Interval-Type and Affine Arithmetic-Type Techniques for Handling Uncertainty in Expert Systems

Martine Ceberio[1], Vladik Kreinovich[1], Sanjeev Chopra[1], Luc Longpré[1], Hung T. Nguyen[2], Bertram Ludäscher[3], and Chitta Baral[4]

[1]NASA Pan-American Center for Earth & Environmental Studies
University of Texas, El Paso, TX 79968, USA
contact email vladik@cs.utep.edu
[2]Math. Dept., New Mexico State University
Las Cruces, NM 88003, USA
[3]Department of Computer Science,
University of California, Davis , CA 95616, USA
[4]Department of Computer Science & Engineering
Arizona State University, Tempe, AZ 85287-5406, USA

Abstract

Expert knowledge consists of statements S_j (facts and rules). The facts and rules are often only true with some probability. For example, if we are interested in oil, we should look at seismic data. If in 90% of the cases, the seismic data was indeed helpful in locating oil, then we can say that if we are interested in oil, then with probability 90% it is helpful to look at the seismic data. In more formal terms, we can say that the implication "if oil then seismic" holds with probability 90%. Another example: a bank A trusts a client B, so if we trust the bank the bank A, we should trust B too; if statistically, this trust was justified in 99% of the cases, we can conclude that the corresponding implication holds with probability 99%.

If a query Q is deducible from facts and rules, what is the resulting probability $p(Q)$ in Q? We can describe the truth of Q as a propositional formula F in terms of S_j, i.e., as a combination of statements S_j linked by operators like &, $\vee$, and $\neg$; computing $p(Q)$ exactly is NP-hard, so heuristics are needed.

Traditionally, expert systems use technique similar to straightforward

interval computations: we parse F and replace each computation step with corresponding probability operation. Problem: at each step, we ignore the dependence between the intermediate results F_j; hence intervals are too wide. Example: the estimate for $P(A \vee \neg A)$ is not 1. Solution: similarly to affine arithmetic, besides $P(F_j)$, we also compute $P(F_j \,\&\, F_i)$ (or $P(F_{j_1} \,\&\, \ldots \,\&\, F_{j_d})$), and on each step, use all combinations of l such probabilities to get new estimates. Results: e.g., $P(A \vee \neg A)$ is estimated as 1.

1 Formulation of the Problem

Expert knowledge usually consists of statements S_j: facts and rules. The main objective is, given a query Q, to check whether Q follows from the expert knowledge. In this paper, we will use the standard Prolog-type notations, in which a statement that a is true is described as $a \leftarrow$, and a rule "if a_1, a_2, ..., and a_m are true then b must be true" is described as $b \leftarrow a_1, a_2, \ldots, a_m$.

For example, in the knowledge base

$$S_1 : a \leftarrow b, \quad S_2 : b \leftarrow, \quad S_3 : a \leftarrow c, \quad S_4 : c \leftarrow$$

S_1 is a rule "if b then a", S_3 is a rule "if c then a", S_2 is a fact (b is true), and S_4 is a fact (c is true). If we ask a query $Q \stackrel{\text{def}}{=}$ "a?", then the answer is "yes": since we know that b is true, and that b implies a, we can conclude that a is true. In other words, Q follows from S_1 and S_2. Prolog-type inference engines are tools that provide such inference; see, e.g., [10].

The problem with this approach is that the experts' facts and rules are often only true with some probability. If a query Q is deducible from facts and rules, what is the resulting probability $p(Q)$ that Q is actually true? For example, in a geophysical situation, we may have the following two rules: to find oil, we must look for certain subterranean structures; to find these structures, we must analyze gravity data. If these rules were absolutely true, then we would be able to conclude that to find oil, we must analyze gravity data. In reality, we know that in search for oil, looking for specific subterranean structures is only helpful in 80% of the cases, so the first rule is true with probability 0.8. We also know that the gravity data can detect only 90% of such structures, so the second rule is only true with probability 0.9. What is the resulting probability that gravity data will help in a specific search for oil?

Let us describe this problem in more precise terms. We can usually describe deducibility of Q as a propositional formula F in terms of S_j, i.e., as a combination of statements linked by operators like "and" (&), "or" ($\vee$), and "not" ($\neg$). For example, for the above knowledge base, for Q to be true, either both S_1 and S_2 must be true, or both S_3 and S_4 must be true. In this case, $F = (S_1 \,\&\, S_2) \vee (S_3 \,\&\, S_4)$. The general algorithm for describing such a propositional formula is given in [7].

As a result, we arrive at the following problem:

- we have a propositional combination F of known statements S_j;
- we know the probabilities $p(S_j)$ of different statements;
- we must determine the probability $p(F)$.

Since the events S_j may be statistically dependent, we may get different values for $p(F)$ depending on whether the values are independent or, say, positively correlated. So, to be more precise:

- we must determine the interval $\mathbf{p}(F)$ of possible values of $p(F)$.

How is this problem solved now?

2 Traditional Approach

It is known that in general, the problem of finding the exact bounds for $p(F)$ is NP-hard; see, e.g., [10]. This problem is NP-hard even if all the probabilities $p(S_j)$ are equal to 1, because it is equivalent to the propositional satisfiability problem, a known NP-hard problem.

Traditionally, expert systems use technique similar to straightforward interval computations [6]. Namely, for simple formulas we know the corresponding probability bounds [12]: if we know the bounds $[\underline{a}, \overline{a}]$ for $p(A)$ and $[\underline{b}, \overline{b}]$ for $p(B)$, then:

- $p(\neg A)$ is in the interval $[1 - \overline{a}, 1 - \underline{a}]$;
- $p(A \,\&\, B)$ is in the interval $[\max(\underline{a} + \underline{b} - 1, 0), \min(\overline{a}, \overline{b})]$;
- $p(A \vee B)$ is in the interval $[\max(\underline{a}, \underline{b}), \min(\overline{a} + \overline{b}, 1)]$.

In the general case, we parse F and replace each computation step with the corresponding probability operation.

For example, let $F = (A \,\&\, B) \vee (A \,\&\, \neg B)$ and $p(A) = p(B) = 0.6$. The compiler would start with $F_1 = A$ and $F_2 = B$, then it would compute $F_3 = \neg B$, $F_4 = F_1 \,\&\, F_2$, $F_5 = F_1 \,\&\, F_3$, and finally $F = F_4 \vee F_5$. Thus, according to the above procedure, we first find the bounds for $p(F_3) = p(\neg B)$, then for $p(F_4) = p(A \,\&\, B)$ and $p(F_5) = p(A \,\&\, \neg B)$, and finally, the bounds for $p(F)$. As a result, we get $p(\neg B) = 1 - 0.6 = 0.4$,

$$\mathbf{p}(A \,\&\, B) = [\max(0.6 + 0.6 - 1, 0), \min(0.6, 0.6)] = [0.2, 0.6],$$

$$\mathbf{p}(A \,\&\, \neg B) = [\max(0.6 + 0.4 - 1, 0), \min(0.6, 0.4)] = [0, 0.4],$$

$$\mathbf{p}(F) = [\max(0, 0.2), \min(0.4 + 0.6, 1)] = [0.2, 1.0].$$

What is the actual range of $p(F)$? In this problem, F is equivalent to A, so $p(F) = 0.6$. Thus, similarly to interval computations, we can see that the resulting interval contains excess width.

The second example is to estimate $p(A \vee \neg A)$ for $p(A) = 0.6$. The desired answer is, of course, $p(A \vee \neg A) = 1$. However, when parsing $A \vee \neg A$, we get $F_1 = A$, $F_2 = \neg A$, and $F = F_1 \vee F_2$. So, in the traditional approach, we estimate $p(F_1) = 0.6$, $p(F_2) = 1 - p(F_1) = 1 - 0.6 = 0.4$, and

$$\mathbf{p}(F_1 \vee F_2) = [\max(0.4, 0.6), \min(0.4 + 0.6, 1)] = [0.6, 1]$$

– excess width again.

3 How We Can Improve the Interval Estimates: Idea

We have just seen, on two examples, that the traditional approach leads to excess width. In order to see how we can improve this approach, let us trace where the excess width appears in the estimation of $p(A \vee \neg A)$. We know $p(A) = 0.6$; so, since $F_1 = A$, we know the exact value of $p(F_1)$: $\mathbf{p}(F_1) = [0.6, 0.6]$.

Next, in the traditional approach, we use the known interval $\mathbf{p}(F_1)$ to estimate the interval $\mathbf{p}(F_2)$ for $F_2 = \neg F_1$. Here, $\mathbf{p}(F_2) = 1 - [0.6, 0.6] = [0.4, 0.4]$. This is also the exact value of the corresponding probability.

Finally, in the traditional approach, we use the known bounds $\mathbf{p}(F_1) = [0.6, 0.6]$ and $\mathbf{p}(F_2) = [0.4, 0.4]$ to estimate the bound for $F = F_1 \vee F_2$. At this stage, we do get excess width. The reason for this excess width is that we use a general formula for $\mathbf{p}(A \vee B)$, the formula that only takes into account the intervals $\mathbf{p}(F_1)$ and $\mathbf{p}(F_2)$ and that does not take into account that in this particular case, there is a special relation between the events F_1 and F_2 – these events are incompatible.

In order to take this missing information into account, it is desirable, once we come to a new intermediate result F_j, to not only estimate the interval $\mathbf{p}(F_j)$, but to also estimate intervals $\mathbf{p}(F_j \,\&\, F_k)$, $\mathbf{p}(F_j \vee F_k)$ of possible values of the probabilities of different propositional combinations of F_j and the previous intermediate results F_k.

Next, when we turn to computing similar probabilities involving the next intermediate result F_j, we taken into account not only the known bounds $\mathbf{p}(F_k)$ for $k < j$, but also the known bounds on the probabilities of propositional combinations $\mathbf{p}(F_k \,\&\, F_l)$, $\mathbf{p}(F_k \vee F_l)$, etc., for different pairs $k, l < j$.

4 Our Inspiration: Affine and Taylor Arithmetic Techniques

In interval computations, one way to decrease the excess width is to use affine or Taylor arithmetic; see, e.g., [3, 4, 8]. The main reason for excess width is that when we apply operations from interval arithmetic step-by-step, we only take into account the intervals of possible values of the previous result, and we ignore the possible relation between these results. For example, when we use straightforward interval computations to estimate the range of the function $y = x_1 \cdot (1 - x_1)$ on the interval $[0, 1]$, then we parse the function into $r_1 := x_1$, $r_2 := 1 - r_1$, $y := r_1 \cdot r_2$, and replace each operation with real numbers by the corresponding operation from interval arithmetic. As a result, we get $\mathbf{r}_1 := [0, 1]$; $\mathbf{r}_2 := 1 - [0, 1] = [0, 1]$, and $\mathbf{y} := [0, 1] \cdot [0, 1] = [0, 1]$ – while the exact range is $[0, 0.25]$. The main reason for the excess width is that when we estimated the range of $y = r_1 \cdot r_2$, we took into account the ranges for r_1 and r_2, but not the fact that these variables are actually strongly related: in this particular case, $r_2 = 1 - r_1$.

To take this relation into account, in affine arithmetic, for each intermediate result y_j, we not only keep the interval $\mathbf{y}_j$ of its possible values, we also keep the relation between this value y_j and the values $\Delta x_i \stackrel{\text{def}}{=} x_i - \widetilde{x}_i$ (where $\widetilde{x}_i$ is a midpoint of the input interval $\mathbf{x}_i$) – in the form of a linear dependence $y_j = a_{0j} + a_{1j} \cdot \Delta x_1 + \ldots + a_{nj} \cdot \Delta x_n + \varepsilon_j$, where a_{ij} are exactly known coefficients and for the remainder term ε_j, we know the interval bounds $[-\Delta_j, \Delta_j]$. We start with the original inputs x_i presented in this form, i.e., as

$$x_i = \widetilde{x}_i + 0 \cdot \Delta x_1 + \ldots + 0 \cdot \Delta x_{i-1} + (-1) \cdot \Delta x_i + 0 \cdot \Delta x_{i+1} + \ldots.$$

Then, on each intermediate step, when we use an arithmetic operations $\oplus$ to compute the next intermediate result y_j as $y_k \oplus y_l$, where $k, l < j$, we use known linear representations for y_k and y_l to find a similar representation for y_k. For example, if we know that

$$y_k = a_{0k} + a_{1k} \cdot \Delta x_1 + \ldots + a_{nk} \cdot \Delta x_n + \varepsilon_k$$

and

$$y_l = a_{0l} + a_{1l} \cdot \Delta x_1 + \ldots + a_{nl} \cdot \Delta x_n + \varepsilon_l,$$

with $\varepsilon_k \in [-\Delta_k, \Delta_k]$ and $\varepsilon_l \in [-\Delta_l, \Delta_l]$, then for $\oplus = +$, we get

$$y_j = a_{0j} + a_{1j} \cdot \Delta x_1 + \ldots + a_{nj} \cdot \Delta x_n + \varepsilon_j$$

with $\varepsilon_j \in [-\Delta_j, \Delta_j]$, where $a_{ij} = a_{ik} + a_{kj}$ and $\Delta_j = \Delta_k + \Delta_l$. For multiplication $\oplus = \cdot$, the corresponding formula is more complex because the new bound Δ_j must also include the bound for the new quadratic terms.

The coefficients a_{ij} describe the relation between the intermediate result y_j and the input values $x_1, \ldots, x_n$ – and thus, also between different intermediate results.

Our main idea is similar to affine arithmetic: for each intermediate result F_j, in addition to an interval of possible values for $p(F_j)$, we also compute intervals of possible values for pairs: $p(F_j \,\&\, F_i)$, $p(F_j \vee F_i)$ for all previous expressions F_i and for all possible propositional functions of two variables. These intervals describe the relation between the intermediate results.

Affine arithmetic only takes linear dependencies into account; to account for more complex dependencies, we can use, e.g., quadratic Taylor models, in which we also keep track of the quadratic terms in the dependence of y_j on the inputs $x_1, \ldots, x_n$, i.e., of terms of the type $a_{jkl} \cdot x_k \cdot x_l$. It is also possible to use cubic terms, etc.

Similarly, in our case, in addition to keeping track of the probabilities of propositional combinations of pairs F_j and F_k, we can pick an order $d \geq 2$, and also, on each step, estimate intervals for propositional combinations of k intermediate results, such as $p(F_{j_1} \,\&\, \ldots \,\&\, F_{j_d})$. Similar to Taylor techniques, the higher the order d, the more we take dependencies into account (so, in general, the more accurate the results), but the longer the computations (since we must compute more terms).

5 How We Can Implement This Idea

In order to implement the above idea, we must be able to use the previously known bounds to compute the new ones. This can be done by using *linear programming*; see, e.g., [12]. Indeed, we can describe both known and estimated probabilities as sums of probabilities of atomic statements $F_{i_1}^{\varepsilon_1} \,\&\, \ldots \,\&\, F_{i_m}^{\varepsilon_m}$, where $\varepsilon \in \{-, +\}$, F^+ means F, and F^- means $\neg F$. Then, we use linear programming (LP) to get desired bounds on the unknown probability. There exist efficient polynomial-time algorithms for solving LP problems, so these bounds can be computed efficiently.

Let us first illustrate the use of LP on the example when we already know the analytical solution: we know $p(A) = a = 0.6$ and $p(B) = b = 0.6$ and we want to estimate $p(A \vee B)$. Here, we have two basic statement A and B, so both the known probabilities $p(A)$, $p(B)$, and the desired probability $p(A \vee B)$ can be describe in terms of the probabilities of 4 possible atomic statements: $p_{++} \stackrel{\text{def}}{=} p(A \,\&\, B)$, $p_{+-} \stackrel{\text{def}}{=} p(A \,\&\, \neg B)$, $p_{-+} \stackrel{\text{def}}{=} p(\neg A \,\&\, B)$, $p_{--} \stackrel{\text{def}}{=} p(\neg A \,\&\, \neg B)$. These probabilities must be non-negative, and they must add up to 1.

Specifically, $p(A) = p(A \,\&\, B) + p(A \,\&\, \neg B)$, $p(B) = p(A \,\&\, \neg B) + p(\neg A \,\&\, B)$, and $p(A \vee B) = p(A \,\&\, B) + p(A \,\&\, \neg B) + p(\neg A \,\&\, B)$. So, the known information about the 4 probabilities $p_{++}, \ldots,$ can be described in terms of the following

constraints:

$$p_{++} + p_{+-} = a; \quad p_{++} + p_{-+} = b; \quad p_{++} + p_{+-} + p_{-+} + p_{--} = 1;$$

$$p_{++} \geq 0; \quad p_{+-} \geq 0; \quad p_{-+} \geq 0; \quad p_{--} \geq 0.$$

To find the largest possible value $\overline{p}(A \vee B)$ of the probability $p(A \vee B)$, we must thus maximize the expression $p_{++} + p_{+-} + p_{-+}$ under the above constraints. Similarly, to find the smallest possible value $\overline{p}(A \vee B)$ of $p(A \vee B)$, we must thus minimize the expression $p_{++} + p_{+-} + p_{-+}$ under the above constraints.

In both cases, we must optimize an objective function which is a linear combination of the 4 unknowns $p_{++}, \ldots$, under constraints each of which is either a linear equality or linear inequality. So, both problems are LP problems.

It is known that in general, the solution of a LP problem is attained at one of the vertices of the corresponding set, i.e., when the largest possible number of inequalities become equalities. In this particular case, we have 4 inequalities $p_{..} \geq 0$ which can become equalities. One can easily check that $p(A \vee B)$ is the smallest when $p_{-+} = 0$, and $p(A \vee B)$ is the largest when $p_{--} = 0$. In both cases, we get the desired bounds $\max(a, b) = 0.6$ and $\min(a + b, 1) = 1$.

6 Examples

Let us show that by estimating the bounds for probabilities of pairs, we indeed get narrower intervals.

Let us first show this on the example of $F = A \vee \neg A$ for $p(A) = 0.6$, in which the parsing leads to $F_1 = A$, $F_2 = \neg A$, and $F = F_1 \vee F_2$. In the traditional approach, we estimated $\mathbf{p}(F_1)$, $\mathbf{p}(F_2)$, and then used these two intervals to estimate the bounds for $p(F_1 \vee F_2)$.

In the new approach, we similarly start with $\mathbf{p}(F_1) = [0.6, 0.6]$. On the first step, we handle the intermediate statement F_2. In the traditional approach, we use the logical relation $F_2 = \neg F_1$ between the new intermediate result F_2 and the previous result F_1 to estimate the range of possible values for $p(F_2)$. In the new approach, we use this logical relation not only estimate the bound for $p(F_2)$, but also to estimate the bounds for $p(F_1 \oplus F_2)$ for different propositional combinations $F_1 \oplus F_2$ of F_1 and F_2 (such as $F_1 \,\&\, F_2$, $F_1 \vee F_2$, etc.). Because of this relation between F_i, we have $p(F_1 \,\&\, F_2) = 0$, $p(F_1 \,\&\, \neg F_2) = 0.6$, $p(\neg F_1 \,\&\, F_2) = 0.4$, $p(F_1 \vee F_2) = 1$, $p(F_1 \vee \neg F_2) = 0.6$, $p(\neg F_1 \vee F_2) = 0.4$, and $p(\neg F_1 \vee \neg F_2) = 1$.

According to the new approach, on the next step, when we estimate $p(F)$ for $F = F_1 \vee F_2$, we take into account not only the previously computed bounds on $p(F_1)$ and $p(F_2)$, but also the previously computed bounds on the propositional combinations of F_1 and F_2. Since we already know, from the previous step, that $p(F_1 \vee F_2) = 1$, we thus conclude that $p(F) = 1$. As a result, we get the exact desired probability, with no excess width.

Comment. A general argument against expert systems and fuzzy logic (see, e.g., [9]) is that, e.g., $p(A \vee \neg A)$ is estimated based only on the probabilities $p(A)$ and $p(\neg A)$ – e.g., as $\max(p(A), p(\neg A))$, while the correct value of $p(A \vee \neg A)$ is 1. Our solution: in addition to probabilities of individual intermediate statements, keep probabilities of pairs, triples, etc.

For $(A \,\&\, B) \vee (A \,\&\, \neg B)$, we have 6 intermediate statements F_j, so it is difficult to describe all corresponding propositional combinations of pairs without making the paper too long. Let us show, however, that the resulting bound is indeed narrower than the interval $[0.2, 1]$. Indeed, since $F_4 = F_1 \,\&\, F_2$, on the corresponding step, we conclude that $p(\neg F_1 \,\&\, F_4) = 0$; this conclusion can be confirmed if we explicitly describe the corresponding LP problem. Similarly, since F_5 is defined as $F_1 \,\&\, F_3$, we conclude that $p(\neg F_1 \,\&\, F_5) = 0$. So, when we estimate the probability $p(F) = p(F_4 \vee F_5)$, we can take into consideration not only bounds for $p(F_4)$ and $p(F_5)$ (as in the traditional approach), but also the values $p(F_1) = 0.6$, $p(\neg F_1 \,\&\, F_4) = 0$, and $p(\neg F_1 \,\&\, F_5) = 0$.

For the three variables F_1, F_4, and F_5, we can form 8 atomic probabilities $p_{+++} = p(F_1 \,\&\, F_4 \,\&\, F_5)$, $p_{++-} = p(F_1 \,\&\, F_4 \,\&\, \neg F_5)$, etc. It is easy to see that, due to conditions $p(\neg F_1 \,\&\, F_4) = p_{-++} + p_{-+-} = 0$ (hence $p_{-++} = p_{-+-} = 0$) and $p(\neg F_1 \,\&\, F_5) = p_{-++} + p_{-+-} = 0$ (hence $p_{-+-} = 0$), all the terms in the remaining the expression for $p(F_4 \vee F_5) = p_{+++} + p_{++-} + p_{+-+}$ are also included in the expression for $p(F_1) = p_{+++} + p_{++-} + p_{+-+} + p_{+--} = 0.6$, hence $p(F_4 \vee F_5) \le 0.6$.

This estimate is better than the traditional estimate in which we were only able to conclude that $p(F_4 \vee F_5) \le 1$.

A detailed analysis shows (see, e.g., [1]) that if we only use probabilities for pairs of statements, we still get excess width. However, for triples, we can already get the exact probability: indeed, each intermediate statement F_j is obtained by applying a propositional operation (we will denote it by $\oplus$) to two previous statements F_k and F_l $(k, l < j)$. In this case, we get the constraint $p(F_j \,\&\, \neg(F_k \oplus F_l)) = 0$ and $p(\neg F_j \,\&\, (F_k \oplus F_l)) = 0$; these constraints are, in effect, equivalent to stating that F_j is equivalent to $F_k \oplus F_l$. Thus, LP under these constraints is equivalent to computing the exact bounds on the desired probability $p(Q)$.

Similarly, for $(A \,\&\, B) \vee (A \,\&\, C)$, the traditional method leads to excess width in comparison with $A \vee (B \,\&\, C)$; if we keep probabilities of triples of statements, we get the exact interval for $p((A \,\&\, B) \vee (A \,\&\, C))$ – i.e., we get *distributivity.*

7 Computational Complexity of the New Method and How to Make It Feasible

If we use pairs, then, on the j-th stage of the new procedure, we know the bounds on the probabilities of the previous intermediate statements $F_1, \ldots, F_{j-1}$, and

on the probabilities of the propositional combinations $F_k \oplus F_l$ for all pairs (k, l) for which $k, l < j$.

According to the above approach, to find such bounds, we must consider 2^{j-1} atomic statements $F_1^{\varepsilon_1} \,\&\, \ldots \,\&\, F_{j-1}^{\varepsilon_{j-1}}$. At the last stage, we need to consider a LP problem with exponentially many (2^n) variables; solving this problem requires exponential time.

To reduce the computation time to a feasible amount, i.e., to the time which is polynomial in the length n of the input formula F, we propose to do the following. In addition to the parameter d which describes whether we consider only bounds on probabilities $p(F_i)$ (when $d = 1$), or also bounds on pairs $p(F_j \oplus F_k)$ (when $d = 2$), or probabilities on triples (when $d = 3$), we introduce another parameter l with the following meaning.

At the j-th step, when we look for the bounds for $p(F_j)$, we have $O(j^d)$ constraints corresponding to propositional combinations of groups of d previous statements F_k. Instead of considering all these constraints, we pick l of them, and solve the problem of minimizing and maximizing the desired probabilities $p(F_j)$, $p(F_j \oplus F_k)$, etc., under the selected l constraints. As a result, we get an interval that is guaranteed to contain the range for the desired probability.

We repeat this procedure for each subset of l constraints, and take the intersection of the resulting estimates. Let us show that for every l, we get a polynomial-time algorithm. Indeed, in each estimation, we have a formula whose probability we are estimating, and l formulas coming from l known constraints (i.e., l formulas for which we have already computed the probability bounds on the previous stages). Each of the $l + 1$ propositional combinations involves $\le d$ statements F_k, so overall, they involve $m \le (l + 1) \cdot d$ expressions $F_{k_1}, \ldots, F_{k_m}$. So, the corresponding LP problem contains 2^m variables – probabilities of atomic statements $F_{i_1}^{\varepsilon_1} \,\&\, \ldots \,\&\, F_{i_m}^{\varepsilon_m}$.

What is the running time of this algorithm? A propositional function $f(x_1, \ldots, x_d)$ of d propositional variables can be described as a function from the set $\{0, 1\}^d$ of 2^d possible combinations x_i to the set $\{0, 1\}$ of possible truth values. Thus, there are exactly 2^{2^d} such functions. For fixed d and l, this means that we have $O(1)$ such functions.

One the j-th step, we have j intermediate results $F_1, \ldots, F_j$. We have $O(j^d)$ possible combinations of $\le d$ such values, so we have $O(j^d)$ probability bounds. To compute each of $O(j^d)$ new bounds, we consider all possible subsets of l probabilities. There are $O((j^d)^l) = O(j^{d \cdot l})$ such subsets. For each subset, for fixed d and l, the value m is bounded by a constant: $m = O(1)$. There are $2^m = O(1)$ possible combinations, so each LP requires $O(1)$ time. So, overall, on step j, we need $O(j^d) \cdot O(j^{d \cdot l}) \le M \cdot j^{d \cdot (l+1)}$ steps for some constant M. Overall, we need $\le M(1^{d \cdot (l+1)} + \ldots + n^{d \cdot (l+1)})$ steps, where the number n of parsing steps is bounded by the length of the formula F. It is known that $1^a + 2^a + \ldots + n^a = O(n^{a+1})$, so overall, this algorithm requires $O(n^{d \cdot (l+1)+1})$ steps. In other words, the running time grows polynomially with the length of

the formula F – so this algorithm is feasible.

It is worth mentioning that when $d \to \infty$ and $l \to \infty$, we get exact results; however, computation time grows exponentially with d and l, so we cannot realistically use too large values d and l.

8 Case Study: Computer Security

Up to now, we considered a general problem of handling interval-valued probabilistic uncertainty in expert systems. In general, as we have mentioned, the problem of computing the exact interval of possible values of probability is NP-hard, so we proposed heuristic algorithms which provide reasonable enclosures for this interval.

NP-hardness means that (unless P=NP), there is no hope of finding an efficient algorithm which would solve *all* the problems from this class. However, for many practically useful subclasses, it is often possible to design efficient algorithms. In this section, we describe a class of problems for which there is an efficient algorithm for handling interval-valued uncertainty. This class of problems is related to computer security and trust in general.

In the traditional approach to trust, we either trust an agent or not. If we trust an agent, we allow this agent full access to a particular task. For example, I trust my bank to handle my account; the bank (my agent) outsources money operations to another company (sub-agent), etc. The problem with this approach is that I may have only 99.9% trust in bank, bank in its contractor, etc. As a result, for long chains, the probability of a security leak may increase beyond any given threshold. To resolve this problem, we must keep track of trust probabilities.

Let us describe this idea in precise terms. We have a finite set A; its elements are called *agents*. For some pairs (a, b) of agents, we know that an agent a has some degree of direct trust in an agent b. We will denote the set of all such pairs by E. For each pair $(a, b) \in E$, we know the probability $p_0(a, b)$ with which a directly trusts b. We can estimate this probability, e.g., as a frequency in which a similar trust was justified.

Our objective is to describe, for given two agents f and s, the resulting probability $p_t(f, s)$ with which the agent f should trust the agent s.

Let us show how this problem can be described in precise terms. The pair (A, E), where $E \subseteq A \times A$, forms a *graph* in which agents are nodes and possible trust pairs are edges. To each edge (a, b), we associate a value $p_0(a, b) \in [0, 1]$.

Some of the trusts may be misplaced: an agent a may trust an agent b with a certain probability, but b may be misusing a's trust. Let $E' \subseteq E$ denote the (unknown) set of pairs in which the trust is justified; we will call this set the *actual trust set*.

We do not know for sure who is trustworthy and who is not, so at best, we can find some information about the probabilities $p(E')$ of different trust sets

E'. First, these probabilities must add up to 1: $\sum\limits_{E'} p(E') = 1$. Second, for every pair $(a, b) \in E$, the probability that a directly trusts b, i.e., the probability that the edge (a, b) belongs to the actual trust set E', should be equal to $p_0(a, b)$: $\sum_{E':(a,b)\in E'} p(E') = p_0(a, b)$.

Once the probability distribution $p(E')$ is fixed, we can determine the probability $p_t(f, s)$ with which f should trust s as the probability that in the actual trust set E', there is a path leading from f to s. If we denote the existence of such a path by $f \stackrel{E'}{\to} s$, then the desired probability $p_t(f, s)$ can be described as $\sum_{E':f\stackrel{E'}{\to}s} p(E')$.

We may have different probability distributions $p(E')$ which are consistent with the data $p_0(a, b)$; for different distributions, we may have different values of the trust $p_t(f, s)$. In security situations, it is desirable to find the *guaranteed* level of trust, i.e., the smallest possible value of $p_t(f, s)$ over all possible probability distributions which are consistent with the data $p_0(a, b)$. We will denote this smallest possible value by $\underline{p}_t(f, s)$; in these terms, our objective is to compute $\underline{p}_t(f, s)$.

This problem can be viewed as a particular case of the general problem of dealing with probabilities in expert systems. Indeed, here, for every agent a, we have a statement "a" meaning that f trusts a. We have a fact $f \to$ meaning that f trust himself. For each edge $(a, b) \in E$ (meaning that a trusts b), we have a rule $b \leftarrow a$ (meaning that if f trusts a, he should also trust b), which holds with probability $p_0(a, b)$. The query is "s?" – i.e., with what probability should we trust s.

Let us show that in this particular problem, we can efficiently compute the desired probability. Namely, let us define the *length* ("distrust") of an edge as $d_0(a, b) \stackrel{\text{def}}{=} 1 - p_0(a, b)$. We can naturally extend this definition to *paths*, i.e., sequences $(a_0, \ldots, a_n)$ in which $(a_i, a_{i+1}) \in E$ for all i. Namely, the *length* $\ell(\gamma)$ of a path $\gamma = (a_0, \ldots, a_n)$ is defined as usual: $\ell(\gamma) \stackrel{\text{def}}{=} \sum\limits_{i=0}^{n-1} d_0(a_i, a_{i+1})$. The length of the shortest path from f to s is defined as follows:

$$\underline{d}_t(f, s) \stackrel{\text{def}}{=} \min\{\ell(\gamma) \,|\, \gamma \text{ is a path from } f \text{ to } s\}.$$

Proposition. $\underline{p}_t(f, s) = \max(1 - \underline{d}_t(f, s), 0)$.

So, we can use Dijkstra's algorithm (see, e.g., [2]) to find the shortest path in a graph, and then compute $\underline{p}_t(f, s)$.

Proof. Let us first prove that if the probability distribution $p(E')$ is consistent with the given information, then $d_t(f, s) \le \underline{d}_t(f, s)$, where $d_t(f, s) \stackrel{\text{def}}{=} 1 - p_t(f, s)$.

Indeed, let $\gamma_0 = (a_0, a_1, \ldots, a_n)$ be the shortest path from $a_0 = f$ to $a_n = s$; then, $\underline{d}_t(f, s) = d_0(a_0, a_1) + \ldots + d_0(a_{n-1}, a_n)$.

If in the actual trust set E', there is a path from each node a_i to the next one a_{i+1}, then, combining these nodes, we will have a path from $a_0 = f$ to $a_n = s$.

Thus, if there is no path from f to s, this means that at least one of the connections (a_i, a_{i+1}) is not present in E'. Let us denote, by $N_t(f,s)$, the condition that there is no path in E' from f to s, and by $N_0(a_i, a_{i+1})$, the condition that in the actual trust set E', there is no direct connection between a_i and a_{i+1}. In terms of these notations, the above statement takes the following form:

$$N_t(f,s) \supset (N_0(a_0,a_1) \vee \ldots \vee N_0(a_{n-1},a_n)).$$

Hence, $d_t(f,s) = p(N_t(f,s) \leq p(N_0(a_0,a_1) \vee \ldots \vee N_0(a_{n-1},a_n))$. It is known that $p(A \vee B) \leq p(A) + p(B)$, and that $p(N_0(a_i,a_{i+1})) = d_0(a_i,a_{i+1})$. So, $d_t(f,s) \leq d_0(a_0,a_1) + \ldots + d_0(a_{n-1},a_n)$. Due to our choice of a_i, we know that the right-hand side of this inequality is equal to $\underline{d}_t(f,s)$. Thus, indeed $d_t(f,s) \leq \underline{d}_t(f,s)$.

To complete the proof, we produce a distribution $p(E')$ for which $p_t(f,s) \leq \max(1 - \underline{d}_t(f,s), 0)$. To describe this distribution, we start with a random variable ω which is uniformly distributed on the interval $[0,1]$. For each value $\omega \in [0,1]$, we define the set $E'(\omega) \subseteq E$ as follows. Let $\pi(x) \stackrel{\text{def}}{=} x - \lfloor x \rfloor$. We then define $E'(\omega)$ as the set of all edges $(a,b) \in E$ for which $\omega \notin \pi(I(a,b))$, where $I(a,b) \stackrel{\text{def}}{=} [\underline{d}_t(f,a), \underline{d}_t(f,a) + d_0(a,b)]$. As a result of this definition, we get different sets $E' \subseteq E$ with different probabilities $p(E')$. Since $\pi(I(a,b))$ has width $p_0(a,b)$, the distribution $p(E')$ is consistent with $p_0(a,b)$.

Induction proves (see [5] for details) that for every path starting at $a_0 = f$, if all its edges (a_i, a_{i+1}) are in $E(\omega)$, then $\omega \geq \underline{d}_t(a_0, a_n)$. Hence,

$$p_t(f,s) \leq \max(1 - \underline{d}_t(f,s), 0).$$

The statement is proven, so the above algorithm has been justified.

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCC5-209, by NSF grants EAR-0112968, EAR-0225670, and EIA-0321328, by Army Research Laboratories grant DATM-05-02-C-0046, by NIH grant 3T34GM008048-20S1, and by Applied Biomathematics.

The authors are thankful to all the participants of SCAN'04 for valuable discussions.

References

[1] S. Chopra, *Affine arithmetic-type techniques for handling unceryainty in expert systems*, University of Texas at El Paso, Department of Computer Science, Master's Thesis, 2005.

[2] Th. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 2001.

[3] L. H. de Figueiredo and J. Stolfi, *Self-Validated Numerical Methods and Applications*, Brazilian Mathematics Colloquium monograph, IMPA, Rio de Janeiro, Brazil, 1997.

[4] J. Hoefkens, M. Berz, and K. Makino, "Controlling the Wrapping Effect in the Solution of ODEs for Asteroids", *Reliable Computing*, 1999, Vol. 9, No. 1, pp. 21–41.

[5] P. Jaksurat, E. Freudenthal, M. Ceberio, and Vladik Kreinovich, "Probabilistic Approach to Trust: Ideas, Algorithms, and Simulations", *Proceedings of the Fifth International Conference on Intelligent Technologies InTech'04*, Houston, Texas, December 2–4, 2004.

[6] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics*, Springer, London, 2001.

[7] F. Lin and J. Zhao, "On tight logic programs and yet another translation from normal logic programs and propositional logic", *Proc. AAAI'03*, 2003, pp. 853–864.

[8] A. Neumaier, "Taylor Forms – Use and Limits", *Reliable Computing*, 2003, Vol. 9, No. 1, pp. 43–79.

[9] H. T. Nguyen and E. A. Walker, *First Course in Fuzzy Logic*, CRC Press, Boca Raton, Florida, 1999.

[10] S. Russell and P. Norvig, *Artificial Intelligence: a modern approach*, Prentice Hall, Englewood Cliffs, New Jersey, 2002.

[11] R. Trejo, V. Kreinovich, and C. Baral, "Towards Feasible Approach to Plan Checking Under Probabilistic Uncertainty: Interval Methods", *Proc. of the 17th National Conference on Artificial Intelligence AAAI'2000*, Austin, TX, July 30–August 3, 2000, pp. 545–550.

[12] P. Walley, *Statistical Reasoning with Imprecise Probabilities*, Chapman & Hall, N.Y., 1991.