Computationally Tractable, Conceptually Plausible Classes of Link Matrices for the Inquery Inference Network

Warren R. Greiff
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

September, 1996

Abstract

This paper presents the PIC matrices, a computationally efficient subclass of link matrices that may be considered for the interpretation of query operators in the INQUERY inference network. The PIC class of matrices is formally defined; a $O(n^2)$ algorithm, PIC-EVAL, is presented for the evaluation of members of this class; and a proof of the correctness of the algorithm is given. A further specialization of this class that can be evaluated even more efficiently is discussed. Finally, a generalization of the PIC class, for which the input probabilities may be viewed as weighted, is defined, and a simple modification of the PIC-EVAL algorithm that allows for the evaluation of matrices of this extended class is given.

1 Introduction

The search engine of the INQUERY system is based on the use of inference networks as developed in Howard Turtle's doctoral dissertation. This document presents ideas for the efficient evaluation of link matrices. We begin with a brief description of how inference networks are used in INQUERY and the problems of efficient computation that present themselves. We then introduce the notation that will be adopted for this paper and, using it, give a formal definition of PIC matrices, the subclass of link matrices that will be the focus of our attention.

1.1 link matrices in INQUERY

The retrieval mechanism in the INQUERY system is modeled as an inference network. An inference network is a directed acyclic graph that is used to represent a discrete joint probability distribution, each node associated with a distinct variable.

The topology of an inference network is interpreted as encoding a set of conditional independence conditions. If the nodes corresponding to the variables, $P_1, \ldots, P_n$, are the immediate predecessors (*parents*) of a node, Q, and $Z_1, ..., Z_s$ are all other nodes that are not *descendents* of (i.e. are not reachable from) Q, as shown in figure 1, then Q is considered to be conditionally independent of $Z_1, ..., Z_s$ given $P_1, \ldots, P_n$. That is:

$$pr(Q \text{ is true} \mid P_1, \ldots, P_n, Z_1, \ldots, Z_s) = pr(Q \text{ is true} \mid P_1, \ldots, P_n)$$

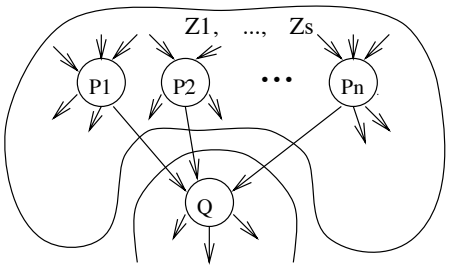

Figure 1: part of an inference net

Given probabilities for the root nodes (i.e. nodes with no parents), the network may be processed in a top-down fashion in order to produce the probabilities relevant to each of its nodes. As a consequence of the independence assumptions implicit in the network topology, once probabilities, $p_1, \ldots, p_n$, have been produced for the parents of a node, Q, the probability that Q takes the value $y \in D_q$ is given by:

$$pr(Q = y) = \sum_{x_1 \in D_1, \ldots, x_n \in D_n} pr(Q = y \mid P_1 = x_1, \ldots, P_n = x_n) pr(P_1 = x_1) \cdots pr(P_n = x_n)$$

where $D_1, \ldots, D_n$, D_q are the sets of values that may be assumed by the variables, $P_1, \ldots, P_n$, and Q, respectively.

In INQUERY each node corresponds to a proposition; that is, a variable that may take on one of two values: *true* or *false*. For example, in figure 1, each P_i might correspond to the proposition that some document under consideration is "*about*" some concept, c_i, while Q corresponds to the proposition that the document is *relevant*.

Since all the variables are binary valued in INQUERY, the dependence of a child on its parents can be given via the specification of:

$$\begin{array}{l} pr(Q \text{ is true} \mid P_1 = b_1, \ldots, P_n = b_n) \\ pr(Q \text{ is false} \mid P_1 = b_1, \ldots, P_n = b_n) \end{array} \quad \textbf{and}$$

for each:

$$< b_1, \ldots, b_n > \in \{\text{true}, \text{false}\}^n$$

Equivalently, the values:

$$\begin{array}{rcll} \bar{\alpha}_R & = & pr(Q \text{ is false} \mid & i \in R \Rightarrow P_i \text{ is true} \\ & & & i \notin R \Rightarrow P_i \text{ is false}) \quad \textbf{and} \\ \alpha_R & = & pr(Q \text{ is true} \mid & i \in R \Rightarrow P_i \text{ is true} \\ & & & i \notin R \Rightarrow P_i \text{ is false}) \end{array}$$

must be specified for every possible subset, R, of $\{1, \ldots, n\}$. In terms of these conditional properties, the probability that a child node is true can be calculated once the probabilities of truth, $p_1, \ldots, p_n$, are known for each of its parent nodes:

$$\begin{array}{rcl} pr(Q \text{ is false}) & = & \displaystyle\sum_{R \subseteq \{1,\ldots,n\}} \bar{\alpha}_R \prod_{i \in R} p_i \prod_{i \notin R} (1 - p_i) \quad \textbf{and} \\ pr(Q \text{ is true}) & = & \displaystyle\sum_{R \subseteq \{1,\ldots,n\}} \alpha_R \prod_{i \in R} p_i \prod_{i \notin R} (1 - p_i) \end{array}$$

The set of coefficients involved is conveniently organized as a 2×2^n matrix:

	$P_{0...000}$	$P_{0...001}$	$P_{0...010}$	$\cdots$	$P_{1...111}$
Q false	$\bar{\alpha}_{0...000}$	$\bar{\alpha}_{0...001}$	$\bar{\alpha}_{0...010}$	$\cdots$	$\bar{\alpha}_{1...111}$
Q true	$\alpha_{0...000}$	$\alpha_{0...001}$	$\alpha_{0...010}$	$\cdots$	$\alpha_{1...111}$

where $\alpha_{b_1,b_2,...b_n}$ is the probability that Q is true subject to the condition that the parents P_i such that $b_i = 1$ are true, and the parents P_i such that $b_i = 0$ are false; $\bar{\alpha}_{b_1,b_2,...b_n}$ is the corresponding probability that Q is false and is equal to $1 - \alpha_{b_1,b_2,...b_n}$. This matrix, known as a *link matrix*, may be visualized as linking the child node with the parent nodes as is shown in figure 2 [1].

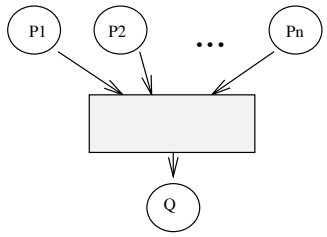

Figure 2: *link matrix* links child to parents

INQUERY examines documents one by one. For each, the inference network is used to evaluate evidence that the document satisfies an information need expressed by the user. A given link matrix form can be viewed as defining an operator for combining evidence. For example, suppose the propositions P_1, P_2, P_3, state that three queries q_1, q_2, q_3, respectively, have been (in some sense) *satisfied* by the document currently under scrutiny. A link matrix connecting the parents nodes, P_1, P_2, P_3, with the child node, Q, can be viewed as a way of forming a query, q, that is a composite of the individual sub-queries. The child node, Q, would correspond to the proposition that the combined query, q, has been satisfied. The specification of the coefficients of the link matrix defines the way the sub-queries are *combined* in that it gives all the information necessary for determining the probability that q has been satisfied given the probabilities, p_1, p_2, p_3, that the individual subqueries have been satisfied.

One, admittedly arbitrary, way of defining a 3-ary query composition operator for forming the query, q, from the sub-queries q_1, q_2, q_3, might be to specify that q is to be considered satisfied with:

- 80% probability if q_1 is satisfied, independent of the whether or not q_2 and q_3 are satisfied;
- 50% probability if q_1 is not satisfied, but q_2 and q_3 are both satisfied
- 10% probability in any other situation.

This particular operator corresponds to the link matrix:

P_{000}	P_{001}	P_{010}	P_{011}	P_{100}	P_{101}	P_{110}	P_{111}
.1	.1	.1	.5	.8	.8	.8	.8

Where the column under P_{011}, for example, gives the probability that Q is true given that P_1 is false, P_2 is true, and P_3 is true. Clearly, this link matrix has the desired effect. Typically, none of the parents will be known to be either true or false with certainty. Rather, evidence corresponding to each of P_1, P_2, P_3 will, in general, be estimated to be present with certain probabilities: p_1, p_2, p_3. Given these probabilities, the probability that Q is true can be calculated as:

$$\begin{aligned} pr(Q \text{ is true}) = \quad & .1 \cdot \bar{p}_1\bar{p}_2\bar{p}_3 + .1 \cdot \bar{p}_1\bar{p}_2 p_3 + .1 \cdot \bar{p}_1 p_2\bar{p}_3 + .5 \cdot \bar{p}_1 p_2 p_3 \\ + \quad & .8 \cdot p_1\bar{p}_2\bar{p}_3 + .8 \cdot p_1\bar{p}_2 p_3 + .8 \cdot p_1 p_2\bar{p}_3 + .8 \cdot p_1 p_2 p_3 \end{aligned}$$

[1] Since the columns of a link matrix must sum to one, from this point on only the row corresponding to Q true will be shown.

1.2 efficiency of link matrix evaluation

Each distinct assignment of values to the coefficients of a link matrix defines a different operator. This provides a large class of operators to choose from when determining, for example, what operators might be made available to users for the composition of queries. Two problems with this class of operators present themselves; both direct consequences of the number of coefficients in the link matrix. Since each parent is a proposition that may assume 2 values, there are 2^n combinations of possible values for the set of n parents. For each matrix, 2^n values must be specified. Equally problematic is that once the coefficients have been specified, the calculations required for the evaluation of the child, Q, given the probabilities, $p_1, \ldots, p_n$, for the parents is exponential in the number of parents. For the general case, the n term product:

$$\prod_{i \in R} p_i \prod_{i \in \{i_1,\ldots,i_2\}-R} \bar{p}_i$$

must be calculated for each of the 2^n possible subsets of n parents; i.e. each of the 2^n possible combinations of true parents. The vast universe of possible operators from which to choose, coupled with the intractable computational properties of these operators in the general case motivates the search for a subclass of link matrices that:

- correspond to psychologically plausible ways of combining evidence, and
- can be put into a tractable computational form

In his dissertation, Howard Turtle refers to link matrix classes with the above two properties as *canonical forms*. He presents four such canonical forms; each general in the sense that there is an operator (or set of operators) for any number of parents. Here we see the form of these matrices for the case of n=3 parents:

L_{and}:

P_{000}	P_{001}	P_{010}	P_{011}	P_{100}	P_{101}	P_{110}	P_{111}
0	0	0	0	0	0	0	1

L_{or}:

P_{000}	P_{001}	P_{010}	P_{011}	P_{100}	P_{101}	P_{110}	P_{111}
0	1	1	1	1	1	1	1

L_{sum}:

P_{000}	P_{001}	P_{010}	P_{011}	P_{100}	P_{101}	P_{110}	P_{111}
0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{2}{3}$	1

$L_{weighted\text{-}sum}$:

P_{000}	P_{001}	P_{010}	P_{011}	P_{100}	P_{101}	P_{110}	P_{111}
0	$w_3 t$	$w_2 t$	$(w_2+w_3)t$	$w_1 t$	$(w_1+w_3)t$	$(w_1+w_2)t$	$(w_1+w_2+w_3)t$

where $t = \frac{w_q}{w_1+\ldots+w_n}$

These canonical forms were motivated by operators traditionally found in IR systems. L_{and} and L_{or} can be considered probabilistic versions of the AND and OR boolean operators offered in boolean retrieval systems. It can easily be shown that, given probabilities, $p_1, \ldots, p_n$, for the n constituent queries being satisfied, the probability that the composite query is satisfied can be calculated simply as:

$$pr(Q \text{ is true}) \quad = \quad p_1 p_2 \cdots p_n$$

for L_{and}, and

$$pr(Q \text{ is true}) \quad = \quad 1 - (1-p_1)(1-p_2)\ldots(1-p_n)$$

for L_{or}.

The coefficient for a column of the general n-ary L_{sum} matrix for which i of the parents are true is, $\frac{i}{n}$. If the probabilities are viewed as weights of evidence, the matrix can be viewed as an operator which averages these

weights. That is:

$$pr(Q \text{ is true}) = \frac{p_1 + p_2 + ... + p_n}{n}$$

The $L_{weighted\text{-}sum}$ matrix is a generalization of the L_{sum} matrix in that a weighted average is calculated with, $w_1, \ldots, w_n$ being the, fixed but arbitrarily chosen, weights of the n parents, and w_q a weight associated with the child node which is applied to the resultant average.

$$\begin{aligned} pr(Q \text{ is true}) &= (w_1 p_1 + w_2 p_2 + ... + w_n p_n)t \\ &= \frac{w_1 p_1 + w_2 p_2 + ... + w_n p_n}{w_1 + w_2 + ... + w_n} w_q \end{aligned}$$

1.3 PIC matrices

One interesting class of link matrices worthy of study are those, such as the L_{sum} matrix discussed above, that do not distinguish among the parents. In these matrices, the coefficient in each column is a function only of the number of parents that are true for the corresponding event, independent of which parents they happen to be. After introducing notational conventions that will be used throughout this discussion, a more formal definition of this class of matrices is presented.

Notation: In this paper we will concentrate on the evaluation of one particular link matrix which we shall refer to as $\boldsymbol{L_0}$. We will consistently use $\boldsymbol{n}$ for the number of parent nodes associated with this matrix; $\boldsymbol{P_1, ..., P_n}$ to represent the propositions corresponding to these parents nodes; and $\boldsymbol{Q}$ to represent the proposition corresponding to the child node.

Notation: We shall use $\boldsymbol{p_i}$ to denote the probability that the $\boldsymbol{i^{th}}$ parent is true and $\boldsymbol{\bar{p}_i}$ to denote the probability that it is false:

$$\begin{aligned} p_i &= pr(P_i \text{ is true}) \\ \bar{p}_i &= pr(P_i \text{ is false}) = (1 - p_i) \end{aligned}$$

Notation: Given an arbitrary subset, R, of $\{1, \ldots, n\}$, $\boldsymbol{\alpha_R}$ shall denote the link matrix coefficient corresponding to the conditional probability that Q is true given that the parents, P_i such that $i \in R$, are true and the other parents are false:

$$\alpha_R = pr(\text{Q is true} \quad | \quad \forall i = 1, \ldots, n: \quad i \in R \Rightarrow P_i \text{ is true}, \quad i \notin R \Rightarrow P_i \text{ is false})$$

Definition: When the conditional probability that Q is true is a function only of the number of parents that are true we shall say that the *parent indifference criterion* is met, or simply that the matrix is a PIC matrix. Equivalently,

$$\forall R_1, R_2 \subseteq \{1, \ldots, n\}: \quad |R_1| = |R_2| \Rightarrow \alpha_{R_1} = \alpha_{R_2}$$

Notation: When a matrix satisfies the parent indifference criterion, $\boldsymbol{\alpha_j}$ shall be used to denote the conditional probability that Q is true given that exactly j of the parents are true.

$$\alpha_j = pr(Q \text{ is true} \mid |\{0 \leq i \leq n | P_i \text{ is true}\}| = j)$$

The sequence of link matrix coefficients, $\alpha_0, \ldots, \alpha_n$, can be viewed as a function from the integers $\{0, 1, \ldots, n\}$ to the interval [0,1]. It is appropriate to note that we have in mind, and will be exploring, coefficients corresponding to non-decreasing functions. Viewing the PIC matrices as operators for the combination of evidence,

our interest is in those operators for which more pieces of *individual evidence* (i.e. greater number of true parents) translates to greater probability that the *combined evidence* is present. Nonetheless, the formal development to follow does not depend on this property of the functions, and so it has not been incorporated as part of the definition of PIC matrices.

1.4 this study

Howard Turtle has suggested further study of matrices which satisfy what I have called the parent indifference criterion, which has been the initial impulse behind the work discussed here. It should be noted, that the L_{and} and L_{or} matrices meet the parent indifference criterion. As such, they are special cases of the class of matrices under study. A satisfactory approach to computing with PIC matrices will allow for the definition of less stringent versions of these operators, which may be, Turtle speculates, more in accord with users' intuitive understanding of what they are trying to express when they utilize them.

In what follows, I will present an algorithm for evaluating an arbitrary PIC matrix which requires $O(n^2)$ operations. Later I define a more general class of matrices for which the parent indifference criterion is relaxed to allow for a weighting of the parents. A simple modification of the basic algorithm expands its applicability to include this more general class.

Again following Howard Turtle's suggestions, I also explore specializations of the PIC matrix class. Viewing the set of coefficients of a PIC matrix as a function on the number of parents that are true, we take a look at link matrices associated with piecewise linear functions. We see that these may be evaluated in $O(n)$ time in certain restricted cases.

2 Basic Algorithm

In this section, an algorithm, PIC-EVAL, is presented for evaluating link matrices that meet the parent indifference criterion. Starting with the original link matrix, L_0, PIC-EVAL, in effect, generates a sequence of smaller and smaller link matrices, $L_1, L_2, \ldots, L_n$. In the process, it eliminates from consideration each probability in turn (Figure 3).

2.1 description of the PIC-EVAL algorithm

To begin, the probability associated with parent node, P_1, is fixed and the matrix, L_0, with $n+1$ coefficients and n parents (figure 3a) is converted to an n coefficient link matrix with $n-1$ parents (figure 3b). The matrix, L_1, that results from this transformation is equivalent to the original matrix, L_0, in the following sense:

> *for any set of probabilities for parents, P_2, ..., P_n, L_1 yields the same pr(Q is true) that L_0 would produce given the same probabilities for P_2, ..., P_n, together with the probability p_1 for P_1.*

Notation: With respect to the execution of the PIC matrix evaluation algorithm: $\boldsymbol{L_i}$ shall refer to the link matrix generated during the i^{th} iteration, and the coefficient of L_i associated with j parents being true shall be referred to as $\boldsymbol{\alpha(i,j)}$. We note that for the initial matrix, L_0:

$$\alpha(0,j) = \alpha_j \quad \forall i = 0, \ldots, n$$

As shown in figure 3, each matrix has one coefficient fewer than the previous one and connects with one fewer parent node. After n iterations we arrive at L_n with exactly one coefficient, $\alpha(n,0)$, as seen in figure 3e. Given that all parent probabilities have been accounted for, this coefficient may be interpreted as the desired probability that the child is true. This general strategy is embodied in the PIC-EVAL algorithm given in figure 4.

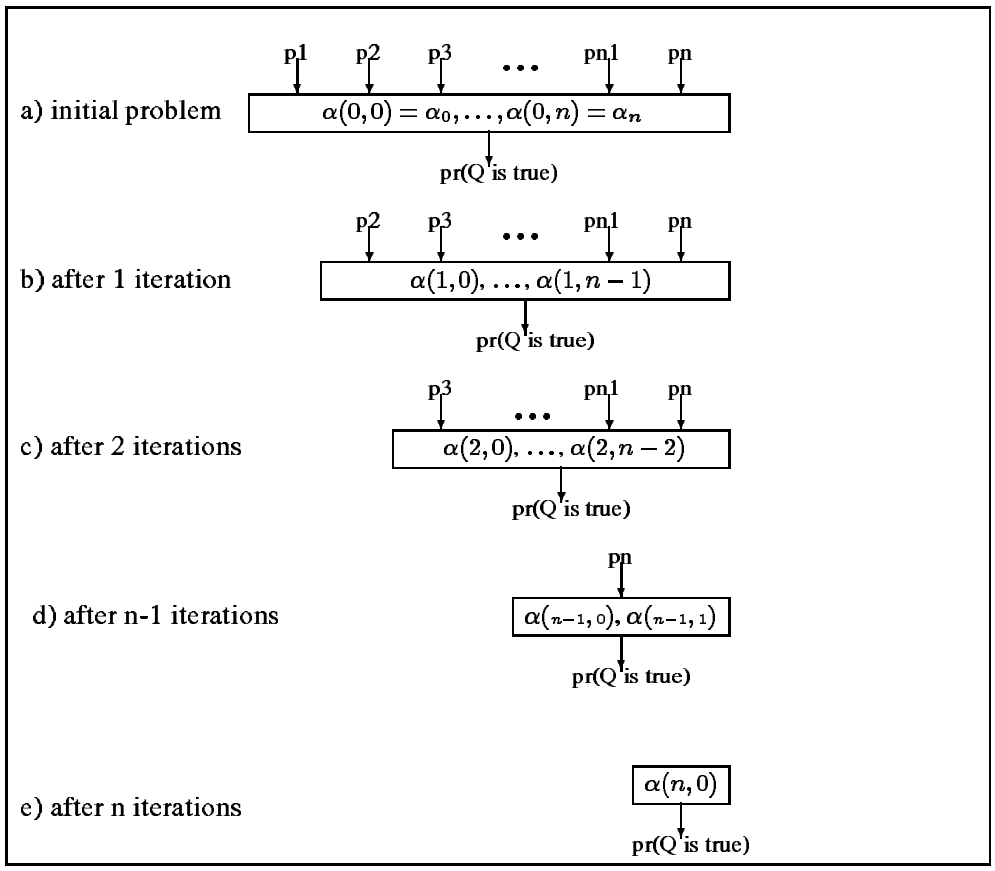

Figure 3: iterations in the evaluation of a PIC matrix

inputs:

- $n+1$ coefficients, $\alpha_0, \ldots, \alpha_n$, of a PIC matrix
- probabilities $p_1, \ldots, p_n$, for the n parent nodes.

output:

- the probability that the child node is true.

PIC-EVAL:

for $j = 0, \ldots, n$
  $\alpha[0, j] = \alpha_j$

for $i = 1, \ldots, n$
  for $j = 0, \ldots, n - i$
    $\alpha[i, j] \leftarrow \alpha[i-1, j] * (1 - p_i) + \alpha[i-1, j+1] * p_i$

return $\alpha[n, 0]$

Figure 4: PIC-EVAL algorithm

2.2 correctness of PIC-EVAL algorithm

The correctness of the pic algorithm follows directly from Lemma 1, which is more easily expressed by adopting the following notation.

Notation: Given an arbitrary subset, R, of $\{i_1, \ldots, i_2\}$, $\pi_R^{\{i_1,\ldots,i_2\}}$ shall denote the probability that precisely the parents, P_i such that $i \in R$, are true, while those parents, P_i such that $i \notin R$, are false. Equivalently:

$$\pi_R^{\{i_1,\ldots,i_2\}} = \prod_{i \in R} p_i \prod_{i \in \{i_1,\ldots,i_2\}-R} \bar{p}_i$$

Notation: The notation $\sigma_j^{\{i_1,\ldots,i_2\}}$ will be used to denote the probability that exactly j of the propositions of $\{P_{i_1}, \ldots, P_{i_2}\}$ are true. We observe that:

$$\sigma_j^{\{i_1,\ldots,i_2\}} = \sum_{\substack{R \subseteq \{i_1,\ldots,i_2\} \\ |R|=j}} \pi_R^{\{i_1,\ldots,i_2\}}$$

It is worth noting that the probability that Q is true can be given by:

$$\begin{aligned} pr(Q \text{ is true}) &= \sum_{R \subseteq \{1,\ldots,n\}} [\alpha_R \prod_{i \in R} p_i \prod_{i \in \{1,\ldots,2\}-R} \bar{p}_i] \\ &= \sum_{R \subseteq \{1,\ldots,n\}} [\alpha_R \pi_R^{\{1,\ldots,n\}}] \end{aligned}$$

And that when the parent indifference criterion is met:

$$\begin{aligned} pr(Q \text{ is true}) = \sum_{R \subseteq \{1,\ldots,n\}} [\alpha_R \pi_R^{\{1,\ldots,n\}}] &= \sum_{j=0}^{n} \sum_{\substack{R \subseteq \{1,\ldots,n\} \\ |R|=j}} [\alpha_R \pi_R^{\{1,\ldots,n\}}] \\ &= \sum_{j=0}^{n} \sum_{\substack{R \subseteq \{1,\ldots,n\} \\ |R|=j}} [\alpha_j \pi_R^{\{1,\ldots,n\}}] \\ &= \sum_{j=0}^{n} [\alpha_j \sum_{\substack{R \subseteq \{1,\ldots,n\} \\ |R|=j}} \pi_R^{\{1,\ldots,n\}}] \\ &= \sum_{j=0}^{n} \alpha_j \sigma_j^{\{1,\ldots,n\}} \end{aligned}$$

Lemma 1 *Assuming all coefficients $\alpha(i,j)$ are those produced by the* PIC-EVAL *algorithm, then $\forall i = 0, 1, \ldots, n$*

$$\sum_{j=0}^{n-i} \alpha(i,j) \sigma_j^{\{i+1,\ldots,n\}} = pr(Q \textit{ is true})$$

Lemma 1 effectively states that the set of coefficients, $\alpha(i,0), \ldots, \alpha(i,n-i)$, can be interpreted as specifying a PIC matrix, L_i, connecting parent nodes $P_{i+1}, \ldots, P_n$, to Q, and that this matrix is, in a sense, equivalent to the original matrix, L_0.

proof: This lemma is proved by induction on the value of i. For $i = 0$, we have:

$$\sum_{j=0}^{n} \alpha(0,j) \sigma_j^{\{1,\ldots,n\}} = \sum_{j=0}^{n} \alpha_j \sum_{\substack{R \subseteq \{1,\ldots,n\} \\ |R|=j}} \pi_R^{\{1,\ldots,n\}}$$

$$
\begin{aligned}
&= \sum_{j=0}^{n} \sum_{\substack{R\subseteq\{1,\ldots,n\} \\ |R|=j}} [\alpha_R \prod_{i\in R} p_i \prod_{i\notin R} \bar{p}_i] \\
&= \sum_{R\subseteq\{1,\ldots,n\}} [\alpha_R \prod_{i\in R} p_i \prod_{i\notin R} \bar{p}_i] \\
&= pr(Q \text{ is true})
\end{aligned}
$$

Assume the theorem to be true for $i = k$:

$$\sum_{j=0}^{n-k} \alpha(k,j)\sigma_j^{\{1,\ldots,n\}} = pr(Q \text{ is true})$$

Then the sum for $i = k+1$ is:

$$
\begin{aligned}
&\sum_{j=0}^{n-k-1} \alpha(k+1,j)\sigma_j^{\{k+2,\ldots,n\}} \\
&= \sum_{j=0}^{n-k-1} [\alpha(k,j)\bar{p}_{k+1} + \alpha(k,j+1)p_{k+1}]\sigma_j^{\{k+2,\ldots,n\}} \\
&\quad \text{(by step (5) of the algorithm)} \\
&= \sum_{j=0}^{n-k-1} \alpha(k,j)\bar{p}_{k+1}\sigma_j^{\{k+2,\ldots,n\}} + \sum_{j=0}^{n-k-1} \alpha(k,j+1)p_{k+1}\sigma_j^{\{k+2,\ldots,n\}} \\
&= \sum_{j=0}^{n-k-1} \alpha(k,j)\bar{p}_{k+1}\sigma_j^{\{k+2,\ldots,n\}} + \sum_{j=1}^{n-k} \alpha(k,j)p_{k+1}\sigma_{j-1}^{\{k+2,\ldots,n\}} \\
&\quad \text{(as a result of changing the variable for the second summation)} \\
&= \alpha(k,0)\bar{p}_{k+1}\sigma_0^{\{k+2,\ldots,n\}} \qquad (1) \\
&\quad + \sum_{j=1}^{n-k-1} \alpha(k,j)\bar{p}_{k+1}\sigma_j^{\{k+2,\ldots,n\}} + \sum_{j=1}^{n-k-1} \alpha(k,j)p_{k+1}\sigma_{j-1}^{\{k+2,\ldots,n\}} \\
&\quad + \alpha(k,n-k)p_{k+1}\sigma_{n-k-1}^{\{k+2,\ldots,n\}}
\end{aligned}
$$

The term $\sigma_0^{\{k+2,\ldots,n\}}$, expresses the probability that none of the propositions, $P_{k+2}, \ldots, P_n$, is true. It is composed of only one product. Hence, the first term of (1) reduces to:

$$
\begin{aligned}
\alpha(k,0)\bar{p}_{k+1}\sigma_0^{\{k+2,\ldots,n\}} &= \alpha(k,0)\bar{p}_{k+1} \prod_{l=k+2}^{n} \bar{p}_l \\
&= \alpha(k,0) \prod_{l=k+1}^{n} \bar{p}_l \\
&= \alpha(k,0)\sigma_0^{\{k+1,\ldots,n\}} \qquad (2)
\end{aligned}
$$

Similarly, $\sigma_{n-k+1}^{\{k+2,\ldots,n\}}$, which expresses the probability that all $n-k+1$ of the propositions $\{P_{k+2}, \ldots, P_n\}$ are true, is composed of only one product. Therefore, the last term of (1) reduces to:

$$
\begin{aligned}
\alpha(k,n-k)p_{k+1}\sigma_{n-k+1}^{\{k+2,\ldots,n\}} &= \alpha(k,n-k)p_{k+1} \prod_{l=k+2}^{n} p_l \\
&= \alpha(k,n-k) \prod_{l=k+1}^{n} p_l \\
&= \alpha(k,n-k)\sigma_{n-k}^{\{k+1,\ldots,n\}} \qquad (3)
\end{aligned}
$$

Combining the middle two terms of (1):

$$\begin{aligned}
&\sum_{j=1}^{n-k-1} \alpha(k,j)\bar{p}_{k+1}\sigma_j^{\{k+2,\ldots,n\}} + \sum_{j=1}^{n-k-1} \alpha(k,j)p_{k+1}\sigma_{j-1}^{\{k+2,\ldots,n\}} \\
&= \sum_{j=1}^{n-k-1} \alpha(k,j)[(\bar{p}_{k+1}\sigma_j^{\{k+2,\ldots,n\}}) + (p_{k+1}\sigma_{j-1}^{\{k+2,\ldots,n\}})] \\
&= \sum_{j=1}^{n-k-1} \alpha(k,j)[(\bar{p}_{k+1} \sum_{\substack{R\subseteq\{k+2,\ldots,n\}\\|R|=j}} \pi_R^{\{k+2,\ldots,n\}}) + (p_{k+1} \sum_{\substack{R\subseteq\{k+2,\ldots,n\}\\|R|=j-1}} \pi_R^{\{k+2,\ldots,n\}})] \\
&= \sum_{j=1}^{n-k-1} \alpha(k,j)[(\sum_{\substack{R\subseteq\{k+1,\ldots,n\}\\|R|=j}} \bar{p}_{k+1}\pi_R^{\{k+2,\ldots,n\}}) + (\sum_{\substack{R\subseteq\{k+1,\ldots,n\}\\|R|=j-1}} p_{k+1}\pi_R^{\{k+2,\ldots,n\}})] \\
&= \sum_{j=1}^{n-k-1} \alpha(k,j)[\sum_{\substack{R\subseteq\{k+1,\ldots,n\}\\|R|=j\wedge k+1\notin R}} \pi_R^{\{k+1,\ldots,n\}} + \sum_{\substack{R\subseteq\{k+1,\ldots,n\}\\|R|=j\wedge k+1\in R}} \pi_R^{\{k+1,\ldots,n\}}] \\
&= \sum_{j=1}^{n-k-1} [\alpha(k,j) \sum_{\substack{R\subseteq\{k+1,\ldots,n\}\\|R|=j}} \pi_R^{\{k+1,\ldots,n\}}] \\
&= \sum_{j=1}^{n-k-1} \alpha(k,j)\sigma_j^{\{k+1,\ldots,n\}} \qquad (4)
\end{aligned}$$

Finally, applying equations (2), (3), and (4) to equation (**1**), we have:

$$\begin{aligned}
&\sum_{j=0}^{n-k-1} \alpha(k+1,j)\sigma_j^{\{k+2,\ldots,n\}} \\
&= \alpha(k,0)\sigma_0^{\{k+1,\ldots,n\}} + [\sum_{j=1}^{n-k-1} \alpha(k,j)\sigma_j^{\{k+1,\ldots,n\}}] + \alpha(k,n-k)\sigma_{n-k}^{\{k+1,\ldots,n\}} \\
&= \sum_{j=0}^{n-k} \alpha(k,j)\sigma_j^{\{k+1,\ldots,n\}} \\
&= pr(Q \text{ is true}) \qquad \textbf{(by the induction hypothesis)}
\end{aligned}$$

Setting $i = n$ in Lemma 1 immediately yields the following theorem which states that the PIC-EVAL algorithm is correct.

Theorem 1 *Given that $\alpha(n,0)$ has been produced by the* PIC-EVAL *algorithm:*

$$\alpha(n,0) = pr(Q \textit{ is true})$$

2.3 efficiency of the PIC-EVAL algorithm

The PIC-EVAL algorithm is clearly executed in $O(n^2)$ time. The initialization requires $0(n)$ and the main loop is executed precisely

$$\sum_{i=0}^{n}\sum_{j=0}^{n-i} 1 = \sum_{i=0}^{n}(n-i+1) = \sum_{i=1}^{n+1} i = (n+2)(n+1)/2 = O(n^2)$$

times. Also, the constant factor is small. On top of the base iteration control overhead, two multiplications and one addition are required in each iteration.

Although, for the purposes of exposition, the algorithm has been shown as requiring $O(n^2)$ space as well as time, $O(n)$ space is easily achieved since only one row's worth of cells need be maintained at any one time.

3 Piecewise Linear Functions

In this section we will look at certain PIC matrices for which the evaluation algorithm can be made more efficient.

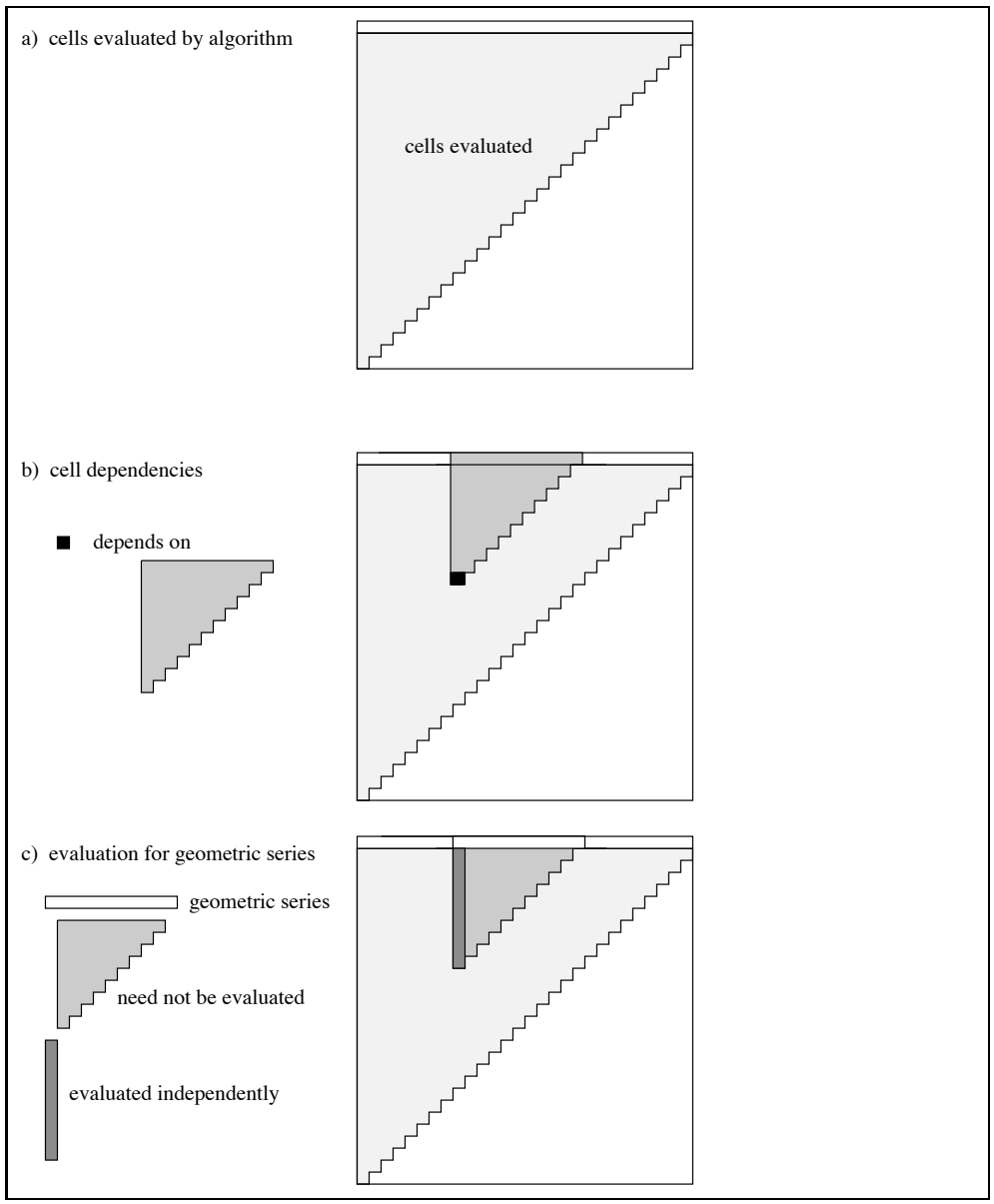

Figure 5: cell evaluation during execution of PIC-EVAL

The PIC-EVAL algorithm can be viewed as filling a triangular portion of a square matrix as shown in figure 5a.

The first row is initialized with the α_j values and then each row from 1 to n is processed in turn. Within each row, the cells are set from left to right, with each cell set to the sum of:

- the cell immediately above it, and
- the cell above it and to the right.

As a consequence of this, the value of each cell, $\alpha(i,j)$, is dependent only on the values of cells in a triangle extending directly above it on the left and at a 45^o angle to the right, as is shown in figure 5b. That is, $\alpha(i,j)$ depends on only those $\alpha(k,l)$ such that:

$$\begin{array}{ccccc} 0 & \leq & k & < & i \\ j & \leq & l & < & j+m-i \end{array}$$

When a subsequence of the PIC matrix coefficients forms an arithmetic progression,

$$\alpha_m, \alpha_m + \Delta, \alpha_m + 2\Delta, \ldots, \alpha_m + s\Delta$$

the cell values in the triangular subsection immediately below these coefficients can be expressed directly in terms of α_m, Δ, and the parent probabilities, $p_1, \ldots, p_s$. Of these cell values, the only ones that are needed for calculations outside of the triangle are those on the leftmost edge:

$$\alpha(0,j), \ldots, \alpha(i,j)$$

Hence, if there were a direct method for determining these cell values, the rest of the cells in the triangle need not be calculated at all. Exactly, how this may be accomplished follows from the following Lemma.

Lemma 1 *Given a* PIC *matrix whose coefficients,*

$$\alpha_m, \alpha_{m+1}, \alpha_{m+2}, \ldots, \alpha_{m+s}$$

are of the form:

$$\alpha_m, \alpha_m + \Delta, \alpha_m + 2\Delta, \ldots, \alpha_m + s\Delta$$

i.e., are such that:

$$\alpha_{m+j} = \alpha_m + j\Delta \quad \forall j = 0, ..., s$$

then, $\forall i = 0, \ldots, s \quad \forall j = 0, \ldots, s-i$*:*

$$\alpha(i, m+j) = \alpha_m + j\Delta + \Delta \sum_{l=1}^{i} p_j$$

proof: By induction on i. The base case follows directly from the hypothesis of the lemma.

$$\begin{array}{rcl} \alpha(0, m+j) = \alpha_{m+j} & = & \alpha_m + j\Delta \\ & = & \alpha_m + j\Delta + \Delta \sum_{l=1}^{0} p_j \end{array}$$

Assuming the lemma to be true for i=k-1:

$$\begin{array}{l} \alpha(k, m+j) \\ \quad = \quad \alpha(k-1, m+j)(1-p_k) \quad + \quad \alpha(k-1, m+j+1)p_k \\ \quad = \quad (\alpha_m + j\Delta + \Delta \sum_{l=1}^{k-1} p_l)(1-p_k) \quad + \quad (\alpha_m + (j+1)\Delta + \Delta \sum_{l=1}^{k-1} p_l)p_k \end{array}$$

$$
\begin{aligned}
&= \alpha_m + (j\Delta + \Delta\sum_{l=1}^{k-1} p_l)(1-p_k) + ((j+1)\Delta + \Delta\sum_{l=1}^{k-1} p_l)p_k \\
&= \alpha_m + j\Delta + (\Delta\sum_{l=1}^{k-1} p_l) - j\Delta p_k - (\Delta\sum_{l=1}^{k-1} p_l)p_k \\
&\quad + (j+1)\Delta p_k + (\Delta\sum_{l=1}^{k-1} p_l)p_k \\
&= \alpha_m + j\Delta + (\Delta\sum_{l=1}^{k-1} p_l) - j\Delta p_k + (j+1)\Delta p_k \\
&= \alpha_m + j\Delta + (\Delta\sum_{l=1}^{k-1} p_l) + \Delta p_k \\
&= \alpha_m + j\Delta + (\Delta\sum_{l=1}^{k} p_l)
\end{aligned}
$$

In particular, each of the cells at the left edge of the triangle can be computed as:

$$
\begin{aligned}
\alpha(i,m) &= \alpha_m + 0\Delta + (\Delta\sum_{k=1}^{i} p_k) \\
&= \alpha_m + 0\Delta + (\Delta\sum_{k=1}^{i-1} p_k) + \Delta p_i = \alpha(i-1,m)p_i + \Delta p_i
\end{aligned}
$$

which requires a total of only s additions and s multiplications, replacing the $\sum_{i=1}^{s} = s(s+1)/2$ general cell computations which would normally be executed.

This technique can result in substantial savings if the number of coefficients involved, and hence the size of the triangle involved, is large. In his notes, Howard Turtle, has suggested looking at the case where the PIC matrix coefficients correspond to a piecewise linear function on the number of true parents. For example, figure 6a, shows a function with three linear pieces. The matrix coefficients in this case comprise three arithmetic series, each associated with a corresponding savings in cost of evaluation. Since the domain of the function is discrete, it is not strictly necessary that the pieces *connect*. Therefore, when speaking of piecewise linear functions, we shall also consider functions such as that shown in figure 6b.

The form of the 2-piece piecewise linear functions shown in figures 6c and 6d are of interest because they can be interpreted as generalizations of the L_{and} and L_{or} link matrices. We will see in a moment that evaluation of these functions is particularly efficient. The function shown in figure 6c, for instance, generalizes the L_{and} matrix in that the conditional probability that Q is true may:

- take on a (presumably small) value greater than 0 when all parents are false.
- rise (presumably slowly) at a constant rate as more parents are known to be true.
- rise suddenly for some number of true parents m less than n - although m must be less than n by some (presumably small) constant value, independent of n.
- rise at a constant rate as the number of parents known to be true goes from m to n
- take on a value less than 1 when all parents are true.

In a similar fashion, the form shown in figure 6d, generalizes the L_{or} matrix.

Although the savings realized for a piecewise linear function may be significant, the asymptotic cost of execution will not be affected unless all of the pieces save one cover a constant number of coefficients [2]. If $\alpha_m, \alpha_{m+1}, \ldots, \alpha_s$, is an arithmetic series, then $s(s+1)/2$ cell operations can be eliminated in favor of s

[2] Prof. David Barrington has pointed out that computational improvement may be realizable even if this condition is not fully met. For example, if one piece grows with n, while all others grow only as $log(n)$, the asymptotic running time will be only $O(nlog(n))$.

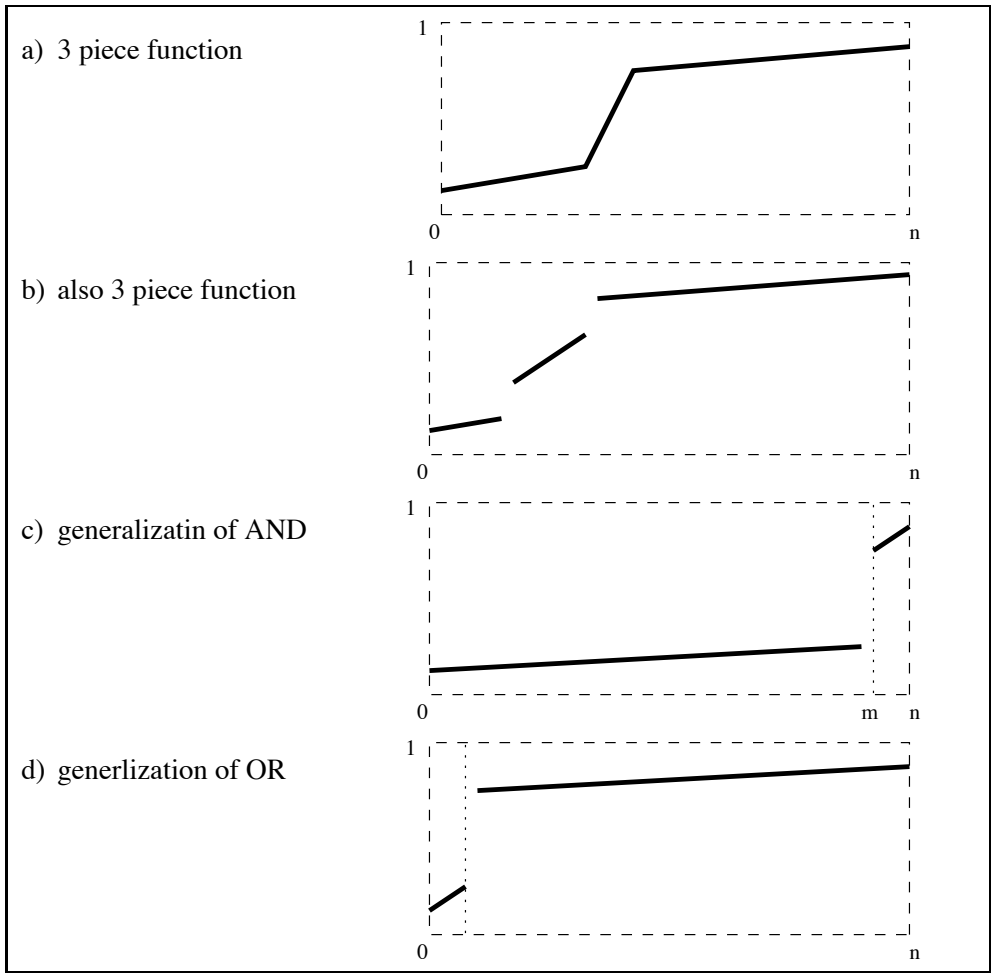

Figure 6: piecewise linear functions

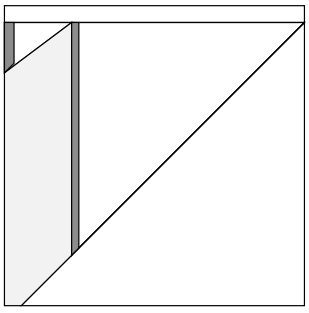

Figure 7: evaluation of generalized OR

additions. This leaves,

$$\begin{aligned} n(n+1)/2 - s(s+1)/2 &= (n^2+n-s^2-s)/2 \\ &= ns - s^2 + (n^2 - ns + n + s^2 - ns - s)/2 \\ &= (n-s)s + (n(n-s+1) - s(n-s+1)/2 \\ &= (n-s)s + (n-s)(n-s+1)/2 \end{aligned}$$

When $(n-s)$ is constant, the second term is constant and the first term grows with s. Since s must grow with n, if $n-s$ is to be kept constant, the cost of evaluation is $O(n)$. Figure 7 shows, pictorially, how the array for the generalized OR function is evaluated. The cells which are evaluated are restricted to a fixed width strip at the left of the array whose length grows with n.

4 Weighted Parents

In the previous section we defined, and developed an $O(n^2)$ algorithm for, the class of link matrices for which the conditional probabilities are dependent only on the number of parents that are true. In this section, we generalize this result, beginning with the class of link matrices under consideration.

The goal of this generalization is to allow for the possibility that the truth of each proposition, P_i, may have a different impact on the probability, *pr*(Q is true). There will always be one parent whose impact is at least as great as any other. For the purposes of this exposition we will assume, without loss of generality, that parent, P_1, has this property. The impact of each other parent may, then, be *weighted* by some factor, $0 \leq w_i \leq 1.0$. For generality, we will say that all parents are weighted and that the weight, w_1, of parent P_1 is equal to 1. We shall say that a link matrix, satisfies the *weighted-parent indifference criterion* when the probability that the child is true is strictly a function of the number of parents that are true, once the weights of the parents have been taken into consideration. Formally,

Definition: We shall say that the *weighted-parent indifference criterion* is met when:

$$\begin{aligned} &\exists\ (w_1 = 1.0, \\ &\quad 0 \leq w_2, \ldots, w_n \leq 1.0 \\ &\quad 0 \leq \alpha_0, \ldots, \alpha_n \leq 1.0) \\ &\qquad \forall\ R \subseteq \{1, \ldots, n\} \qquad \alpha_R = \alpha_j \prod_{i \in R} w_i \end{aligned}$$

We shall refer to a link matrix satisfying the weighted-parent indifference criterion as a WPIC matrix. A WPIC matrix is completely determined when two sets of parameters have been given: $\alpha_0, \ldots, \alpha_n$ and $w_1, \ldots, w_n$. When all the parents have a weight of, $w_i = 1$, the WPIC matrix reduces to a simple PIC matrix, where the coefficient, α_j, specifies the probability that Q is true when it is known that j parents are true.

For the general WPIC matrix, however, rather than specify what the probability that Q is true *is*, for j true parents, the coefficient, α_j, specifies what that probability *would be* for j true parents, if all the parents had the same impact; that is, *if all the weights were 1*. A weight, $w_i < 1$, indicates that when P_i is one of the true parents, the probability, *pr*(Q is true), is less than it would be were P_i to have the same weight as P_1. The weight, w_i, gives the factor by which *pr*(Q is true) must be *discounted* when P_i, rather than a parent whose impact is equal to that of P_1, is one of the true parents [3].

The basic algorithm requires only a minor modification in order that WPIC matrices be processed correctly. The WPIC-EVAL algorithm incorporating the necessary modification is given in figure 8.

[3] Another way of understanding the WPIC matrices is in terms of *gated* inputs. Each parent is viewed as one input to an AND gate whose other input corresponds to an independent *activation* variable. Only when both the parent and the activation variable are both true is a value of true seen by the Q node. The parent weight becomes the probability that the activation variable is true. From this, we see that the class of WPIC matrices can be viewed as a generalization of the *noisy or* matrices often utilized in Bayesian Network applications.

inputs:

- $n+1$ coefficients, $\alpha_0, \ldots, \alpha_n$, of a PIC matrix
- probabilities $p_1, \ldots, p_n$, for the n parent nodes.

output:

- the probability that the child node is true.

PIC-EVAL:

for $j = 0, \ldots, n$
  $\alpha[0, j] = \alpha_j$

for $i = 1, \ldots, n$
  for $j = 0, \ldots, n - i$
    $\alpha[i, j] \leftarrow \alpha[i-1, j] * (1 - p_i) + \alpha[i-1, j+1] * w_i * p_i$

return $\alpha[n, 0]$

Figure 8: WPIC-EVAL algorithm

Lemma 1 and Theorem 1, used to demonstrate the correctness of the PIC-EVAL algorithm, can be modified to contemplate weights. The reasoning used for their proofs applies with very minor modifications. Appendix A, presents generalized versions of the notation introduced for PIC matrices and a modified version of Lemma 1. For reference, the complete proof for the new version of the lemma is included, although it is essentially the same as that given for Lemma 1 in section 2.

It should be observed here that, with respect to computational efficiency, the only difference between the two versions of the algorithms is that the weighted algorithm requires an extra multiplication during each execution of the main loop.

5 Summary

In this paper I have described:

- the PIC matrices, a subclass of link matrices which would appear to correspond to a natural class of retrieval operators.
- an $O(n^2)$ algorithm, PIC-EVAL, for the evaluation of arbitrary PIC matrices.
- an improvement to the general PIC-EVAL algorithm when a subsequence of the link matrix coefficients form an arithmetic series. This improvement results in an $O(n)$ algorithm for certain forms of link matrices which may be considered generalizations of Turtle's L_{and} and L_{or} matrices.
- the WPIC matrices, which allow for the weighting of parent nodes and a simple modification to the basic algorithm which adjusts for this generalization.

In closing, some aspects of the link matrix evaluation problem intimately related to the ideas presented here, can be mentioned.

Piecewise linear functions may be evaluated more efficiently, in some cases much more efficiently, than an arbitrary PIC matrix. Unfortunately it does not immediately follow that these gains in efficiency may be realized when the parents are weighted. A closer look at this problem may yield an interesting subclass of the WPIC matrices that may also be evaluated more efficiently.

When a subsequence of the PIC matrix coefficients form an arithmetic series, the need for a full evaluation of the triangle underneath the subsequence is eliminated. It may be fruitful to investigate other properties that

might result in similar shortcuts to the computation of the entire $O(n^2)$ matrix that is required for the evaluation of an arbitrary PIC matrix.

In a recent discussion with Prof. Eliot Moss, he suggested to me an interesting approach to the efficient evaluation of WPIC matrices. He suggests using

$$\hat{j} = \sum_{i=1}^{n} w_i p_i$$

as an *estimate of the number of parents that are true.* An arbitrary function on the real interval $[0, 1]$ now takes the place of the link matrix. This function is evaluated at $\hat{j}$ to yield an approximation of the probability, $pr(Q$ is true). While this technique is not equivalent to the specification of a link matrix, under appropriate conditions it might provide a practical alternative.

6 Appendix A: Correctness of WPIC-EVAL

Notation: For the purposes of the analysis of WPIC-EVAL, we will adopt the following notation which modifies that used previously for the analysis of the PIC-EVAL version.

$$\pi_R^{\{i_1,\ldots,i_2\}} = \prod_{i\in R} p_i \prod_{i\in\{i_1,\ldots,i_2\}-R} \bar{p}_i$$

$$\sigma_j^{\{i_1,\ldots,i_2\}} = \sum_{\substack{R\subseteq\{i_1,\ldots,i_2\}\\ |R|=j}} \pi_R^{\{i_1,\ldots,i_2\}}$$

Lemma 1 *Assuming all coefficients $\alpha(i,j)$ are those produced by the* WPIC-EVAL *algorithm, we have:* $\forall i = 0,1,\ldots,n$

$$\sum_{j=0}^{n-i}\alpha(i,j)\sigma_j^{\{i+1,\ldots,n\}} = pr(Q \textit{ is true})$$

proof: This lemma is proved by induction on the value of i. For $i=0$, we have:

$$\begin{aligned}
\sum_{j=0}^{n}\alpha(0,j)\sigma_j^{\{1,\ldots,n\}} &= \sum_{j=0}^{n}\alpha_j \sum_{\substack{R\subseteq\{1,\ldots,n\}\\ |R|=j}} \pi_R^{\{1,\ldots,n\}} \\
&= \sum_{j=0}^{n}\sum_{\substack{R\subseteq\{1,\ldots,n\}\\ |R|=j}} [\alpha_j \prod_{i\in R} w_i p_i \prod_{i\notin R}\bar{p}_i] \\
&= \sum_{j=0}^{n}\sum_{\substack{R\subseteq\{1,\ldots,n\}\\ |R|=j}} [\alpha_j \prod_{i\in R} w_i \prod_{i\in R} p_i \prod_{i\notin R}\bar{p}_i] \\
&= \sum_{j=0}^{n}\sum_{\substack{R\subseteq\{1,\ldots,n\}\\ |R|=j}} [\alpha_R \prod_{i\in R} p_i \prod_{i\notin R}\bar{p}_i] \\
&= \sum_{R\subseteq\{1,\ldots,n\}} [\alpha_R \prod_{i\in R} p_i \prod_{i\notin R}\bar{p}_i] \\
&= pr(Q \text{ is true})
\end{aligned}$$

Assume the theorem to be true for $i=k$:

$$\sum_{j=0}^{n-k}\alpha(k,j)\sigma_j^{\{1,\ldots,n\}} = pr(Q \text{ is true})$$

Then the sum for $i=k+1$ is:

$$\begin{aligned}
&\sum_{j=0}^{n-k-1}\alpha(k+1,j)\sigma_j^{\{k+2,\ldots,n\}} \\
&= \sum_{j=0}^{n-k-1}[\alpha(k,j)\bar{p}_{k+1} + \alpha(k,j+1)w_{k+1}p_{k+1}]\sigma_j^{\{k+2,\ldots,n\}} \\
&\quad \textbf{(by step (5) of the algorithm)} \\
&= \sum_{j=0}^{n-k-1}\alpha(k,j)\bar{p}_{k+1}\sigma_j^{\{k+2,\ldots,n\}} + \sum_{j=0}^{n-k-1}\alpha(k,j+1)w_{k+1}p_{k+1}\sigma_j^{\{k+2,\ldots,n\}}
\end{aligned}$$

$$= \sum_{j=0}^{n-k-1} \alpha(k,j)\bar{p}_{k+1}\sigma_j^{\{k+2,\ldots,n\}} + \sum_{j=1}^{n-k}\alpha(k,j)w_{k+1}p_{k+1}\sigma_{j-1}^{\{k+2,\ldots,n\}}$$

(as a result of changing the variable for the second summation)

$$\begin{aligned} = \; & \alpha(k,0)\bar{p}_{k+1}\sigma_0^{\{k+2,\ldots,n\}} \\ & + \sum_{j=1}^{n-k-1} \alpha(k,j)\bar{p}_{k+1}\sigma_j^{\{k+2,\ldots,n\}} + \sum_{j=1}^{n-k-1} \alpha(k,j)w_{k+1}p_{k+1}\sigma_{j-1}^{\{k+2,\ldots,n\}} \\ & + \alpha(k,n-k)w_{k+1}p_{k+1}\sigma_{n-k-1}^{\{k+2,\ldots,n\}} \end{aligned} \tag{5}$$

The term $\sigma_0^{\{k+2,\ldots,n\}}$, expresses the probability that none of the propositions, $P_{k+2},\ldots,P_n$, is true. It is composed of only one product. Hence, the first term of (5) reduces to:

$$\begin{aligned} \alpha(k,0)\bar{p}_{k+1}\sigma_0^{\{k+2,\ldots,n\}} &= \alpha(k,0)\bar{p}_{k+1}\prod_{l=k+2}^{n}\bar{p}_l \\ &= \alpha(k,0)\prod_{l=k+1}^{n}\bar{p}_l \\ &= \alpha(k,0)\sigma_0^{\{k+1,\ldots,n\}} \end{aligned} \tag{6}$$

Similarly, $\sigma_{n-k+1}^{\{k+2,\ldots,n\}}$, which expresses the probability that all $n-k+1$ of the propositions $\{P_{k+2},\ldots,P_n\}$ are true, is composed of only one product. Therefore, the last term of (5) reduces to:

$$\begin{aligned} \alpha(k,n-k)w_{k+1}p_{k+1}\sigma_{n-k+1}^{\{k+2,\ldots,n\}} &= \alpha(k,n-k)w_{k+1}p_{k+1}\prod_{l=k+2}^{n}w_lp_l \\ &= \alpha(k,n-k)\prod_{l=k+1}^{n}w_lp_l \\ &= \alpha(k,n-k)\sigma_{n-k}^{\{k+1,\ldots,n\}} \end{aligned} \tag{7}$$

Combining the middle two terms of (5):

$$\begin{aligned} & \sum_{j=1}^{n-k-1} \alpha(k,j)\bar{p}_{k+1}\sigma_j^{\{k+2,\ldots,n\}} + \sum_{j=1}^{n-k-1} \alpha(k,j)w_{k+1}p_{k+1}\sigma_{j-1}^{\{k+2,\ldots,n\}} \\ = \; & \sum_{j=1}^{n-k-1} \alpha(k,j)[(\bar{p}_{k+1}\sigma_j^{\{k+2,\ldots,n\}}) + (w_{k+1}p_{k+1}\sigma_{j-1}^{\{k+2,\ldots,n\}})] \\ = \; & \sum_{j=1}^{n-k-1} \alpha(k,j)[(\bar{p}_{k+1}\sum_{\substack{R\subseteq\{k+2,\ldots,n\}\\|R|=j}}\pi_R^{\{k+2,\ldots,n\}}) + (w_{k+1}p_{k+1}\sum_{\substack{R\subseteq\{k+2,\ldots,n\}\\|R|=j-1}}\pi_R^{\{k+2,\ldots,n\}})] \\ = \; & \sum_{j=1}^{n-k-1} \alpha(k,j)[(\sum_{\substack{R\subseteq\{k+1,\ldots,n\}\\|R|=j}}\bar{p}_{k+1}\pi_R^{\{k+2,\ldots,n\}}) + (\sum_{\substack{R\subseteq\{k+1,\ldots,n\}\\|R|=j-1}}w_{k+1}p_{k+1}\pi_R^{\{k+2,\ldots,n\}})] \\ = \; & \sum_{j=1}^{n-k-1} \alpha(k,j)[\sum_{\substack{R\subseteq\{k+1,\ldots,n\}\\|R|=j\wedge k+1\notin R}}\pi_R^{\{k+1,\ldots,n\}} + \sum_{\substack{R\subseteq\{k+1,\ldots,n\}\\|R|=j\wedge k+1\in R}}\pi_R^{\{k+1,\ldots,n\}}] \\ = \; & \sum_{j=1}^{n-k-1} [\alpha(k,j)\sum_{\substack{R\subseteq\{k+1,\ldots,n\}\\|R|=j}}\pi_R^{\{k+1,\ldots,n\}}] \\ = \; & \sum_{j=1}^{n-k-1} \alpha(k,j)\sigma_j^{\{k+1,\ldots,n\}} \end{aligned} \tag{8}$$

Finally, applying equations (6), (7), and (8) to equation (5), we have:

$$
\begin{aligned}
&\sum_{j=0}^{n-k-1} \alpha(k+1,j)\sigma_j^{\{k+2,\ldots,n\}} \\
&= \alpha(k,0)\sigma_0^{\{k+1,\ldots,n\}} + [\sum_{j=1}^{n-k-1} \alpha(k,j)\sigma_j^{\{k+1,\ldots,n\}}] + \alpha(k,n-k)\sigma_{n-k}^{\{k+1,\ldots,n\}} \\
&= \sum_{j=0}^{n-k} \alpha(k,j)\sigma_j^{\{k+1,\ldots,n\}} \\
&= pr(Q \text{ is true}) \qquad \textbf{(by the induction hypothesis)}
\end{aligned}
$$