values then a TLC would be proved, and the process of backward chaining takes an assertion and its negation as two different assertions. A propagation-UOA-set will always be associated with a TLC, since it is derived by using backward chaining directly on the original system. Another advantage of using the propagation-UOA-set is that it is computationally efficient to identify a good propagation-UOA-set.

The new IG strategy is to select a low-cost propagation-UOA-set and pick up a UOA from the set to ask. We use a labeling algorithm, which is an extension of the labeling algorithm for the Horn system, to identify a small propagation-UOA-set.

The following is the procedure of the new IG algorithm, NewS. Let K denote the knowledge base that is in form of "if...then..." rules, D(i) the index set of rules that have literal L_i as the conclusion, I(i,d) the index set of the premises of rule R(i,d), Cost(i) the questioning cost of UOA A_i. Note that here we are dealing with literals rather than assertions since we take an assertion and its negation separately. There are n assertions in the knowledge base K, therefore there are 2n literals such that $L_i=A_i$, $L_{n+i}=\neg A_i$, for i=1,2,...,n.

The New IG Strategy with Propagation-UOA-Set (NewS).

```
BEGIN {Main} ;
Step 1. Identifying a propagation-UOA-set of K:
        Label the unconfirmed literals and rules by using the procedure LABELING, taking an assertion and its
        negation separately;
        Select a potential TLC $A_c$ that has the smallest label;
        Trace back from $A_c$ and find the propagation-UOA-set, say U, which is associated with smallest label;
Step 2. Asking Question:
        Select a question $A_k$ to ask from the propagation-UOA-set U;
        Get the value of $A_k$ from an information resource.
END {Main} ;
Procedure LABELING ;
BEGIN { LABELING } ;
Step 1: Label each UOA $A_i$ as $C_i$=Cost(i).
Step 2: Alternately use procedure PROPAGATE and procedure ASSIGN to label the literals and clauses until no
        new labels are applied in either subroutine.
END { LABELING } ;
Procedure PROPAGATE ;
BEGIN { PROPAGATE } ;
  While there is an unlabeled element t:
  If the element t is a rule R(i,d) and all its premises are labeled, then label rule R(i,d) as:
        $C_{i,d} = \Sigma_{j \in I(i,d)} C_j$
  If the element t is a literal $L_i$ and all the rules with $L_i$ as the conclusion are labeled, then label literal $L_i$ as:
        $C_i = C_{i,\delta(i)}$
        where $\delta(i) \in D(i)$ is such that
        $C_{i,\delta(i)} = MIN_{d \in D(i)} \{C_{i,d}\}$.
END { PROPAGATE } ;

Procedure ASSIGN ;
BEGIN { ASSIGN } ;
  Among all labeled rules whose conclusions are not labeled, choose the one, say (k,d*), with the smallest label.
        That is:
        $C_{k,d^*} = MIN_{R(i,d) \text{ labeled and } A_i \text{ unlabeled}} \{C_{i,d}\}$.
  Let: $C_k = C_{k,d^*}$
END { ASSIGN } .
```

The characteristics of the new IG strategy NewS include the follows:

(a) The calculations of the labels are in the original propositional system, rather than the associated Horn system, so that the selected UOA is guaranteed to be relevant to a TLC.

(b) It takes an assertion and its negation as two different assertions in inference-guiding to make sure that the propagation-UOA-set identified is associated with a TLC. For example, in a small database K={$A_4 \wedge \rightarrow A_2$, $A_3 \rightarrow \neg A_2$, $A_2 \rightarrow A_1$} with A_1 as the TLC, A_4 and A_3 as UOAs, the propagation-UOA-set identified by NewS is {A_4}, which is associated with TLC A_1. But if we did not differentiate an assertion from its negation, then we would have two propagation-UOA-set, {A_4} and {A_3}, in which {A_3} would not be associated with TLC A_1 at all. However, in logical inference, an assertion and its negation must not be taken as two independent assertions, since they are closely related semantically: If A is known to be true then ¬A must be false, and vice versa.

(c) It uses the upper bound of the cost of deriving the value of an assertion as the criterion of labeling the assertion. That is, each assertion's label is the minimum of the upper bounds of the total costs to derive the assertion's value. The reason of doing so is to make the algorithm efficient in identifying a propagation-UOA-set.

(d) It allows the knowledge base to have cycles. It is not uncommon for a knowledge base to have cycles. A robust algorithm should be able to work on knowledge bases with cycles. In this IG strategy, cycles are dealt with in the sub-routine ASSIGN.

(e) The computational complexity of the algorithm is log-linear to the size of the knowledge base in terms of the number of assertions [Wang et al. (1990)].

However, the propagation-UOA-set has its weaknesses. In a propositional system, not all UOA-sets are propagation-UOA-sets. In other words, not all UOA-sets can be found by backward chaining propagation. For example, $\{\neg A_4 \wedge A_6 \rightarrow A_3, \neg A_3 \wedge \neg A_5 \rightarrow A_1, A_4 \wedge \neg A_6 \rightarrow A_1, A_3 \wedge \neg A_5 \rightarrow A_1\}$ is a small knowledge base in which A_1 is TLC, A_4, A_5 and A_6 are UOAs. There are two propagation-UOA-sets, $\{A_4, A_6\}$ and $\{A_4, A_5, A_6\}$, which can be identified by backward propagation. $\{A_5\}$ is not a propagation-UOA-set. But $\{A_5\}$ is a UOA-set, since if A_5's value is "false", then TLC A_1 must be 'true' (i.e., A_1 is derived) no matter the values of A_4 and A_6.

Moreover, a propagation-UOA-set does not guarantee to be a 'minimal' set of UOAs that could derive a TLC. In the above example, $\{A_4, A_5, A_6\}$ is a propagation-UOA-set, but it is not a minimal set of UOAs for deriving TLC A_1 since just A_5 itself is sufficient to derive TLC A_1, if A_5="false", without using A_4 and A_6.

Despite the weaknesses of the propagation-UOA-set, we expect NewS to be more effective than CRS because NewS eliminates the waste of picking up a UOA-set that does not lead to any TLC in any cases. We need computational experiments to verify this hypothesis and to quantify the improvements.

COMPUTATIONAL EXPERIMENTS

We have carried out computational experiments to test the new IG strategy NewS by comparing it with the three existing ones reviewed in Section IV. In the experiments, inference scenarios were generated randomly. We used the Jeroslow-Wang (JW) algorithm [Jeroslow et al. (1990)] with modification [Wang (2003)] for logical inference. The experiments were run on Gateway 2000 personal compute Pentium 350. Programs were written in Borland C^{++} 4.5.

We created knowledge bases of "if...then..." clauses randomly with parameters including number of clauses, number of propositional assertions, length of a clause, number of UOAs, and number of TLCs. The sign of an assertion (i.e., whether it is a negation) was determined randomly with half-by-half chance. The questioning cost of a UOA was determined randomly with 90% chance in the range of [1, 100], and 10% chance in the range of [1001, 2000]. The purpose to put the cost structure in this way was to test whether an IG strategy could be intelligent enough so that the high cost UOAs were avoided unless they had to be used for inference-guiding.

Table 1 shows the experiment results on 200 randomly generated knowledge bases that are clustered into twenty groups. Each group contains ten knowledge bases of same values of parameters in number of clauses, number of assertions, length of a clause, number of UOAs, and levels of rules. The strategies are compared upon number of questions asked (*Questions*), total questioning cost of deriving a TLC (*TCost*), average cost per UOA questioned (*Cost/Que*), CPU time for logical inference (*InfTime*) which is for Phase 1 of the TLC-oriented process, and CPU time for inference-guiding (*IGTime*) which is for Phase 2 of the TLC-oriented process. In the table, *WRS* stands for the wild randomness IG strategy, *BCS* for the backward chaining IG strategy, *CRS* for cost-reducing IG Strategy, and *NewS* for the new IG strategy with propagation-UOA-set. The five columns on the left provide the knowledge base parameters, where *n* stands for number of clauses, *m* for number of assertions, *len* for the maximum number of assertions in a clause (i.e., length of a clause), *UOAs* for number of UOAs, *TLCs* for number of top-level-conclusions. The sixth column, *KBs*, gives number of knowledge bases tested in a group.

Table 1: Experiment Results on Four IG Strategies.

n=number of clauses, *m*=number of assertions, *len*=max length of a clause
UOAs=number of UOAs, *TLCs*=number TLCs
KBs=number of knowledge bases with the parameters on the left.

n	m	len	UOAs	TLCs	KBs		WRS	BCS	CRS	NewS
100	100	3	50	5	10	Questions	23.3	14.8	10.5	9.2
						TCost	5934.05	2848	741.05	479.25
						Cost/Que	254.68	192.43	70.58	52.09
						InfTime	2.315	1.523	1.105	0.978
						IGTime	0.035	0.078	0.073	0.077
100	100	4	50	5	10	Questions	28	15.2	12.4	10.2
						TCost	5035.85	3736.3	841.1	778.9
						Cost/Que	179.85	245.81	67.83	76.36
						InfTime	2.877	1.636	1.317	1.096
						IGTime	0.033	0.075	0.077	0.09
100	100	3	40	5	10	Questions	30	16.5	14.5	11.2
						TCost	6104.75	3774.9	1521.75	1405.9
						Cost/Que	203.49	228.78	104.95	125.53
						InfTime	2.98	1.693	1.522	1.179
						IGTime	0.028	0.076	0.085	0.102
200	200	3	100	10	10	Questions	51.6	22.2	7.6	6
						TCost	10353.4	4211.4	195.3	159.3
						Cost/Que	200.65	189.70	25.70	26.55
						InfTime	13.693	6.403	2.261	1.826
						IGTime	0.135	0.286	0.123	0.15
300	300	3	150	15	10	Questions	68.8	27.8	15.4	12.1
						TCost	12997.6	4724.6	307.3	247.7
						Cost/Que	188.92	169.95	19.95	20.47
						InfTime	35.825	16.029	9.337	7.436
						IGTime	0.231	0.675	0.456	0.6
500	500	3	250	25	10	Questions	84.9	25.1	12.4	10
						TCost	17647.4	4421.6	206.2	173.9
						Cost/Que	207.86	176.16	16.63	17.39
						InfTime	124.271	40.792	21.142	17.403
						IGTime	0.508	1.46	0.86	1.25
800	800	3	400	20	10	Questions	111.6	22.2	11.2	10.8
						TCost	21997.2	3723.9	169.5	158.2
						Cost/Que	197.11	167.74	15.13	14.65
						InfTime	226.052	51.298	26.366	25.424
						IGTime	1.285	3.13	1.651	3.007
00	200	4	100	10	10	Questions	61.6	13.3	9.4	8
						TCost	11382.4	2517.3	234.4	214.7
						Cost/Que	184.78	189.27	24.94	26.84
						InfTime	16.687	4.204	3.051	2.612
						IGTime	0.149	0.188	0.167	0.229
500	350	3	175	9	10	Questions	66.6	19.4	14.1	10.1
						TCost	13718.5	4161.9	289.5	190.8
						Cost/Que	205.98	214.53	20.53	18.89
						InfTime	30.731	10.067	7.35	5.425
						IGTime	0.387	0.751	0.674	0.857

n	m	len	UOAs	TLCs	KBs		WRS	BCS	CRS	NewS
						TCost	6120.1	3634.1	261.9	243.3
						Cost/Que	168.13	200.78	25.93	26.16
						InfTime	9.978	5.142	3.039	2.845
						IGTime	0.122	0.305	0.22	0.325
600	400	3	200	15	10	Questions	55.2	20.1	12.1	9.3
						TCost	10912.5	3014.4	196.7	148.8
						Cost/Que	197.69	149.97	16.26	16.00
						InfTime	51.287	19.891	12.448	9.791
						IGTime	0.44	0.968	0.758	1.066
500	300	3	150	10	10	Questions	42.4	21.1	9.7	7.3
						TCost	7868.6	3845.8	139.3	121.2
						Cost/Que	185.58	182.27	14.36	16.60
						InfTime	22.05	11.375	5.616	4.305
						IGTime	0.288	0.705	0.446	0.567
600	400	3	200	11	10	Questions	50.1	18.3	9.9	6.6
						TCost	9612.3	3508.5	175.1	115.2
						Cost/Que	191.86	191.72	17.69	17.45
						InfTime	34.745	13.744	7.705	5.159
						IGTime	0.427	0.917	0.637	0.768
600	300	3	150	9	10	Questions	48.8	15.7	11.3	8.2
						TCost	8616.1	2479.3	170.5	128.7
						Cost/Que	176.56	157.92	15.09	15.70
						InfTime	25.736	9.275	6.944	5.178
						IGTime	0.369	0.575	0.572	0.732
600	300	4	150	9	10	Questions	57.7	25.4	13.4	11.2
						TCost	12237.05	4956.1	315	301.9
						Cost/Que	212.08	195.12	23.51	26.96
						InfTime	32.647	16.572	9.182	7.843
						IGTime	0.452	1.008	0.696	0.987
400	200	5	100	6	10	Questions	53.7	26.1	9.4	7.2
						TCost	10200.9	4759.1	248.6	198.1
						Cost/Que	189.96	182.34	26.45	27.51
						InfTime	16.505	9.437	4.7	3.808
						IGTime	0.271	0.615	0.334	0.384
500	250	5	125	7	10	Questions	63.6	31.9	23.1	19.2
						TCost	14399.2	7130.1	1270.2	943.2
						Cost/Que	226.40	223.51	54.99	49.13
						InfTime	26.397	15.889	11.488	10.475
						IGTime	0.393	0.988	0.82	1.131
600	300	5	150	9	10	Questions	68.8	32.1	13.2	11.5
						TCost	15572.1	6974.4	351.7	338.3
						Cost/Que	226.34	217.27	26.64	29.42
						InfTime	42.792	23.26	11.346	10.498
						IGTime	0.543	1.321	0.762	1.079

n	m	len	UOAs	TLCs	KBs		WRS	BCS	CRS	NewS
600	300	5	200	10	10	Questions	69.5	34.8	12.1	10.3
						TCost	15181.6	7822.6	327.4	298.2
						Cost/Que	218.44	224.79	27.06	28.95
						InfTime	43.279	25.368	11.096	10.312
						IGTime	0.528	1.408	0.695	1.006
800	400	5	200	10	10	Questions	86.3	51.1	10.3	9.4
						TCost	16223.7	8459	260.9	239
						Cost/Que	187.99	165.54	25.33	25.43
						InfTime	74.854	50.687	13.081	12.224
						IGTime	0.959	3.157	0.9	1.433
Avg. Questions (UOAs asked to reach a TLC)							57.9	23.6	12.1	9.9
Avg. TCost (total questioning cost to reach a TLC)							11605.77	4535.17	411.17	344.23
Avg. Cost/Que (cost per UOA asked)							200.29	192.49	33.97	34.93
Avg. InfTime (inference time to reach a TLC)							41.785	16.714	8.505	7.291
Avg. IGTime (IG time to reach a TLC)							0.379	0.934	0.550	0.792

We summarize our observations and analyses of the experiment results as follows:

(1) CRS vs. NewS:

The average total questioning cost to reach a TLC was 344.23 for NewS, compared to 411.17 for CRS. NewS asked 9.9 questions on average to reach a TLC, while CRS asked 12.1. But the two strategies had similar costs per UOA asked (Cost/Que). That implies that the power of saving cost by NewS, comparing to CRS, resides in selecting more relevant UOAs to ask, rather than lower-cost UOAs.

CRS and NewS had similar capacities of selecting low-cost UOAs. But NewS was better in selecting fewer and more relevant questions, which made its total cost of deriving a TLC (TCost) lower. That verifies our hypothesis that replacing the PH-UOA-set by the propagation-UOA-set can reduce the wastes by asking fewer questions.

(2) BCS and WRS vs. NewS:

(2.a) BCS and WRS asked much more questions. To reach a TLC, BCS asked 23.6, WRS asked 57.9, while NewS asked only 9.9 questions;

(2.b) BCS and WRS cost much more to reach a TLC. To reach a TLC, it cost BCS 4,535.17, WRS 11,605.77, and NewS only 344.23.

Comparing to BCS and WRS, NewS was able to not only choose fewer UOAs, but also avoid high cost questions and pick up lower cost questions.

(3) In the experiments, 90% of UOAs cost 1-100 (with average at 50), 10% of UOAs cost 1,001-2,000 (with average at 1,500), uniformly distributed in each range. So, the overall expected cost per question was 1,500*10%+50*90%=195.

Look at the average Cost/Que (cost per UOA asked) for each strategy:

WRS:	200.29;	BCS:	192.49;
CRS:	33.97;	NewS:	34.93.

(3.a) WRS and BCS were running around the overall expected cost, 195, per question. That was because neither strategy considers the cost in the process of picking up a question.

(3.b) CRS and NewS were running at the average cost per question significantly lower than 195, and even lower than 50 that is the expected questioning cost for the group of 90% low-cost questions. That was because both strategies not only managed to avoid the 10% high-cost questions, but also were able to select the lower-cost questions within the group of 90% low-cost questions.

(4) BCS vs. WRS

Although both WRS and BCS are strategies of randomness, WRS cost much more to reach a TLC than BCS (11,605.77 vs 4,535.17). That was because WRS did not use the concept of UOA-set at all, whereas BCS selected questions from an 'implicit' UOA-set, so that WRS asked more questions (57.9) than BCS (23.6), though the cost per question of WRS was similar to that of BCS.

There was no explicit procedure in the BCS strategy to select a UOA-set, but BCS did identify a UOA-set implicitly before picking up a UOA, since a UOA identified by the backward chaining procedure in BCS must be in some UOA-set. However, such a UOA-set was pick up in BCS randomly. In this sense, BCS could be named as 'random UOA-set IG strategy'.

(5) CPU time:

The *total CPU time* to reach a TLC on average is the sum of average logical inference time and average inference-guiding time, i.e., total CPU time = (avg. InfTime) + (avg. IGTime). For each of the four strategies, the total CPU time was:

WRS:	42.164 seconds;	BCS:	17.648 seconds;
CRS:	9.055 seconds;	NewS:	8.083 seconds.

CRS and NewS used substantially less time overall to reach a TLC. The main reason was the number of questions. All the four strategies used a same inference procedure – JW algorithm. The difference of total inference times (InfTime) among the four strategies, therefore, reflected the difference in number of questions asked (since after each question is asked, inference procedure is applied to see whether a TLC is reached).

The CPU times for inference-guiding (IGTime) were relatively small comparing to the total CPU times:

WRS:	0.379 seconds;	BCS:	0.934 seconds;
CRS:	0.550 seconds;	NewS:	0.792 seconds.

They were the CPU times for selecting UOAs to ask about. Note that, in terms of complications of IG procedure, the four strategies are ranked as NewS, CRS, BCS, WRS, from the most complicated to the simplest. WRS used least time in UOA selection since the selection procedure was extremely simple, even though the number of questions selected was the highest among the four methods. CRS and NewS took shorter time in UOA selection because number of questions needed was small by CRS or NewS, even though the IG procedures of CRS and NewS were more complicated than BCS. NewS took longer time than CRS because the complication of the IG procedure, even though NewS asked fewer questions.

(6) Total CPU time vs. total process time.

Table 1 records only the CPU times for inference and inference-guiding. But it usually takes an intelligent agent much more time to derive a TLC. The time of acquiring the value of the selected UOA from an information source is not counted in Table 1. Let us define the *total process time (TPT)* as the sum of total CPU time and the data acquisition time. The total CPU time is the 'on-line' time, while the data acquisition time is the 'off-line' time. In most cases, the off-line time is much longer than the on-line time. And the data acquisition time is proportional to number questions asked. So, taking the off-line time of acquiring data into account, the new strategy NewS would be even better and more efficient than the other three strategies since NewS asked fewest questions to reach a TLC.

(7) There seems no pattern that shows the advantages of NewS over CRS in terms of the size of a knowledge base. But the advantages of NewS over WRS and BCS were more significant as number of UOAs getting larger. It could

be explained as that when number of UOAs was getting larger, there were more choices in UOA selection, therefore a better chance for a lower-cost question to be selected by NewS.

(8) In summary, NewS was the most economical method in terms of cost of reaching a TLC, and the time to reach a TLC by NewS was faster than those by the other three current IG strategies, especially when the off-line time of acquiring data was taken into account.

MANAGERIAL IMPLICATIONS AND APPLICATIONS

Since the intelligent agents and intelligent systems have many applications in management, the improved new IG approach has managerial implications by making those intelligent systems smarter in figuring out the key missing data and reaching the conclusion quickly and economically. Here are examples of managerial circumstances in which the new IG approach can be applied.

(a) Systems for consulting and management decision support: A good IG strategy is required for such systems to keep asking relevant and to-the-point questions, keep avoiding silly and irrelevant ones, and provide professional advices.

(b) Systems for training / education: A training/ education intelligent system should be trainee/student–centered. The new IG approaches makes the system know quickly about individual trainees / students by asking a few questions.

(c) Intelligent systems for problem-diagnosing or trouble-shooting for management, especially for emergency management: IG is essential to quickly and economically diagnose where the 'trouble' is, and get the problem solved.

(d) Real-time control systems in aircrafts, space crafts, atomic reactors, and chemical processes: In case some unexpected situation occurs, the new IG approach helps obtain key relevant information about what happens and how to react in short time.

(e) Intelligent searching in a huge database: The new IG approach helps narrow down the searching domain, when matching an inexact piece of information or a blurred image in a huge database.

(f) Distributed database systems: The new IG approach helps the central database management system to figure out which sub-systems should be called and what information should be asked in response to an information request of inter-system optimization.

(g) Intelligent agents: Since an intelligent agent is an intelligent system capable of independent actions, a good IG approach is essential for every movement / action by quickly making judgment on the situation and determining what to do.

CONCLUSION

Inference-guiding is very important for the efficiency and effectiveness of an intelligent agent. Searching for the optimal solution in the IG problem, however, is computationally hard. The new IG approach presented in this paper, NewS, applies the concept of the propagation-UOA-set directly on the original propositional system, and is more effective and economical in selecting key missing data than the three existing strategies in our computational experiments.

The new IG approach makes the intelligent systems in management more intelligent when the given data are not sufficient to reach a conclusion. Management may benefit from the new IG approach by receiving more effective decision supports of the intelligent systems.

Further research on inference-guiding includes developing more accurate criteria for selecting the UOA-set and UOA, integrating uncertainties of knowledge and data into IG approaches, and exploring inference-guiding on other knowledge base structures such as the Bayesian network [Russell et al. (2003)].

REFERENCES

Awad, E. M. (1996). Building Expert Systems: Principles, Procedures, and Applications. Minneapolis/St. Paul, MN:West Publishing

Bonissone, P. P. (1993). Knowledge Representation and Inference in First-generation Knowledge-based Systems, in M. Grabowski and W. A. Wallace (eds.) Advances in Expert Systems for Management, Vol. 1, Greenwich, CT:JAI Press

Cook, S. A. (1971). The Complexity of Theorem-Proving Procedures, Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, pp.151-158, New York. ACM Press

Duda, R., J. Gasching and P. Hart (1979). "Model design in the PROSPECT Consultant System for Mineral Exploration, Expert Systems in the Micro-Electronic Age," Edinburgh Univ. Press, UK

Davis, M. and H. Putnam (1960). A Computing Procedure for Quantification Theory, Journal of the Association for Computing Machinery, 7(3), 201-215

Davis, M., G. Logemann, and D. Loveland (1962), A Machine Program for Theorem-Proving, Communications of the Associations for Computing Machinery, 5, 394-397

Gu, J, P.W. Purdom, J. Franco and B.W. Wah (1997). Algorithms for Satisfiability (SAT) Problem: A Survey. DIMACS volume series on Discrete Mathematics and Theoretical Computer Science: The Satisfiability Problem, American Mathematical Society

Hayes-Roth, E., D.A. Waterman and D.B. Lenat (1983). "Building Expert Systems," Addison-Wesley, Reading, MA

Horowitz, E and S.Sahni (1978). Fundamentals of Computer Algorithms. Rockville, MD, Computer Science Press

Jeroslow, R. G. and J. Wang (1990). Solving Propositional Satisfiability Problems, Annals of Mathematics and Artificial Intelligence, 1, 167-187

Luger, G. F. and W.A.Stubblefield (1989). Artificial Intelligence and the Design of Expert Systems, The Benjamin/Cummings Publishing Company, Inc.

Mellish, C.S. (1985), "Generalized Alpha-beta Pruning as a Guide to Expert System Question Selection," Expert System 85, Proc. 5th Technical Conf. of the British Computer Society Specialist Group on Expert Systems, Univ. of Warwick, (Cambridge Univ. Press, Cambridge CB2 1RP)

Palopoli, L. and R. Torlone (1997). Generalised Production Rules as a Basis for Integrating Active and Deductive Databases, IEEE Transactions on Knowledge and Data Engineering, 9, #6

Robinson, J. A. (1965). A Machine-Oriented Logic Based on the Resolution Principle, Journal of the Association for Computing Machinery, 12, 23-41

Russell, S., and P. Norvig (2003). Artificial Intelligence, a Modern Approach, Prentice Hall, Pearson Education Inc., Upper Saddle River, New Jersey

Ramakrisshnan, R. and J. D. Ullman (1995). A Survey of Research in Deductive Database Systems, Journal of Logic Programming, 23(2), 125-149

Turban, E. and J. Aronson (2001). Decision Support Systems and Intelligent Systems, Prentice Hall, Upper Saddle River, New Jersey

Ullman, J. D. (1989), Principles of Database and Knowledge-Based Systems, Computer Science Press, Rockville, Maryland

Wang, J. (2005). A Cost-Reducing Question-Selection Algorithm for Propositional Knowledge-Based Systems, Annals of Mathematics and Artificial Intelligence, 44, 35-60

Wang, J. (2003). A Weight-Balanced Branching Rule for SAT. Mathematical and Computer Modeling, 38, 595-609

Wang, J. (1998-A). Inference Guiding in Propositional Knowledge Bases, Annals of Mathematics and Artificial Intelligence, 23, 345-356

Wang, J. (1998-B). Backtracking Tactics in the Backtrack Method for SAT, Mathematical and Computer Modeling, 28, No. 2, 1-12

Wang, J. (1997). Branching Rules for Propositional Satisfiability Test, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 35, 351-364

Wang, J. and E.Triantaphyllou (1996). A Cost-Effective Question-asking Strategy for Horn Clause Systems. Annals of Mathematics and Artificial Intelligence, 17, 359-380

Wang, J., and J. Vande Vate (1990). "Question-Asking Strategies for Horn Clause Systems," Annals of Mathematics and Artificial Intelligence, 1, 359-370