Assignment Methods for Incidence Calculus

R.G. McLean[1], A. Bundy, and W. Liu
Department of Artificial Intelligence, University of Edinburgh, Scotland.

July 1993

Abstract.

Incidence calculus is a mechanism for probabilistic reasoning in which sets of possible worlds, called incidences, are associated with axioms and probabilities are then associated with these sets. Inference rules are used to deduce bounds on the incidences of formulae which are not axioms and bounds for the probability of such a formula can then be obtained.

In practice an assignment of probabilities directly to axioms may be given and it is then necessary to find an assignment of incidences which will reproduce these probabilities. We show that this task of assigning incidences can be viewed as a tree searching problem and two techniques for performing this search are discussed. One of these is a new proposal involving a depth first search while the other incorporates a random element. A Prolog implementation of these methods has been developed. The two approaches are compared for efficiency and the significance of their results discussed. Finally we discuss a new proposal for applying techniques from linear programming to incidence calculus.

Keywords: incidence calculus, probabilistic reasoning, incidence assignment, linear programming.

[1]Current address and all correspondence to: R.G. McLean, Department of Computer Science, University of Warwick, Coventry, CV4 7AL, England.

1 Introduction and Notation

Incidence calculus is a method of "reasoning under uncertainty" introduced in [1]. Such techniques are required for automated reasoning in many expert systems and a number of different mechanisms have been proposed.

The most common approach involves treating uncertainty as a "generalised truth value" and associating a number with each axiom of the system. The uncertainty of a compound formula is then computed as some function of the uncertainties of its subformulae. [1] refers to this as a "purely numeric mechanism".

While such a numeric mechanism allows uncertainties to be calculated in a rapid and simple way there is often no clear connection with standard probability theory. Also a purely numeric mechanism can lead to a situation where semantically equivalent formulae are assigned different uncertainties. Further details of the limitations of numeric mechanisms can be found in [1].

Incidence calculus attempts to base uncertain reasoning on classical probability theory. Instead of attaching uncertainties directly to formulae, each relevant formula has an associated incidence representing a set of possible worlds in which the formula holds. Formally the incidence is taken to be a set in some probability space. Thus an axiom A has associated with it a set $i(A)$ and a probability $p(i(A))$ is then associated with this set.

Incidence calculus therefore provides a clear connection with standard probability theory. It also ensures that semantically equivalent formulae always have the same associated probability. On the other hand programs that reason with incidence calculus need to manipulate sets and can be more complicated than programs that perform reasoning by numerical means alone.

Incidence calculus was first described in [1] and soundness and completeness results for some of its algorithms were presented in [2]. A more formal basis for the theory is presented in [3]. An introductory review of incidence calculus is presented in [4].

2 Incidence Calculus

Following [2] we shall restrict our attention to propositional logic using only connectives for negation and conjunction. There is no real loss of generality here since any propositional formula is semantically equivalent to a formula involving only these two connectives.

Let X be a set then the *propositional language* generated by X can be thought of as the smallest set $P(X)$ with the properties

1. $X \subseteq P(X)$;
2. $\sim \phi \in P(X)$ for all $\phi \in P(X)$;
3. $\phi \wedge \psi \in P(X)$ for all $\phi, \psi \in P(X)$.

The members of a propositional language will be called *formulae*.

The following is based on the definition of an incidence function given in [3].

Definition. Let W be a nonempty set and let $\mathbb{P}(W)$ be the power set of W (i.e. the set of all subsets of W) then a map $i : P(X) \to \mathbb{P}(W)$ will be called an *incidence function* if the following properties hold

$$\begin{array}{rcll} i(\sim \phi) & = & i(\phi)' & \forall \phi \in P(X) \\ i(\phi \wedge \psi) & = & i(\phi) \cap i(\psi) & \forall \phi, \psi \in P(X) \end{array}$$

Here $i(\phi)'$ denotes the complement $w\backslash i(\phi)$ of $i(\phi)$ in W. The set W is called a *set of possible worlds* and for $\phi \in P(X)$ the subset $i(\phi)$ of W is called the *incidence* of the formula ϕ.

Note that if $i : P(X) \to \mathbb{P}(W)$ is an incidence function then we can generate further incidence functions by permuting the possible worlds. If $\pi : W \to W$ is a one-to-one correspondence then the map $j : P(X) \to \mathbb{P}(W)$ defined by

$$j(\phi) = \{\pi(w) | w \in i(\phi)\} \qquad \forall \phi \in P(X)$$

is an incidence function.

We now discuss the incidence calculus inference mechanism.

Let $\mathcal{A}$ be a set of formulae from $P(X)$, let W be a nonempty set of possible worlds and let i be a map from $\mathcal{A}$ to $\mathbb{P}(W)$. We regard $\mathcal{A}$ as the set of axioms for an incidence calculus theory and i as an assignment of incidences set up by the user of a program based on such a theory. To estimate the incidence of a formula which is not an axiom we begin by setting up a lower bound $\emptyset$ and an upper bound W. The inference mechanism then refines these bounds using the known incidences of the axioms. The bounds for incidences can be described by a pair of maps (L, U) where $L(\psi)$ is a lower bound for the incidence of ψ and $U(\psi)$ an upper bound.

Definition. An *assignment* is a pair (L, U) of maps from some finite subset of a propositional language to a power set.

An incidence calculus *rule* takes an assignment (L_F, U_F) and produces a new assignment (L_G, U_G) by altering the value of either L_F or U_F on at most one formula. A rule can be specified by stating which of the maps is altered, the formula whose assignment is changed and the value of the new assignment on that formula.

The following rules are given by [1].

$$\begin{aligned}
\text{Not1}: \quad U_G(\alpha) &= L_F(\sim\alpha)' \cap U_F(\alpha) \\
\text{Not2}: \quad L_G(\alpha) &= U_F(\sim\alpha)' \cup L_F(\alpha) \\
\text{Not3}: \quad U_G(\sim\alpha) &= L_F(\alpha)' \cap U_F(\sim\alpha) \\
\text{Not4}: \quad L_G(\sim\alpha) &= U_F(\alpha)' \cup L_F(\sim\alpha) \\
\text{And1}: \quad U_G(\alpha) &= (U_F(\alpha \wedge \beta) \cup L_F(\beta)') \cap U_F(\alpha) \\
\text{And2}: \quad L_G(\alpha) &= L_F(\alpha \wedge \beta) \cup L_F(\alpha) \\
\text{And3}: \quad U_G(\beta) &= (U_F(\alpha \wedge \beta) \cup L_F(\alpha)') \cap U_F(\beta) \\
\text{And4}: \quad L_G(\beta) &= L_F(\alpha \wedge \beta) \cup L_F(\beta) \\
\text{And5}: \quad U_G(\alpha \wedge \beta) &= U_F(\alpha) \cap U_F(\beta) \cap U_F(\alpha \wedge \beta) \\
\text{And6}: \quad L_G(\alpha \wedge \beta) &= (L_F(\alpha) \cap L_F(\beta)) \cup L_F(\alpha \wedge \beta).
\end{aligned}$$

Additional rules for disjunction and implication have been obtained in [5].

For each of the above the *contributions* of a rule are all the expressions $L_F(\phi)$ and $U_F(\psi)$ appearing on the right hand side with the exception of the last such expression.

We now summarise the inference mechanism. Further details will be found in [1].

The Incidence Calculus Inference Mechanism. Let $\mathcal{D}$ be a finite subset of a propositional language $P(X)$ and let (L_1, U_1) be an assignment where L_1 and U_1 are maps from $\mathcal{D} \to I\!P(W)$ for some set W of possible worlds. Suppose also we have an initial queue of rules i.e. a finite list whose members are inference rules of the form listed above. Construct new assignments $(L_2, U_2), (L_3, U_3), \ldots$ as follows: suppose that (L_n, U_n) has been defined then

1. if there is a $\phi \in \mathcal{D}$ with $L_n(\phi) \not\subseteq U_n(\phi)$ then terminate with (L_n, U_n);
2. if the queue is empty then terminate with (L_n, U_n);
3. otherwise remove the first rule from the queue and use it to update (L_n, U_n) to the new assignment (L_{n+1}, U_{n+1}). If the update leads to $L_{n+1}(\phi) \neq L_n(\phi)$ then add all rules having $L_n(\phi)$ as a contribution to the queue. If the update leads to $U_{n+1}(\phi) \neq U_n(\phi)$ then add all rules having $U_n(\phi)$ as a contribution to the queue.

It can be shown that the updating of assignments by the inference mechanism will eventually terminate [2].

The inference mechanism plays a fundamental role in Bundy's inconsistency detection algorithm and legal assignment finder algorithm. Given a map $i : \mathcal{A} \to I\!P(W)$ representing an assignment of incidences to axioms the inference mechanism can be applied to the initial assignment (L_0, U_0) defined by

$$L_0(\phi) = i(\phi) = U_0(\phi) \qquad \forall \phi \in \mathcal{A}$$

$$L_0(\phi) = \emptyset, \quad U_0(\phi) = W \qquad \forall \phi \in \mathrm{sf}(\mathcal{A}) \backslash W$$

to produce a final assignment (L_N, U_N). Here $\mathrm{sf}(\mathcal{A})$ denotes the set of all formulae which are subformulae of some member of $\mathcal{A}$.

We say that a map $i : \mathcal{A} \to I\!P(W)$ is *consistent* if it can be extended to an incidence function $i : P(X) \to I\!P(W)$. There are three possible outcomes from the inference mechanism:

1. $L_N(\phi) = U_N(\phi)$ for all $\phi \in \mathrm{sf}(\mathcal{A})$. In this case i is consistent.
2. $L_N(\phi) \supset U_N(\phi)$ for some $\phi \in \mathrm{sf}(\mathcal{A})$. In this case i is not consistent.
3. $L_N(\phi) \subseteq U_N(\phi)$ for all $\phi \in \mathrm{sf}(\mathcal{A})$ where the inclusion is strict for at least one ϕ. In this case i may or may not be consistent.

In case 3 the consistency or inconsistency of i can be determined by applying Bundy's legal assignment finder algorithm [1]. In fact this algorithm will determine all incidence functions which are extensions of i. It does this by splitting an assignment satisfying condition 3 into two new assignments and applying the inference mechanism to each. The split is made by choosing a formula ψ for which $L_N(\psi) \subset U_N(\psi)$ and choosing a $w \in U_N(\psi) \backslash L_N(\psi)$. The first new assignemnt is obtained by adding w to $L_N(\psi)$ and the second by removing w from $U_N(\psi)$.

3 The Problem of Incidence Assignment

In an expert system it is usual to initially assign a number representing an uncertainty (e.g. a probability) to each axiom. If such a system is to base it's reasoning on incidence calculus then it is necessary to assign incidences to the axioms and also to find a probability measure which can reproduce the original given numerical uncertainties.

We now discuss some methods for carrying out this incidence assignment procedure. These methods assume that the incidences will be events in some equiprobable probability space. (In a equiprobable space $(W, \mathbb{P}(W), p)$ the probability $p(E)$ of a subset E of a finite set W of possible worlds is given by $p(E) = |E|/|W|$ where $|S|$ denotes the number of elements in the set S.) The number of possible worlds required can then be estimated from the given uncertainties — if each uncertainty is required to n decimal places then we may use 10^n possible worlds.

Statement of the Problem. We are given a subset $\mathcal{A}$ of a propositional language $P(X)$, a map $u : \mathcal{A} \to [0, 1]$, and a probability space $(W, \mathbb{P}(W), p)$. We wish to find a consistent map $i : \mathcal{A} \to \mathbb{P}(W)$ (representing an assignment of incidences) with

$$u(\phi) = p(i(\phi)) \qquad \forall \phi \in \mathcal{A}.$$

The members of $\mathcal{A}$ are the *axioms* of the incidence calculus theory.

When both $\mathcal{A}$ and W are finite there is a straightforward but inefficient way of searching for a suitable incidence function. There are only finitely many maps $i : \mathcal{A} \to \mathbb{P}(W)$ with $u(\phi) = p(i(\phi))$ for all $\phi \in \mathcal{A}$ so we need only search for such maps and apply the legal assignment finder to each until a consistent i is found. If no such i is found then there is no assignment of incidents which can reproduce the given uncertainties.

A more efficient procedure consists of choosing a suitable value of $i(\psi)$ for some $\psi \in \mathcal{A}$ and then using the inference mechanism to calculate bounds on $i(\phi)$ for $\phi \neq \psi$. To obtain values of i on other formulae we need then only search for suitable sets lying between these bounds.

4 A Tree for Incidence Assignment

The assignment methods to be discussed in this paper can be viewed as techniques for searching a tree. We begin by defining this tree.

Incidence Assignment Tree. Given a subset $\mathcal{A} = \{\psi_1, \ldots \psi_n\}$ of a propositional language, a probability space $(W, \mathbb{P}(W), p)$ and a map $u : \mathcal{A} \to [0, 1]$, we define a tree whose vertices are assignments. The root is the assignment (L_0, U_0) given by

$$L_0(\phi) = \emptyset, \quad U_0(\phi) = W \qquad \forall \phi \in \mathrm{sf}(\mathcal{A}).$$

Let (L, U) be a vertex at level r, then for each $E \in \mathbb{P}(W)$ with

$$L(\psi_r) \subseteq E \subseteq U(\psi_r) \qquad \text{and} \qquad u(\psi_r) = p(E)$$

there is a subtree of (L,U) whose root (L_r,U_r) is obtained by applying the incidence calculus inference mechanism to the assignment (L_q,U_q) defined by

$$\begin{array}{rcll} L_q(\phi) & = & L(\phi) & \forall \phi \in \mathrm{sf}(\mathcal{A})\backslash\{\psi_r\} \\ U_q(\phi) & = & U(\phi) & \forall \phi \in \mathrm{sf}(\mathcal{A})\backslash\{\psi_r\} \\ L_q(\psi_r) & = & E & \\ U_q(\psi_r) & = & E & \end{array}$$

with the initial queue of rules containing all inference rules having $L_F(\psi_r)$ or $U_F(\psi_r)$ in their contribution.

In practice it is more efficient to use a modification of the inference mechanism where an extra check is made each time a rule fires. This will cause the inference mechanism to terminate if the uncertainty of some formula does not lie between the bounds calculated from the current assignment. We now state this updated form of the inference mechanism — it is identical to the previous one except for the addition of condition 0.

The Incidence Calculus Inference Mechanism (Version 2). Let $\mathcal{D}$ be a finite subset of a propositional language $P(X)$ let W be a set and let L_1,U_1 be maps from $\mathcal{D}$ to $\mathbb{P}(W)$. Suppose also that there is a map u from some subset of $\mathcal{D}$ to $[0,1]$. Define new assignments $(L_2,U_2),(L_3,U_3),\ldots$ as follows: suppose that (L_n,U_n) has been defined then

0. if there is a $\phi \in \mathcal{D}$ for which $u(\phi)$ is defined and $u(\phi) < p(L(\phi))$ or $u(\phi) > p(U(\phi))$ then terminate with (L_n,U_n) and failure;

1. if there is a $\phi \in \mathcal{D}$ with $L_n(\phi) \not\subseteq U_n(\phi)$ then terminate with (L_n,U_n) and failure;

2. if the queue is empty then terminate with (L_n,U_n);

3. otherwise remove the first rule from the queue and use it to update (L_n,U_n) to the new assignment (L_{n+1},U_{n+1}). If the update leads to $L_{n+1}(\phi) \neq L_n(\phi)$ then add all rules having $L_n(\phi)$ as a contribution to the queue. If the update leads to $U_{n+1}(\phi) \neq U_n(\phi)$ then add all rules having $U_n(\phi)$ as a contribution to the queue.

It is easy to see that a vertex (L,U) of the incidence assignment tree is a leaf if and only if at least one of the following conditions holds:

$\mathcal{L}1$. $L(\phi) = U(\phi)$ for all $\phi \in \mathrm{sf}(\mathcal{A})$;

$\mathcal{L}2$. $L(\phi) = U(\phi)$ for all $\phi \in \mathcal{A}$ and $L(\psi) \subset U(\psi)$ for some $\psi \in \mathrm{sf}(\mathcal{A})$;

$\mathcal{L}3$. $L(\phi) \not\subseteq U(\phi)$ for some $\phi \in \mathrm{sf}(\mathcal{A})$

$\mathcal{L}4$. $u(\phi) < p(L(\phi))$ or $u(\phi) > p(U(\phi))$ for some $\phi \in \mathrm{sf}(\mathcal{A})$

$\mathcal{L}5$. there are no $\phi \in \mathcal{A}$ and $E \in \mathbb{P}(W)$ with $L(\phi) \subseteq E \subseteq U(\phi)$ and $u(\phi) = p(E)$.

Here cases $\mathcal{L}3$, $\mathcal{L}4$ and $\mathcal{L}5$ all represent failure. To find an assignment of incidences we search the tree for a vertex (L,U) satisfying one of the conditions $\mathcal{L}1$ or $\mathcal{L}2$. A map $i : \mathcal{A} \to \mathbb{P}(W)$ can then be defined by

$$i(\phi) = L(\phi) \qquad \forall \phi \in \mathcal{A}.$$

If (L,U) satisfies $\mathcal{L}1$ then i is consistent. If (L,U) satisfies $\mathcal{L}2$ then the legal assignment finder can be applied to determine if i is consistent or not. In the case of an inconsistent i we continue searching for a new leaf satisfying $\mathcal{L}1$ or $\mathcal{L}2$ and repeat the procedure.

Once a consistent leaf (L,U) has been found then for $\phi \in \mathrm{sf}(\mathcal{A})\backslash\mathcal{A}$ we can estimate that the uncertainty of ϕ lies in the interval

$$[p(L(\phi)), p(U(\phi))]\,.$$

To estimate the uncertainty of a formula $\psi \notin \mathrm{sf}(\mathcal{A})$ the inference mechanism can be run using the initial assignment (L_1,U_1) given by

$$\begin{array}{lcll} L_1(\phi) & = & L(\phi) & \forall\phi \in \mathrm{sf}(\mathcal{A}) \\ L_1(\phi) & = & \emptyset & \forall\phi \in \mathrm{sf}(\psi)\backslash\mathrm{sf}(\mathcal{A}) \\ U_1(\phi) & = & U(\phi) & \forall\phi \in \mathrm{sf}(\mathcal{A}) \\ U_1(\phi) & = & W & \forall\phi \in \mathrm{sf}(\psi)\backslash\mathrm{sf}(\mathcal{A}) \end{array}$$

to produce an assignment (L_2,U_2). Then the uncertainty of ψ can be estimated to lie in the interval

$$[p(L_2(\phi)), p(U_2(\phi))]\,.$$

5 Incidence Assignment by Depth First Search

Depth First Incidence Assignment Method. Here a depth first search is performed on the incidence assignment tree until either a consistent assignment is found or all leafs have been checked without a suitable assignment being discovered.

This assignment mechanism has been implemented in Prolog where the depth first search can be carried out using Prolog's backtracking mechanism.

Example. Let $(W,\mathbb{P}(W),p)$ be an equiprobable probability space where $W = \{0,1,2,3,4,5,6,7,8,9\}$ and define $u : \{x,y\} \to [0,1]$ by

$$\begin{array}{lcl} u(x) & = & 0.8 \\ u(y) & = & 0.2. \end{array}$$

Then there are $10!/(8!2!) = 45$ possible choices for $i(x)$ with $p(i(x)) = u(x)$. Similarly there are 45 possible choices for $i(y)$ with $p(i(y)) = u(y)$. Thus there are 2025 incidence functions compatible with the given uncertainties.

Each of these incidence functions i is of the form

$$i(\phi) = \pi\left(i_r(\phi)\right) \qquad \forall\phi \in P(\{x,y\})$$

for some $r \in \{1,2,3\}$ and some permutation π of $W = \{0,1,2,3,4,5,6,7,8,9\}$, where i_1, i_2 and i_3 are the incidence functions with

$$\begin{array}{lcllcl} i_1(x) & = & \{0,1,2,3,4,5,6,7\}, & i_1(y) & = & \{8,9\}, \\ i_2(x) & = & \{0,1,2,3,4,5,6,7\}, & i_2(y) & = & \{7,8\} \\ i_3(x) & = & \{0,1,2,3,4,5,6,7\}, & i_3(y) & = & \{6,7\}. \end{array}$$

Thus there are essentially only three independent incidence functions from which all the others can be obtained.

Each of these three gives a different estimate for the uncertainty of $x \wedge y$:

$$\begin{aligned} p(i_1(x \wedge y)) &= 0 \\ p(i_2(x \wedge y)) &= 0.1 \\ p(i_3(x \wedge y)) &= 0.2. \end{aligned}$$

If a depth first search of the assignment tree terminates with the discovery of the incidence function i_1 then the corresponding estimate of the uncertainty of $x \wedge y$ will be 0. However we know that other possible values of this uncertainty are 0 and 0.1. Thus a search may find an assignment which is an unrepresentative sample of the set of all possible assignments.

It is easy to calculate that here there are 45 possible incidence functions give rise to an uncertainty of 0 for $x \wedge y$, 720 incidence functions giving an uncertainty of 0.1 for $x \wedge y$ and 1260 incidence functions giving an uncertainty of 0.2 for $x \wedge y$. So a depth first search is more likely to find an incidence function i for which $i(x \wedge y)$ has two elements.

The above example illustrates a number of important points about incidence assignment methods:

1. It is possible to have two leafs (L_1, U_1) and (L_2, U_2) of the assignment tree each of which is compatible with the given uncertainties but for which the intervals

   $$[p(L_1(\psi)), p(U_1(\psi))] \qquad \text{and} \qquad [p(L_2(\psi)), p(U_2(\psi))]$$

   are disjoint. Thus calculating uncertainties from a single leaf will in general give estimated bounds rather than exact ones.

2. Using a depth first incidence assignment method to search for all possible assignments which are compatible with some given uncertainties may be very inefficient since time may be wasted discovering a large number of assignments which are permutations of a few basic ones.

3. A search for a single consistent leaf may terminate with an unrepresentative sample which will then lead to poor estimates of uncertainties.

6 A Monte Carlo Method for Incidence Assignment

Some proposals for dealing with the above problems are suggested in [6]. The simplest of these consists of searching for a number of suitable assignments and then taking the average of the uncertainties calculated from each assignment. Another proposal of Corlett and Todd is the "Monte Carlo" method in which the search of the incidence assignment tree is carried out using a suitable random choice of subtree at each vertex. This choice is based on the following observation:

> Given an equiprobable probability space $(\Omega, \mathbb{P}(\Omega), \mu)$ where $\Omega = \{\omega_1, \ldots \omega_N\}$ and a number $v \in [0, 1]$ then performing a sequence of N Bernoulli trials each with probability v of success is likely to produce vN successes. Therefore the subset A of Ω defined by
>
> $$\omega_r \in A \qquad \Longleftrightarrow \qquad \text{the } r\text{'th trial is a success}$$
>
> is likely to satisfy $\mu(A) = v$.

The following algorithm for this Monte Carlo approach is described in [4].

Monte Carlo Incidence Assignment Method. Let $\mathcal{A} = \{\psi_1, \ldots \psi_n\}$ be a subset of a propositional language, let $(W, \mathbb{P}(W), p)$ be an equiprobable probability space and let u be a map from $\mathcal{A}$ to $[0,1]$. The corresponding incidence assignment tree is searched as follows. Let the current assignment be (L,U) at level r. Let $U(\psi_r)\backslash L(\psi_r) = \{w_1, \ldots w_N\}$ and perform a sequence of N Bernoulli trials each with probability

$$\frac{u(\psi_r) - p(L(\psi_r))}{p(U(\psi_r)\backslash L(\psi_r))}$$

of success. Let $b_1, \ldots, b_N$ be the sequence of outcomes (where 1 represents success and 0 represents failure). Define

$$A = \{w_k | b_k = 1, 1 \leq k \leq N\}.$$

Repeat this procedure until a set A is obtained which satisfies

$$p(A) = u(\psi_r) - p(L(\psi_r))$$

then take $E = L(\psi_r) \cup A$ so that

$$p(E) = u(\psi_r).$$

Now search the subtree of (L,U) determined by E.

This assignment method also suffers from the three problems discussed at the end of section 5.

A Prolog implementation of the Monte Carlo Incidence Assignment Method has been developed and is described in [5]. Bernoulli trials are produced using the pseudo-random number generator defined in section 7.8.1 of [7].

7 Examples of Incidence Assignment

We now illustrate the two assignment methods discussed above with some examples. These are based on a table given in [8]. To simplify the discussion here we have rounded off the values to one decimal place.

Suppose we are given a map u defined on a subset of a propositional language by:

$$\begin{aligned} u(a \wedge b \wedge c) &= 0.2 \\ u(a \wedge c) &= 0.4 \\ u(b \wedge c) &= 0.2 \\ u(a) &= 0.8 \\ u(b) &= 0.6 \\ u(c) &= 0.5. \end{aligned}$$

We shall first consider a depth first search using an equiprobable space with ten possible worlds i.e. take $W = \{0,1,2,3,4,5,6,7,8,9\}$ and define $p(S) = |S|/10$ for each subset S

of W. We seek an incidence function $i : P(\{a,b,c\}) \to \mathbb{P}(W)$ with

$$\begin{aligned} |i(a \wedge b \wedge c)| &= 2 \\ |i(a \wedge c)| &= 4 \\ |i(b \wedge c)| &= 2 \\ |i(a)| &= 8 \\ |i(b)| &= 6 \\ |i(c)| &= 5. \end{aligned}$$

The initial assignment (L_0, U_0) is given by

$$L_0(\phi) = \emptyset, \qquad U_0(\phi) = W \qquad \forall \phi \in \{a \wedge b \wedge c, a \wedge c, b \wedge c, a, b, c\}.$$

We attempt to assign incidences to the axioms in the order

$$a \wedge b \wedge c, \quad a \wedge c, \quad b \wedge c, \quad a, \quad b, \quad c.$$

A test run of the Prolog implementation of the depth first assignment method on this example yields the following incidences which are compatible with the given uncertainties:

$$\begin{aligned} i(a \wedge b \wedge c) &= \{0,1\} \\ i(a \wedge c) &= \{0,1,2,3\} \\ i(b \wedge c) &= \{0,1\} \\ i(a) &= \{0,1,2,3,4,5,6,7\} \\ i(b) &= \{0,1,4,5,6,7\} \\ i(c) &= \{0,1,2,3,8\}. \end{aligned}$$

This assignment can now be used to calculate bounds on the uncertainty of an arbitrary formula. For example to find the uncertainty of $(a \vee d) \to c$ we add the new formula and any unassigned subformulae to the current assignment giving them the default values $\emptyset$ and W. When the inference mechanism is run on this it produces the new bounds:

$$\begin{aligned} L((a \vee d) \to c) &= \{0,1,2,3,8\} \\ U((a \vee d) \to c) &= \{0,1,2,3,8,9\} \\ L(a \vee d) &= \{0,1,2,3,4,5,6,7\} \\ U(a \vee d) &= W \\ L(d) &= \emptyset \\ U(d) &= W \end{aligned}$$

so we conclude that the uncertainty of $(a \vee d) \to c$ lies between the probabilities of $\{0,1,2,3,8\}$ and $\{0,1,2,3,8,9\}$ i.e. between 0.5 and 0.6.

In general we may not have an uncertainty assigned to every subformula of the set of axioms. In this case a situation may arise where every axiom has been assigned a suitable incidence, the inference mechanism has been run and we still have $L(\phi) \subset U(\phi)$ for some ϕ. In this case it is necessary to apply the legal assignment finder to determine if the current assignment is consistent.

We now consider the effect of assigning incidences when the axioms are given in a different order. This will illustrate how the algorithm backtracks if it makes an unsuitable choice of incidence and also show that the order plays an important role in the assignment process.

Suppose we assign incidences to the axioms in the order

$$a, b, c, a \wedge c, b \wedge c, a \wedge b \wedge c.$$

After an initial choice assigning a set with eight elements to a, running the inference mechanism will still leave b with the default bounds. Then an arbitrary set with six elements is assigned to b, and an arbitrary set with five elements to c. Since these assignments are made independently it is unlikely that the intersection of the incidents will contain suitable numbers of elements to reproduce the uncertainties of $a \wedge c, b \wedge c$ and $a \wedge b \wedge c$. In this case the algorithm will backtrack and work through the various possible assignments to a, b and c until a suitable one is found.

In a test run of the Prolog implementation of the depth first assignment method backtracking occurred 52 times before a consistent assignment was found. Thus the order in which the axioms are arranged can play a crucial role in determining how efficiently an assignment is found.

We now describe a test run of the Monte Carlo assignment program on the above example. Here the initial choice of an incidence for $a \wedge b \wedge c$ is $\{0, 9\}$. After the inference mechanism is run on this the program then attempts to assign an incidence to $b \wedge c$ which has associated uncertainty 0.4. The first random subset generated is $\{0, 9, 1, 2, 7\}$ which is rejected as two large. Then $\{0, 9, 3\}$ is generated and rejected as two small. On the next attempt a suitable incidence $\{0, 9, 3, 6\}$ is generated. The inference mechanism is then run and the program then moves on to search for an incidence for the next axiom.

When the axioms were given in the order

$$a \wedge b \wedge c, a \wedge c, b \wedge c, a, b, c.$$

the Monte Carlo method returned the following assignment of incidences:

$$\begin{aligned} i(a \wedge b \wedge c) &= \{0, 9\} \\ i(a \wedge c) &= \{0, 3, 6, 9\} \\ i(b \wedge c) &= \{0, 9\} \\ i(a) &= \{0, 2, 3, 5, 6, 7, 8, 9\} \\ i(b) &= \{0, 1, 2, 7, 8, 9\} \\ i(c) &= \{0, 3, 4, 6, 9\}. \end{aligned}$$

When the axioms were presented in the order

$$a, b, c, a \wedge c, b \wedge c, a \wedge b \wedge c.$$

the Monte Carlo program made a bad initial choice of incidences. This caused the search to enter a part of the assignment tree which contained no consistent assignment. This was confirmed by aborting the Monte Carlo method and using a depth first search to confirm that no consistent leafs existed under the current vertex.

A more realistic example where u is given by

$$\begin{aligned} u(a \wedge b \wedge c) &= 0.18 \\ u(a \wedge b) &= 0.52 \\ u(a \wedge c) &= 0.35 \\ u(b \wedge c) &= 0.22 \\ u(a) &= 0.760 \\ u(b) &= 0.640 \\ u(c) &= 0.480 \end{aligned}$$

and 100 possible worlds are used was tested using a SICStus Prolog interpreter on a Hewlett-Packard workstation. Here consistent assignments were found in a user time of 374.520 for a depth first search and 355.060 using the Monte Carlo method.

These examples show that the efficiency of both the depth first and Monte Carlo methods are affected by the order in which the axioms are considered when assigning incidences. A further difficulty of the Monte Carlo method is that the current implementation does not backtrack so that the program can fail to find an assignment even when one exists.

8 Incidence Calculus and Linear Programming

In this section chapter we discuss an alternative method for calculating uncertainties in incidence calculus. Previously we considered techniques which started with a given probability space and searched for an incidence assignment. We now reverse this approach, starting with an incidence assignment and then searching for suitable probabilities.

A map $v : P(X) \to \{0, 1\}$ will be called a *valuation* if it satisfies

1. $v(\sim \phi) = 1 - v(\phi) \quad \forall \phi \in P(X)$;
2. $v(\phi \wedge \psi) = \min\{v(\phi), v(\psi)\} \quad \forall \phi, \psi \in P(X)$.

Let V be the set of all valuations on $P(X)$ then it is easy to verify that the map $\iota : P(X) \to V$ defined by

$$\iota(\phi) = \{v \in V | v(\phi) = 1\} \qquad \forall \phi \in P(X)$$

is an incidence function. We shall call ι the *natural incidence function* on $P(X)$.

The natural incidence function will be used to give our assignment of incidences to formulae. In practice when X is large this choice may produce an unfeasibly large number of possible worlds since V contains $2^{|X|}$ elements. However we can always choose a fixed subset V_0 of V containing say 100 elements and then define an incidence function $j : P(X) \to I\!P(V_0)$ by

$$j(\phi) = \iota(\phi) \cap V_0.$$

We now wish to associate a probability $p(E)$ with each subset E of V or equivalently to find a probability density function $f : V \to [0, 1]$ (i.e. a map satisfying $f(v) \geq 0$ for all $v \in V$ and $\sum_{v \in V} f(v) = 1$) from which p can be calculated using the relation

$$p(E) = \sum_{v \in E} f(v) \qquad \forall E \in I\!P(V).$$

Our problem can be summarised as follows.

Problem *Let V be the set of valuations of a propositional language $P(X)$. Given a map $u : \{\phi_1, \ldots \phi_m\} \to [0, 1]$ defined on a subset of $P(X)$ determine the probability densities $f : V \to [0, 1]$ which satisfy*

$$u(\phi_j) = \sum \{f(v) | v \in \iota(\phi_j)\} \qquad (1 \leq j \leq m) \qquad (*)$$

where $\iota : P(X) \to I\!P(V)$ is the natural incidence function defined by

$$\iota(\phi) = \{v \in V | v(\phi) = 1\}.$$

Let $V = \{v_1, \ldots v_n\}$. Then we can write $(*)$ as

$$\sum_{k=1}^{n} b_{jk} f(v_k) = u(\phi_j) \qquad (1 \leq j \leq m)$$

where

$$b_{jk} = \begin{cases} 1 & \text{if } v_k \in \iota(\phi_j) \\ 0 & \text{if } v_k \notin \iota(\phi_j). \end{cases}$$

Thus finding f is equivalent to determining if there are numbers $f(v_1), \ldots, f(v_n)$ which satisfy the equations

$$\begin{array}{ccccccc} b_{11}f(v_1) & + & \ldots\ldots & + & b_{1n}f(v_1) & = & u(\phi_1) \\ \vdots & & & & \vdots & \vdots & \vdots \\ b_{m1}f(v_1) & + & \ldots\ldots & + & b_{mn}f(v_n) & = & u(\phi_n) \end{array}$$

and which also define a density function i.e. which also satisfy

$$f(v_1) \quad + \quad \ldots\ldots \quad + \quad f(v_n) \quad = \quad 1$$

$$f(v_j) \geq 0 \qquad (1 \leq j \leq n).$$

So we wish to determine solutions of the matrix equation

$$\begin{bmatrix} b_{11} & b_{12} & \ldots & b_{1n} \\ \vdots & \vdots & & \vdots \\ b_{m1} & b_{m2} & \ldots & b_{mn} \\ 1 & 1 & \ldots & 1 \end{bmatrix} \begin{bmatrix} f(v_1) \\ \vdots \\ f(v_n) \end{bmatrix} = \begin{bmatrix} u(\phi_1) \\ \vdots \\ u(\phi_n) \end{bmatrix}$$

for which $f(v_j) \geq 0$ for each j.

A theoretical solution to this problem is easily obtained using standard techniques of linear programming. The general solution set for a matrix equation of this form may be described as follows (for details see section 3 of chapter 2 of [9] or chapter 14 of [10]):

Let z be a column vector with

$$z = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}$$

and suppose that the entries of the $m \times n$ matrix B and $n \times 1$ column vector u are known. If the matrix equation $Bz = u$ has solutions for z satisfying $z_j \geq 0$ for all j and if there is a constant $K > 0$ with

$$z_1 + \ldots + z_n \leq K$$

for every such solution then there are finitely many particular solutions

$$z = e_1, z = e_2, \ldots, z = e_r$$

for which

$$\{z | Bz = u, z_j \geq 0 \ \forall j\} = \left\{ \sum_{j=1}^{r} \lambda_j e_j \,\middle|\, \lambda_1, \ldots \lambda_r \in [0,1] \quad \textit{and} \quad \sum_{j=i}^{r} \lambda_j = 1 \right\}.$$

Moreover there is a unique smallest set $\{e_1, \ldots, e_r\}$ *with this property — its members are called the basic feasible solutions of the equation.*

Thus although the matrix equation we are considering will in general have infinitely many solutions these are completely determined one we know the finite set of basic feasible solutions.

Note. The theorem can be applied to the problem we are discussing by taking $K = 1$ for,since we are searching for density functions, one of our equations will always have the form $z_1 + \ldots + z_n = 1$.

Linear programming provides algorithms for determining the basic feasible solutions of such a matrix equation. However there are problems with such methods, for example:

> ... the number of basic feasible solutions quickly increases as the number of variables increases. For example, a linear programming problem with forty variables and twenty equality constraints could have over 130 billion basic feasible solutions [10, p.195].

However for the purposes of performing calculations in incidence calculus it is probably unnecessary to determine all basic feasible solutions. It may be sufficient to use some random procedure to find a single solution. Alternatively we could attempt to find the "centre of gravity" of the solution set

$$\frac{1}{r}\sum_{k=1}^{r} e_r$$

which could be regarded as giving the average probability density.

We have described a theoretical background for incidence assignment using linear programming methods. The main problems with such an approach lie in the large number of possible worlds that are introduced and in the computational complexity of calculating the probabilities associated with these worlds. Further work is needed to determine if suitable approximation methods can make this approach feasible.

9 Conclusions

We have discussed two methods for assigning incidences to axioms. In each case an incidence with the required probability is selected and then the inference mechanism is used to obtain the bounds on the incidences of other formulae which result from this choice.

Two assignment techniques have been examined. The first performs a depth first search looking for an assignment. The efficiency of this search depends crucially on the order in which the axioms are considered when incidences are being selected. Further work needs to be done in this area to determine the most suitable order. It appears from the examples considered here that it is best to assign incidences first to compound formulae or formulae with small uncertainties.

The advantage of a depth first search is that it will eventually terminate and determine if a consistent assignment is possible (in practice however this may take a very long time). It is also impractical to use this method for determining all possible incidence assignments

as we saw in Section 5 that in general there can be many assignments which are obtained by applying permutations to a much smaller number of ones.

Another disadvantage of the depth first procedure is that it may discover an assignment which gives poor estimates of uncertainties for formulae which are not axioms (for example in section 5 the assignments i_1 and i_3 give extreme values for the uncertainty of $x \wedge y$ while i_2 gives a preferable estimate). An alternative procedure, the Monte Carlo assignment method attempts to resolve this difficulty by making random selections of incidences. It is therefore likely to return an average incidence assignment which should allow good estimates to be made for other uncertainties.

However the current implementation of the Monte Carlo method is unable to detect a situation in which no assignment is possible. In such a case it may continue indefinitely making random choices of incidences.

It is also possible for the Monte Carlo method to make a bad initial choice of incidences leading to a situation where it cannot proceed to find an assignment. A possible improvement here would consist of setting a limit (say 100) to the number of random sets that can be selected at a given vertex in the search tree. After 100 attempts if no suitable incidence has been discovered the algorithm would backtrack and try a different assignment for the previous axiom

We have outlined an alternative approach to calculating uncertainties in incidence calculus which avoids the need to search for an assignment of incidences. At present the method has not been tested in practice against the incidence calculus programs. There are many programs available for solving linear programming problems (the facility is available in some computer algebra systems) so some experiments in this area could easily be carried out in future.

Acknowledgements

RGM wishes to thank SERC for financial support and Flavio Corrêa da Silva for useful discussions on incidence calculus. AB was supported by an SERC Senior Fellowship. WL was supported by a "Colin and Ethel Gordon Scholarship" of Edinburgh University and an ORS award.

References

[1] Bundy, A., Incidence calculus: a mechanism for probabilistic reasoning, *J. Automated Reasoning*, **1** (1985) 263-283.

[2] Bundy, A., Correctness criteria of some algorithms for uncertain reasoning using incidence calculus, *J. Automated Reasoning*, **2** (1986) 109-126.

[3] Corrêa da Silva, F.S. and Bundy, A., A rational reconstruction of incidence calculus, Research Paper 517, Department of Artificial Intelligence, University of Edinburgh (1991).

[4] Bundy, A., Incidence calculus. Research Paper 497, Dept. of Artificial Intelligence, University of Edinburgh, 1990, (to appear in the Encyclopedia of Artificial Intelligence).

[5] McLean, R.G., Testing and Extending the Incidence Calculus, M.Sc. Dissertation, Department of Artificial Intelligence, University of Edinburgh (1992).

[6] Corlett, R.A. and Todd, S.J., A Monte Carlo Approach to Uncertain Inference, in *Artificial Intelligence and its Applications*, A.G. Cohn and J.R. Thomas (eds.), John Wiley (1986) pp.127-137.

[7] Clocksin, W.F. and Mellish, C.S., *Programming in Prolog*, Springer-Verlag, Berlin, Heidelberg, New York (1987).

[8] Kyburg, H.E. Jr., Evidential Probability, Computer Science Technical Report number 376, University of Rochester (1991).

[9] Jeter, M.W., *Mathematical Programming, an introduction to optimization*, Marcel Dekker, New York and Basel (1986).

[10] Rorres C. and Anton, H., *Applications of Linear Algebra*, Wiley, New York (1977).