BELIEF NETWORKS FOR CONSTRUCTION PERFORMANCE DIAGNOSTICS

Brenda McCabe1, Simaan M. AbouRizk2, Member, ASCE, Randy Goebel3

ABSTRACT

Belief networks, also referred to as Bayesian networks, are a form of artificial intelligence that incorporate uncertainty through probability theory and conditional dependence. Variables are graphically represented by nodes whereas conditional dependence relationships between the variables are represented by arrows. A belief network is developed by first defining the variables in the domain and the relationships between those variables. The conditional probabilities of the states of the variables are then determined for each combination of parent states. During evaluation of the network, evidence may be entered at any node without concern to whether the variable is an input or output variable. The probability of each state for the remaining variables, where the state is unknown, is evaluated. An automated approach for the improvement of construction operations involving the integration of belief networks and computer simulation is described. In this application, the belief networks provide diagnostic functionality to the performance analysis of the construction operations. Computer simulation is used to model the construction operations and to validate the changes to the operation recommended by the belief network.

INTRODUCTION

This paper introduces belief networks, a form of artificial intelligence that may be described as a probabilistic-based expert system. Characteristics of belief networks are discussed followed by the evaluation of a singly connected belief network. The next section describes an application that was developed to automatically improve the performance of construction operations by integrating belief networks and computer simulation. The computer simulation is used to model the construction operations and provide performance measures. The belief network evaluates the performance measures, determines the most likely causes of poor performance, and recommends changes to the simulation model. These changes are automatically incorporated into the simulation model and the simulation is rerun. This iterative process was developed to provide automated support during the experimental phase of modeling to the simulationist.

Belief networks were first developed at Stanford University in the 1970s. They fell out of popular research during the 1980s, and have experienced resurgence in the 1990s. In brief, belief networks are a method of representing the dependence and independence amongst a collection of random variables, and to calculate the probabilities of those variables as evidence about their values accumulates. Applications for belief networks, such as diagnostics, forecasting, and

---

1 Assistant Professor, Department of Civil Engineering, University of Toronto, Toronto, ON M5S 1A4
2 Professor, Department of Civil and Environmental Engineering, University of Alberta, Edmonton, AB T6G 2G7
3 Professor, Department of Computing Science, University of Alberta, Edmonton, AB T6G 2H1
decision support have been demonstrated in fields such as medicine and software development (Heckerman et al. 1995).

**Belief Networks**

More specifically, belief networks consist of nodes, representing variables of the domain, and arcs, representing dependence relationships between the nodes. They are directed, acyclic graphs (DAG) that use and exploit Bayes’ Theorem and the concepts of conditional probability. The graphs are directed, as each arc is an arrow that declares that the variable at the head of the arrow is directly dependent on the one at the tail of the arrow. Acyclic means that the arrows may not form a directed cycle or loop in the network. This does not imply that there can only be one path between any two nodes, but it does mean that the path cannot be circular when the direction of the arrows are considered. Nodes that are not directly joined by arcs are either independent, or may be evaluated as conditionally independent through special conditioning.

Network 2 of Figure 1 is singly connected (there is only one path between any two nodes), and because it is singly connected, it has an exact solution. In other words, applying Bayes’ rule to calculate the probability of the value of any node is guaranteed to provide an exact solution. In multiply connected networks, as shown in Network 1 of Figure 1, more than one path exists between any two nodes. If the network is multiply connected, the same application of Bayes’ rule to do the calculations becomes NP-hard (Charniak 1991), which means that there does not exist a non-deterministic polynomial time algorithm to solve the network. In response, many methods have been developed to improve the efficiency of applying the networks, including clustering, (cutset) conditioning, and stochastic simulation (also called random sampling).

Research in artificial intelligence is continuing to improve the representation and search techniques to develop more efficient algorithms for solving the networks.

![Network 1 and Network 2](image)

**Figure 1: Comparison of Two Network Structures**

**Characteristics of Belief Networks**

Belief networks have a built-in independence characteristic that permits evaluation of the network. Consider nodes E and A in Network 2 of Figure 1. The nodes are obviously connected, and therefore have a dependency relationship. However, if the value of a node between them, node C, is known and there is no other undirected path between them that is not blocked by a given or known node, then the two become direction-dependent separated, (d-separated), or conditionally independent of each other given the blocking nodes. Once conditionally
independent, the probability of either node can be evaluated without consideration of the other, thus providing a basis for overall improvements in computation.

Belief networks provide great flexibility in their capacity for accepting input and providing output. For example, suppose an expert system and a belief network were developed for the diagnostics (called diagnostic inference) of equipment breakdowns. When an equipment problem arises, the symptoms would be entered into the two systems, and each would provide the mechanic with the likely cause. Furthermore, assume the two systems performed identically, as far as accuracy is concerned. The typical structure of a diagnostic rule-based system is based on developing rules that record evidence of the form that certain symptoms imply corresponding malfunctions. Such a system can be used to determine the cause of a breakdown given the evidence. However, because of the symmetry of Bayes' rule and its implementation within belief networks, the belief network can also provide information about the symptoms of a malfunction, given the cause of the breakdown (called causal inference) without redeveloping the network. In other words, the belief network has the inherent ability to reverse its logic. In order for the rule-based system to perform the same function, the rules would have to be rewritten in reverse, i.e. that certain malfunctions imply symptoms, before the system could be used in this manner.

The belief network is capable of another mode of operation, called intercausal inference (Henrion et al. 1991). It is used for updating beliefs with the entry of additional evidence. In intercausal inference, new evidence is entered at any point in the network, and the likelihood of the remaining variables is determined and compared to the belief values evaluated before the new evidence was delivered.

Belief networks have some useful characteristics. They have the ability to allow the value of a variable to be entered as a known input, or to evaluate the likelihood of the variable as an output of the system. The ability to adjust variables to be input or output without redesigning the system is not a common characteristic for other forms of artificial intelligence (AI) representation. Rule-based expert systems permit evidence to be entered only at specific points, and the output information is generally fixed. Neural networks have an even less flexible input-output structure in that an entirely new network is required if any variables are added or deleted. Belief networks, on the other hand, may accept evidence at any point in the system, and, likewise, provide output at any point in the system.

Like rule-based expert systems, belief networks may be developed using expert opinion instead of requiring historical data (Charniak 1991). Historical data is required to train neural networks, and although data is not required for genetic algorithms, the development of generic objective functions requires significant resources. The major disadvantage of incorporating expert opinion into belief networks is the general lack of understanding of probability theory. Research has shown that significant errors result from the perception of risk depending on the risk-aversion characteristics of the individual (Tversky and Kahneman 1990).

During development of a knowledge base, the ease of adding variables or states to an existing network is important. The graphical nature of belief networks allows variables to be added or removed without significantly affecting the remainder of the network because modifications to the network may be isolated. Alternatively, additions to neural networks require retraining of the networks. Additions to rule-based expert systems require careful analysis of the rule base to determine the effect of each new rule on the others. This is especially important when there are significantly large numbers of variables in the domain. Belief networks have been found to be
more effective than rule-based expert systems for capturing knowledge when exceptions to the rules are too important to exclude, but too numerous to express explicitly (Chong and Walley 1996).

**Conditional Probabilities and Bayes’ Theorem**

A conditional probability is a probability or likelihood of a variable that is dependent on the state of another variable. Belief networks use Bayes’ Theorem, shown in Equation 1, which follows from the basic conditional probability relationship \( P(A \land B) = P(B \mid A) \times P(A) = P(A \mid B) \times P(B) \).

Bayes’ Theorem may also be used to analyze multiple influences as stated in Equation 2, where the denominator in Equation 2 is an expansion of the denominator in Equation 1. The characters from propositional calculus that have been used in this paper are: \( \mid \) as in \( P(A \mid B) \) denotes a conditional probability of \( A = \text{true} \) given that \( B = \text{true} \), \( \neg \) denotes not, and \( \land \) denotes and.

\[
P(B \mid A) = \frac{P(A \mid B) \times P(B)}{P(A)} \tag{1}
\]

\[
P(B_i \mid A) = \frac{P(A \mid B_i) \times P(B_i)}{\sum_{k=1}^{n} P(A \mid B_k) \times P(B_k)} \tag{2}
\]

**Example Evaluation of a Belief Network**

A singly connected belief network, shown in Figure 2, is presented to illustrate the methods of evaluating a belief network. This network is designed to evaluate remedial action for a truck loading operation. In this operation, the trucks are loaded by loaders. If the loaders are all busy, the trucks form a queue while they wait for the next available loader. The amount of queuing that occurs affects the productivity of the system. For example, if there is no queuing at all, the loader may be idle for long periods of time indicating they are not being fully utilized. However, having several trucks in the queue at all times reduces the efficiency of the truck hauling operation because the trucks are idle. Generally, the optimal productivity occurs when the equipment fleet is balanced i.e. when the number of trucks and the number of loaders are such that neither type of equipment is idle for significant periods. Finally, if too many trucks are used in the project, the condition of the haul road may deteriorate.

**Figure 2: Example Belief Network**

![Belief Network Diagram]

- \( P(TFL) = 0.5 \)
- \( P(TMT) = 0.5 \)
- \( P(AQ \mid TFL \land TMT) = 0.05 \)
- \( P(AQ \mid TFL \land \neg TMT) = 0.35 \)
- \( P(AQ \mid \neg TFL \land TMT) = 0. \)
- \( P(SRS \mid TMT) = 0.15 \)
- \( P(AP \mid AQ) = 0.85 \)
- \( P(AP \mid \neg AQ) = 0.1 \)
All of the variables in this network are binary i.e. contain only two states: true and false. The initial conditional probabilities displayed next to each node show the combinations of the states of the parent nodes and the likelihood that the node is true. The variables TooFewLoaders (TFL) and TooManyTrucks (TMT) affect the variable Acceptable Queuing. If there are too few loaders (TooFewLoaders = true) but the number of trucks is acceptable, (TooManyTrucks = false) then the likelihood that the queuing will be acceptable is only 35%. TooManyTrucks is also a parent of SoundRoadSurface (SRS) in that as the number of trucks increases, the likelihood that the road surface will be damaged will also increase. Finally, the queuing situation will affect the productivity of the system. Note that this is a simplified example and is not intended to provide exhaustive analysis of the causes of poor productivity in this operation.

Now consider a situation where it is known that, while the productivity of the system is acceptable, the road surface has been damaged. The manager wants to know if there are too many trucks in the system. The problem statement is: Find $P(TMT \mid AP \land \neg SRS)$

where $TMT$ represents the true state of the node TooManyTrucks, $AP$ represents the true state of the node AcceptableProductivity (AP), and $\neg SRS$ represents the false state of the SoundRoadSurface node. As all of the information contained in the network relies on conditioning on the parent and not the children, the problem statement must be manipulated until the information required may be read directly from the network. Bayes’ Theorem (Equation 1) is used first to rearrange the problem statement so that it is conditioning on a parent:

$$P(TMT \mid AP \land \neg SRS) = \frac{P(AP \land \neg SRS \mid TMT) \cdot P(TMT)}{P(AP \land \neg SRS)}$$

$P(TMT)$ may be read from the network, but the other two elements require further analysis. Because $TMT$ is known, the two variables $SRS$ and $AP$ are d-separated, and are independent. Therefore, the phrase may be redefined as:

$$P(AP \land \neg SRS \mid TMT) = P(AP \mid TMT) \cdot P(\neg SRS \mid TMT)$$

In order to evaluate $P(AP|TMT)$, the probability of $AP$ must be conditioned on all of the parents of $AP$. Therefore, the node is evaluated for the given information ($TMT = true$), and on all conditions of the remaining parents.

$$P(AP \mid TMT) = P(AP \mid TMT \land AQ) \cdot P(AQ \mid TMT) + P(AP \mid TMT \land \neg AQ) \cdot P(\neg AQ \mid TMT)$$

Note that in the expression $P(AP|TMT\land AQ)$, $AP$ and $TMT$ have become d-separated by $AQ$, and that the probability of $AP$ now only depends upon $AQ$. The term may be expressed as $P(AP|AQ)$, leaving $P(AQ|TMT)$ to be evaluated with all combinations of its parents. Note that $P(\neg AQ|TMT) = 1 - P(AQ|TMT)$.

$$P(AQ \mid TMT) = P(AQ \mid TMT \land TFL) \cdot P(TFL) + P(AQ \mid TMT \land \neg TFL) \cdot P(\neg TFL)$$

The numerator of the problem statement is now in a form whereby the information may be read directly from the network. The denominator may be restated as:

$$P(AP \land \neg SRS) = P(\neg SRS \mid AP) \cdot P(AP)$$

Because $P(\neg SRS|AP)=1 - P(SRS|AP)$, the evaluation of $P(\neg SRS|AP)$ may be simplified to

$$1 - P(SRS \mid AP) = 1 - (P(SRS \mid AP \land TMT) \cdot P(TMT \mid AP) + P(SRS \mid AP \land \neg TMT) \cdot P(\neg TMT \mid AP))$$

where
\[ P(TMT \mid AP) = \frac{P(AP \mid TMT) \cdot P(TMT)}{P(AP)} \]

Again, SRS and AP have been d-separated by TMT, reducing the term \( P(SRS \mid AP \land TMT) \) to \( P(SRS \mid TMT) \), which may be read directly from the network. As the value of \( P(AP \mid TMT) \) has already been evaluated above, all but \( P(AP) \) may be read from the network. AP is now evaluated by conditioning on all combinations of the parents.

\[ P(AP) = P(AP \mid AQ) \cdot P(AQ) + P(AP \mid \neg AQ) \cdot P(\neg AQ) \]

where

\[ P(AQ) = P(AQ \mid TFL \land TMT) \cdot P(TFL) \cdot P(TMT) \]

\[ + P(AQ \mid TFL \land \neg TMT) \cdot P(TFL) \cdot P(\neg TMT) \]

\[ + P(AQ \mid \neg TFL \land TMT) \cdot P(\neg TFL) \cdot P(TMT) \]

\[ + P(AQ \mid \neg TFL \land \neg TMT) \cdot P(\neg TFL) \cdot P(\neg TMT) \]

The network now provides all of the information required to fully evaluate the problem statement. Working upward through the evaluations,

\[ P(AQ) = 0.05 \cdot 0.5 \cdot 0.5 + 0.35 \cdot 0.5 \cdot 0.5 + 0.35 \cdot 0.5 \cdot 0.5 + 0.9 \cdot 0.5 \cdot 0.5 = 0.413 \]

\[ P(AP) = 0.85 \cdot 0.413 + 0.15 \cdot 0.587 = 0.439 \]

\[ P(TMT \mid AP) = 0.29 \cdot 0.5 / 0.439 = 0.330 \]

\[ P(\neg SRS \mid AP) = 1 - (0.15 \cdot 0.330 + 0.9 \cdot 0.670) = 0.348 \]

\[ P(AP \land \neg SRS) = 0.348 \cdot 0.439 = 0.153 \]

\[ P(AQ \mid TMT) = 0.05 \cdot 0.5 + 0.35 \cdot 0.5 \cdot 0.5 = 0.200 \]

\[ P(AP \mid TMT) = 0.85 \cdot 0.2 + 0.15 \cdot 0.8 = 0.290 \]

\[ P(AP \land \neg SRS \mid TMT) = 0.29 \cdot 0.85 = 0.247 \]

and finally,

\[ P(TMT \mid AP \land \neg SRS) = 0.247 \cdot 0.5 / 0.153 = 0.807 \]

Therefore, the manager may conclude with 81% confidence that there are too many trucks in the system.

**Building A Belief Network**

Proper belief network structures may result in a reduction of the number of probabilities required initially and at evaluation time, and may result in better representations of the true system. Poole et al. (1998) outlined the necessary steps for the development of a well-designed belief network. They are:

- Define the relevant variables
- Define the relationship between the variables
- Define the states of the variables. This step requires defining the detail level of the system.
- Define the conditional probabilities of the relationships.

**BELIEF NETWORK APPLICATION**

An automated method of improving simulated construction operations was developed by integrating computer simulation and belief networks (McCabe 1997). The computer simulation model represents the construction operation whereas the belief networks provide expert analysis of the performance of the operation.

The automated process of the prototype is outlined in Figure 3. The user is responsible for the development of the simulation model and for identifying alternative resources. After resource constraints and model parameters are entered, the automated process may begin. The
simulation model parameters are selected and the simulation model is run. The results are communicated to the belief network where the performance indices are calculated and analyzed. If the performance constraints are not all met, then remedial actions are sent from the belief network back to the simulation module. Resource parameters are modified and the simulation is run again. When performance constraints are met, the results are provided to the user.

Figure 3: Overview of Automated Process of Prototype

Computer simulation of construction operations has been used extensively in academic research, but has been used only sparingly in industry. Shi and AbouRizk (1994) pointed out three major reasons for this: the complexity and uniqueness of construction projects; the limited resources available for model development and experimentation; and, the challenges of current simulation environments. Industry practitioners have indicated that simulation has great potential for improving productivity in the construction industry. However, too much effort is required during the experimental stage of simulation modeling. These comments motivated the development of an approach for automated project experimentation and evaluation of simulated operations. A surrogate objective was used for the improvement of the simulated operations: performance. By focusing on performance instead of cost or project duration, the substitution of alternative resources in the simulation model is possible. For example, if the queue length is greater than an acceptable level, then remedial actions may include reducing the number of customers, or decreasing the capacity of the customer to reduce the service time. If minimizing costs were the only objective, the direction for remedial actions to modify the resources would not be as clear.

The belief network is used to evaluate the performance at each resource interaction/queuing location, based upon performance measurement indices. The result is the production of suggestions for remedial actions that may improve the performance. The suggestions are implemented in the simulation model automatically, and the effect of the changes is determined
by running the simulation model with the new parameters. The cycle then begins again with evaluation of the simulation output statistics to determine the performance at each resource interaction location. When the performance constraints are met, the project parameters that resulted in the shortest duration or lowest cost may be retrieved from a database containing the results of each simulation run. The approach is capable of comparing the performance of several construction methods or strategies, to obtain the shortest duration or lowest cost observed overall. A prototype was developed to demonstrate the approach.

The terms server and customer have been used when referring to resources. Borrowed from queuing theory terminology, server generally refers to a resource that is stationary, and provides a service to arriving customers. Example relationships of this type include loaders as servers and trucks as customers, cranes as servers and formwork as the customer, or work space as the server and crews as customers.

PERFORMANCE MEASUREMENT INDICES

In order to improve performance, we must first define it in terms of parameters that can be generated by the system and then operated on by the belief network. To measure performance in the simulation model, five performance indices have been developed. The measures are the queue length index (QL), queue wait time index (QW), customer delay index (CD), server utilization index (SU), and server quantity index (SQ). All of the indices, except the server quantity index, are compared to user-defined lower and upper limits.

QL and QW are measures of the length of the queue and the time spent waiting in the queue. They are compared to the lower and upper limits for the index QL_L and QL_U, or QW_L and QW_U respectively. SU is a measure of the performance of a server, such as a crane or a loader. Again, the utilization is compared to user-defined lower and upper bounds for the utilization index, SU_L and SU_U. SQ compares the number of servers used in the project to the number busy at any point in time throughout the simulation. If there is at least one server idle throughout the simulation run, then the number of servers may be reduced by at least one without affecting the performance of the other resources. CD measures the time a customer spends waiting in queues relative to the cycle time of that customer. The objective is to keep the delays below the user-defined upper limit, CD_U. The lower limit of this index is always zero, therefore it is not explicitly expressed.

The indices are evaluated at each queuing location. Where the value of any of the performance indices do not fall between the lower and upper bounds for that index, a remedial action is required. Evaluation of the performance indices to determine the most appropriate remedial action is performed by the belief network.

The belief network software used in this application was Microsoft© Bayes Networks (MSBN™). MSBN does not permit continuous probability functions, but does provide asymmetric assessment and good error trapping. Asymmetric assessment is a method for structuring the parents of a node into a logical hierarchy, effectively eliminating infeasible combinations of the parent states, or permitting the state of one variable to have precedence over the others. These structures may significantly reduce the number of probabilities that are required to define the network.
Before development begins, therefore, the variables of the system should be determined. Because this is a diagnostic network, the terms cause variables and effect variables will be used, represented by the simulation resource variables and performance indices respectively.

Performance Indices (Effect Variables)
The information or variables that will be input as evidence to the belief network should reflect the current performance of the simulated system. The variables included in the network for this purpose, then, are $QL$, $QW$, $CD$, $SU$, and $SQ$, representing the indices Queue Length, Queue Wait, Customer Delay, Server Utilization, and Server Quantity respectively. The states of the nodes, which identify whether or not the measured performance index is within the acceptable lower and upper limits for that index, are shown in Table 1. State 0 indicates the index is within the acceptable bounds of that index. States 1 and 2 represent the event that the index is either lower than the lower bound, or higher than the upper bound.

Table 1: Performance Variables and Their States

<table>
<thead>
<tr>
<th>Performance Node</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$QL$</td>
<td>$QL \leq QL \leq QL_U$</td>
<td>$QL &lt; QL_L$</td>
<td>$QL &gt; QL_U$</td>
</tr>
<tr>
<td>$QW$</td>
<td>$QW_L \leq QW \leq QW_U$</td>
<td>$QW &lt; QW_L$</td>
<td>$QW &gt; QW_U$</td>
</tr>
<tr>
<td>$CD$</td>
<td>$CD \leq CD_U$</td>
<td>$CD &gt; CD_U$</td>
<td></td>
</tr>
<tr>
<td>$SQ$</td>
<td>$SQ = 0$</td>
<td>$SQ &gt; 0$</td>
<td></td>
</tr>
<tr>
<td>$SU$</td>
<td>$SU_L \leq SU \leq SU_U$</td>
<td>$SU &lt; SU_L$</td>
<td>$SU &gt; SU_U$</td>
</tr>
<tr>
<td>Cost</td>
<td>OK</td>
<td>Optimize</td>
<td></td>
</tr>
<tr>
<td>Duration</td>
<td>OK</td>
<td>Optimize</td>
<td></td>
</tr>
</tbody>
</table>

The objective of the planner, whether it is to minimize the costs and/or to shorten the duration, will affect the type of action that is appropriate to improve the operation. Take the case where the performance indices provide evidence that the queue wait time is too long. If the focus is placed on a shortened duration, then the more likely action would be to increase the number of servers. However, if the cost is a major factor, then reducing the number of customers might be more effective. To accommodate this option, two effect nodes, Cost and Duration, have been added to provide direction toward a specific optimization objective.

The node states may be a mixture of qualitative and quantitative measures, allowing great flexibility in network design. If the state of a variable is known, then it is either true or false. If it is not known, then probability will be evaluated for each state of the variable during the belief network inference mode. A variable is considered to be a possible cause of poor performance when the probability that the state is true is greater than fifty percent e.g. $P(TMS=true) > 50\%$. At this point, the likelihood that the variable is the cause of the poor performance is greater than the likelihood that it is not the cause.

Resource Variables (Causal Variables)
The causal nodes should represent changes to the construction project that are within the control of the planner, and are appropriate for construction. The parameters that may be changed within an operation, without changing the methodology, include the quantity and the capacity of the resources. The causal nodes and their states are shown in Table 2.
There are two methods of describing the variables of conflicting states, such as Too Many Servers and Too Few Servers. First, two variables may be defined, each with a binary state of either true or false. The disadvantage of this design is that it is possible for both of the variables to be evaluated as true, thereby creating a conflict. The advantage of the design is that the variables remain independent.

**Table 2: Causal Variables and Their States**

<table>
<thead>
<tr>
<th>Causal Node</th>
<th>States</th>
</tr>
</thead>
<tbody>
<tr>
<td>Too Many Servers (TMS)</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>Too Few Servers (TFS)</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>Too Many Customers (TMC)</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>Too Few Customers (TFC)</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>Server Too Big (STB)</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>Server Too Small (STS)</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>Customer Too Big (CTB)</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>Customer Too Small (CTS)</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
</tbody>
</table>

An alternative design would entail the use of a single variable for NumberOfServers containing the states “too many”, “too few” and “OK”. In this design, it is impossible for more than one state to have a likelihood of being true greater than 50% because the sum of the probabilities for all of the states of a variable must equal one. However, the states of TooFew and TooMany have now become related because the probability of the states of the node must equal to one. A decision was made to use binary nodes to ensure independence of the evaluated probabilities. The possibility that both TooManyServers and TooFewServers could be evaluated as true is accepted, and will be dealt with in the following manner.

Four pairs of conflicting states are possible: TooManyServers / TooFewServers, TooManyCustomers / TooFewCustomers, ServerTooBig / ServerTooSmall, and CustomerTooBig / CustomerTooSmall. Where conflicting causes for poor performance are suggested by the belief network, the evaluation will be considered inconclusive, and neither cause will be forwarded to the final evaluation. This will not affect other, non-conflicting causes evaluated at that queuing location.

**Conditional Relationships**

The next step in building a belief network is to identify the dependence relationships, shown in Figure 4. The network structure was developed based on guidelines outlined by Russell and Norvig (1995). The authors prescribed a method whereby each variable is ordered to represent dependence of the variables higher in the list, and independence of variables lower in the list. In this case, the list was started by using the causal variables in no particular order followed by the effect variables (performance indices), again in no particular order.

The order of the variables within their groups was not important because they are independent of each other. For example, within the input variables, the length of a queue is not dependent on the server utilization. A server’s utilization on a project may be very high, but the length of the queue at any one location may be high or low. Although it may be shown that the queue length and wait times are correlated, they are evaluated separately because the conditions at each queuing location will be unique. The user is responsible for entering reasonable project constraints. For the causal variables, the number of customers is independent of the capacity of
the customers. However, the number of customers, a cause variable, does affect the server utilization and the queue length, both of which are effect variables.

Figure 4: Belief Network Conditional Relationships

The variables are entered into the network one at a time, in the order that they appear on the list. As each variable is entered, arcs may be placed from any of the existing variables on the network leading to the newly entered variable. Arcs are not allowed from the newly entered
variable to any other node already in the network. In this manner, the network is guaranteed to be acyclic.

Note that the arrows represent a cause and effect relationship between the variables, but the evidence or input to the system will be the effect nodes. The objective of the belief network is to evaluate the cause of any problem that may exist, given the effects. The recommendation for a corrective action to be taken by the simulation model will be the inverse of the cause. For example, if the network was to find that \( P(\text{TooFewCustomers}) > 50\% \) then corrective action may be taken to increase the number of customers in the system.

All of the causal variables affect the queue indices \( QL \) and \( QW \). If the number of resources, whether it is servers or customers, is not correct then the queue length and wait times will be affected. In addition, the capacity of both the servers and the customers will affect the queue indices by affecting the interaction time.

Customer delays are a function of the customer parameters only, just as the server indices are a function of the server parameters. Therefore, the dependence relationships show server variables connected to the nodes \( SU \) and \( SQ \), and customer variables connected to node \( CD \). As the index \( SQ \) is only a gross indication that there are too many servers in the system, only the \( \text{TooManyServers} \) node is connected to it.

If the shortest feasible duration is the primary concern, then the remedial action should tend toward more and/or larger customers instead of fewer and/or smaller servers, for example. While all of the nodes are connected to Duration, there is a bias in the probabilities provided to the network that put more emphasis on the desired nodes than on the others. This biases the resulting recommendations of the network, but in no way limits them. Similarly, the variables that will affect the cost of the project are over-sized resources and having too many of them on site. Therefore, a bias is built into the probabilities of these variables, and the resulting evaluations will bias these output as appropriate.

The blanket assumption that the duration or cost is reduced in the manner discussed without considering the characteristics of the resources in question may not be supportable. Instead of the duration being decreased by using larger customers as assumed, the duration may be increased because, for example, the travel speed of the larger customer is significantly slower than that of a smaller customer. Although the distinct directions to be taken for optimization are very useful, the effect cannot be guaranteed. Therefore, the optimization algorithm will test all four conditions: optimize duration, optimize cost, optimize duration and cost, or focus on performance and not cost or duration. Theoretically, four different optimal states may be found, providing more than one solution that meets the resource and project constraints. The planner may then be presented with numerous options for consideration.

**Conditional Probabilities**

The probabilities were determined by initializing the prior probabilities of the parent nodes (the causal variables) at 35\%, i.e. \( P(\text{TooManyCustomers} = \text{true}) = 35\% \). The conditional probability of each effect variable was evaluated with the beliefs of the authors as to how the various states of the causal nodes in a simulation model would affect the performance indices. Fine-tuning was accomplished by analyzing the resulting probabilities during inference of various situations, and adjusting the conditional probabilities until the desired results were achieved. In addition, fine-tuning could be achieved by comparing the probabilities evaluated using beliefs to probabilities evaluated from data that has been collected.
VALIDATION OF THE BELIEF NETWORK

The belief network was validated by reviewing the probabilities of the causal nodes for all of the combinations of the states of the performance index nodes. Once the results were acceptable, the belief network was considered complete. A small sample of the validation data is shown in Table 3. Performance indices are described by 0, 1 or 2, depending on the state of the index. The output of the belief network is the probability that the causal node is false. Therefore, the recommended remedial actions are those that have a probability less than 0.50, shown in Table 3 as bold and italicized. The rows are organized such that the results are grouped for each instance of the cost and duration combinations.

<table>
<thead>
<tr>
<th>Performance Indices</th>
<th>Causal nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TMS</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0</td>
<td>0.943</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1 0</td>
<td>0.891</td>
</tr>
<tr>
<td>0 0 0 0 0 0 1 0 0</td>
<td>0.841</td>
</tr>
<tr>
<td>0 0 0 0 0 0 1 1 0</td>
<td>0.822</td>
</tr>
<tr>
<td>0 0 0 0 1 0 0 0 0</td>
<td>0.904</td>
</tr>
<tr>
<td>0 0 0 0 1 0 1 0 1</td>
<td>0.880</td>
</tr>
<tr>
<td>0 0 0 0 1 1 0 0 0</td>
<td>0.886</td>
</tr>
<tr>
<td>0 0 0 0 1 1 1 0 0</td>
<td>0.859</td>
</tr>
<tr>
<td>0 2 0 0 1 0 0 0 0</td>
<td>0.943</td>
</tr>
<tr>
<td>0 2 0 0 1 0 1 0 1</td>
<td>0.896</td>
</tr>
<tr>
<td>0 2 0 0 1 1 0 0 0</td>
<td>0.943</td>
</tr>
<tr>
<td>0 2 0 0 1 1 1 0 0</td>
<td>0.890</td>
</tr>
</tbody>
</table>

The first four rows indicate that all of the constraints are met, because all of the variable state indexes representing the performance indices are zero. The first line, where Cost=0 and Duration=0, indicates the event that neither duration nor cost are particularly important. In the second line, the duration is the focus. Cost is the focus of the third evaluation, and finally, both cost and duration are the focus of the fourth evaluation. As the results of the inference by the belief network has shown no probability less than 50%, no recommendations for changes have been made for the first group of data.

In the second group, the customer delay is too high, shown as a value of 1, but all other performance measures are within their acceptable bounds. The evaluation forwarded by the belief network is that the customer is too big, and therefore the remedial action is to reduce the capacity of the customer. The recommendation does not change with the varied evaluation mode of focusing on the cost or duration.

The last group of four data shows different recommendations depending on the focus of the evaluation. In this case, the customer delay index, CD, and the queue wait index, QW, are both too high. The basic evaluation of the belief network is still that the customer is too big. However,
where the cost is the focus, the belief network also found that there were too many customers. Note that the belief network was reluctant to reduce the number of customers in the system, as this may increase the duration. On the other hand, because the server indices SU and SQ are within their bounds, a recommendation to change the servers was not made.

The Prototype
A prototype system has been developed to demonstrate the modeling approach. The software systems used in the prototype are Microsoft® Bayes Networks (MSBN™) Version 1.001 for development and inference of the belief networks, AweSim!™ Version 1.4 by Pritsker Corporation as the simulation language, Microsoft® Visual Basic™ Version 4.0 programming language for integration of the modules, and Microsoft® Access™ for Windows 95 Version 7.0 database for data storage. MSBN, AweSim! and Access all communicate readily with Visual Basic, therefore, these software have been chosen because of ease of integration and the familiarity of this researcher with these systems.

The first screen of the prototype, shown in Figure 5, requires the user to enter the location of the simulation file and the simulation output report file. Note that the scenario number is indicated in the top left corner of the screen. If more than one simulation model or strategy scenario is to be improved and compared, then each scenario may be entered before the automated improvement process begins.

Figure 5: Opening Screen of Prototype

The simulation file is read by the prototype, and required information is automatically extracted. This information is used to provide the subsequent windows with details of the model for the user.

The second screen, shown in Figure 6, prompts the user to enter the user-defined resource performance constraints such as server utilization limits and limits for customer delays. Statistics defined by the user and generated in AweSim! are numbered. To find certain statistics in the output reports, the prototype needs the number related to the duration and cost statistics. A list of the statistics developed in the simulation model has been provided at the bottom of the
screen for convenience. This information was extracted from the simulation file in the previous step.

Figure 6: Screen #2

Shown in Figure 7, the third screen requires the user to link the servers with the customers by entering the number of the AweSim! file in which they interact. Again, the file numbers in which each server interacts is provided to the user for convenience. After the entries are checked, another scenario may be entered or the automated improvement process may begin.

Figure 7: Screen #3
The prototype has been tested with various construction models. Figure 8 shows an example output from the evaluation of two construction scenarios. In this case, the construction scenarios represented two construction strategies of the same method. Note that the lower part of the screen indicates that no solution was found that met all of the user-defined project constraints. However, the user is still presented with solutions.

![Figure 8: Example Output of Prototype](image)

In this case, the lowest cost was observed at simulation Run Number 8 during the evaluation of Scenario 2. The resource parameters that resulted in the lowest cost are shown as: two resources of Type 1, Choice 1; one resource of Type 2, Choice 1; five resources of Type 3, Choice 1; four resources of Type 4, Choice 1; and, twenty-one resources of Type 5, Choice 1. The choice number represents the alternative resource that was used. For example, if two models of loader are available for the construction of the project, then one of the loader models would be Choice 1, and the other would be Choice 2.

Because the lowest cost and shortest duration cases did not meet the user-defined constraints, a mechanism for the user to review the extent of the encroachment was developed. The input and output of each simulation run have been stored in a database. The user is not only able to check the extent of performance failures, but also to check the effects of the changing parameters on the simulation output. A natural improvement of the prototype would be to automatically compare the non-conforming performance with the anticipated performance provided by the user.

**CONCLUSIONS**

In this paper, belief (Bayesian) networks have been introduced and their characteristics described. The evaluation of a network was presented followed by an example application in which an automated approach for improving construction performance was developed including the steps required for the development of a belief network. In this application, the belief network provides diagnostic analysis of the simulated construction performance resulting in the generation of recommendations for remedial actions that may improve the performance.
Because of their inherent flexibility and diagnostic capabilities, it is the opinion of the authors that belief networks have many potential applications in civil engineering including diagnostics and decision support. Research is being undertaken to develop belief network applications to improve the performance of other construction management functions.

REFERENCES